OL® Connect

Enterprise

User Guide

Version 2025.1.1

Objectif Lune

Objectif Lune

OL® Connect

OL Connect (Enterprise edition)
User Guide
Version 2025.1.1
Last Revision: 10/24/2025

All trademarks displayed are the property of their respective owners.

© Upland Software, Inc. Inc. 1994-2025. All rights reserved. No part of this documentation may be reproduced, transmitted or distributed outside of Upland Software, Inc. by any means whatsoever without the express written permission of Upland Software, Inc.. Upland Software, Inc. disclaims responsibility for any errors and omissions in this documentation and accepts no responsibility for damages arising from such inconsistencies or their further consequences of any kind. Upland Software, Inc. reserves the right to alter the information contained in this documentation without notice.

Table of Contents

Welcome to OL Connect 2025.1 **15**

About this documentation ... 16
Additional resources ... 16
OL Connect: a peek under the hood ... 16
The Workflow server ... 17
The OL Connect server ... 18
The OL Connect database ... 18
The File Store ... 19
The engines ... 19
The REST API ... 20
Connect file types ... 20
Log files ... 21
Location ... 21
Name ... 22
Format ... 22

Setup And Configuration **24**

System and hardware considerations ... 24
Antivirus Exclusions ... 24
Database Considerations ... 26
Environment considerations ... 29
Language and Encoding Considerations ... 30
Network considerations ... 31
Performance considerations ... 33
System requirements ... 35
Installation and activation ... 37
Where to obtain the installers ... 37
Installation prerequisites ... 38
User accounts and security ... 39
Installing OL Connect on machines without internet access ... 40
Installation Wizard ... 42
Running OL Connect installer in Silent Mode ... 55
Activating a License ... 64
Migrating to a new workstation ... 67
Information about OL Connect Workflow ... 74
Upgrading ... 74
Server configuration settings ... 94
Connection preferences ... 96
Engine configuration ... 100
Engines preferences ... 109
Language preferences ... 116
Parallel Processing preferences ... 116

Server Clustering ... 124
Known Issues ... 126
Issues with User profile names containing Unicode characters ... 126
DataMapper regular expression issue in 2025.1 ... 127
Script API setting changes in 2025.1 ... 127
Email setting change in 2025.1 ... 128
Dynamic email attachments issue (2024.2.1 specific) ... 128
Adding entries to manually added detail tables (2023.2 onwards) ... 128
Remote translation files issue ... 128
Unicode Variation Selectors issue ... 129
ODBC drivers do not all work with OL Connect ... 129
Incorrect colors in print preview ... 129
Account permission issues ... 129
Incorrect warning when sending email from Designer ... 130
Master Page not affected by style rules for <body> ... 130
Uninstalling ... 143

Walkthroughs **145**

Creating an XML data mapping configuration ... 145
Creating a data mapping configuration ... 146
Extracting data ... 149
Renaming steps, fields and detail tables ... 152
Changing the data type of fields ... 155
Fine-tuning using JavaScript ... 156
What's next? ... 159
Designing an invoice template ... 160
Template requirements ... 160
Creating a template ... 161
Adding personal data ... 165
Adding the dynamic invoice data ... 172
Formatting the Dynamic Table ... 176
Adding a variable address (using a snippet, master page and script) ... 183
The finishing touch ... 185
The finishing touch ... 190
What's next? ... 195
Designing a letter template ... 196
Creating a letter ... 197
Adding personalization ... 200
Working with images ... 205
Working with style sheets ... 209
Using Master Pages ... 214
Adding a barcode ... 215
Using snippets ... 218
What's next? ... 220
Building an automated print process ... 220

Part 1: A simple print process 221
Part 2: Grouping and sorting 234
Part 3 Adding a branch for archiving 240

OL Connect projects **250**

Automation with Workflow 250
Using the REST API 251
Versioning 251
Versioned projects 253
Creating versioned projects 253
Viewing project history 256
Viewing project content 257
Using tags 258
Versioned projects in the cloud 258
Before you start 259
Creating a cloud-based versioned project 260
Keeping the local and online projects in sync 262
Sample Projects 264
Sample Project: Basic Email 265
Sample Project: COTG Timesheets 272
Sample Project: Print Promotional Jobs 279
Sample Project: Print Transactional Jobs 285
Sample Project: Submitting Data with Web Forms 291
Sample Project: Serving a Web Page 296
Workflow processes in OL Connect projects 301
About Workflow processes 301
Common OL Connect Workflow processes 302
OL Connect tasks 303
Email processes with OL Connect tasks 305
Print processes with OL Connect tasks 306
Web processes with OL Connect tasks 309
Capture OnTheGo Workflow processes 310
Batching and commingling 311

OL Connect Automate **317**

Installing and updating nodes 318
Connection settings for OL Connect Server 318
OL Connect resources in Node-RED 319
Flows in an OL Connect application 319
Node-RED: nodes and common techniques 320
Nodes used in OL Connect flows 320
Reading a JSON file 321
Parsing a JSON string 322
Using variables 323

Setting and moving msg properties ... 323
Iterating over items in an array ... 324
Concatenating strings ... 324
OL Connect startup flow ... 325
Triggering a startup flow ... 325
Initializing global variables ... 326
Deploying OL Connect resources ... 326
An OL Connect email flow in Node-RED ... 327
The structure of an OL Connect email flow ... 327
Files used in an OL Connect email flow ... 328
An OL Connect print flow in Node-RED ... 328
The structure of a print flow ... 329
Files used in a print flow ... 331
An OL Connect preview PDF flow in Node-RED ... 331
The structure of a preview PDF flow ... 332
Files used in a preview PDF flow ... 333
An OL Connect web flow in Node-RED ... 333
The structure of an OL Connect web flow ... 334
Files used in an OL Connect web flow ... 335

The DataMapper **336**

DataMapper basics ... 336
Data mapping configurations ... 337
Creating a new data mapping configuration ... 338
Generating a counter ... 341
Opening a data mapping configuration ... 342
Saving a data mapping configuration ... 343
Down-saving a data mapping configuration ... 343
Using the wizard for CSV and Excel files ... 344
Using the wizard for databases ... 346
Using the wizard for JSON files ... 349
Using the wizard for PDF/VT or AFP files ... 351
Using the wizard for XML files ... 353
Providing missing fonts for PDF ... 354
Advanced PCL to PDF options ... 356
Preprocessing AFP input ... 363
Data mapping workflow ... 368
Creating a data mapping workflow ... 368
Testing the extraction workflow ... 370
Data source settings ... 370
Properties and runtime parameters ... 387
Extracting data ... 390
Steps ... 409
The Data Model ... 419
About records ... 420

Creating a Data Model ... 421
Editing the Data Model ... 422
Using the Data Model in templates ... 423
Adding fields and data via Workflow ... 424
Fields ... 425
Detail tables ... 431
Data types ... 436
Data Model file structure ... 446
DataMapper User Interface ... 447
Data Model pane ... 448
The Data Viewer ... 458
Keyboard shortcuts ... 460
Messages pane ... 464
Menus ... 465
Settings pane ... 468
Steps pane ... 486
Step properties pane ... 488
Toolbar ... 527
Welcome Screen ... 529
DataMapper Scripts API ... 531
Using scripts in the DataMapper ... 533
Setting boundaries using JavaScript ... 535
Objects ... 540
Functions ... 576

The Designer **587**

Designer basics ... 587
Features ... 588
Templates ... 589
Context ... 610
Sections ... 612
Print ... 616
Creating a Print template with a Wizard ... 618
Print context ... 624
Print sections ... 629
Pages ... 641
Master Pages ... 648
Media ... 652
Email ... 659
Email template ... 659
Sending email ... 660
Designing an Email template ... 660
Creating an Email template with a Wizard ... 664
Email context ... 667
Email templates ... 669
Email header settings ... 672

Email attachments ... 679
Web ... 682
Creating a Web template with a Wizard ... 683
Web Context ... 687
Web pages ... 688
Forms ... 693
Using Form elements ... 698
Using JavaScript ... 703
Capture OnTheGo ... 707
COTG Forms ... 708
Creating a COTG Form ... 708
Filling a COTG template ... 709
Sending the template to the Workflow tool ... 711
Receiving and extracting data from a COTG Form ... 711
Using COTG data in a template ... 712
Designing a COTG Template ... 714
Capture OnTheGo template wizards ... 717
Using Foundation ... 721
COTG Elements ... 724
Using COTG Elements ... 729
Testing a Capture OnTheGo Template ... 735
Using the COTG plugin ... 739
Dynamically adding COTG widgets ... 742
Saving and restoring custom data and widgets ... 745
Using submitted COTG data in a template ... 749
Capture OnTheGo API ... 751
Content elements ... 761
Element types ... 761
Editing HTML ... 762
Attributes ... 764
Inserting an element ... 765
Selecting an element ... 766
Deleting an element ... 767
Styling and formatting an element ... 767
Barcode ... 768
Boxes ... 823
Business graphics ... 827
COTG Elements ... 835
Date ... 841
Forms ... 842
Form Elements ... 847
Hyperlink and mailto link ... 850
Images ... 852
Table ... 860
Text and special characters ... 865
Snippets ... 867

Adding a snippet to the Resources 867
Creating a snippet 869
Adding a snippet to a section 869
Editing a snippet 869
Renaming a snippet 869
Translating a snippet 870
HTML snippets 870
JSON snippets 872
Handlebars templates 872
Partials 876
Styling and formatting 879
Local formatting versus style sheets 879
Layout properties 880
Styling templates with CSS files 881
Styling text and paragraphs 892
How to position elements 897
Rotating elements 901
Styling a table 902
Styling an image 905
Background color and/or image 908
Border 910
Colors 913
Fonts 917
Locale 921
Spacing 922
Personalizing content 923
Variable data in the text 924
Conditional content 924
Dynamic images and Print section backgrounds 924
Dynamic tables 924
Section clones 925
Snippets 925
Scripts 925
Loading data 926
Variable data in text: expressions 940
Variable data in text: scripts and placeholders 945
Formatting variable data 952
Address Block 955
Showing content conditionally 958
Conditional Print sections 962
Cloning sections 965
Dynamic images 972
Dynamic Table 974
Dynamic Print section backgrounds 994
Personalized URL 996
Handlebars in OL Connect 998

Variable data in text: expressions ... 999
Handlebars expressions ... 1005
Using functions in expressions: Helpers ... 1009
Format Helpers ... 1015
Creating custom Helpers ... 1018
Handlebars templates ... 1023
Partials ... 1027
Handlebars API ... 1030
Using Handlebars templates in a script ... 1032
Preferences ... 1035
Clean-up Service preferences ... 1035
Data Model preferences ... 1039
DataMapper preferences ... 1039
Database Connection preferences ... 1041
Default Locale ... 1044
Editing preferences ... 1044
Email preferences ... 1048
Emmet preferences ... 1049
Engines preferences ... 1053
General preferences ... 1053
Hardware for Digital Signing preferences ... 1053
Language preferences ... 1054
Logging preferences ... 1055
Parallel Processing preferences ... 1057
Print preferences ... 1057
Sample Projects preferences ... 1058
Save preferences ... 1059
Scripting preferences ... 1059
Servers preferences ... 1060
Versioning preferences ... 1063
Web preferences ... 1063
Writing your own scripts ... 1064
Script types ... 1065
Creating a new Standard Script ... 1067
Writing a script ... 1069
Setting the scope of a script ... 1071
Managing scripts ... 1071
Testing scripts ... 1074
Optimizing scripts ... 1078
The script flow: when scripts run ... 1082
Selectors in OL Connect ... 1083
Writing scripts: tips and tricks ... 1089
Loading a snippet via a script ... 1091
Loading content using a server's API ... 1093
Using scripts in Dynamic Tables ... 1095
Control Scripts ... 1098
Post Pagination Scripts ... 1114

Translating templates ... 1120
Translating a template ... 1121
Tagging elements for translation ... 1122
Pluralization ... 1125
Exporting and importing translation files ... 1126
Designer User Interface ... 1128
Dialogs ... 1129
Keyboard shortcuts ... 1223
Menus ... 1229
Panes ... 1241
Toolbars ... 1264
Welcome Screen ... 1270
Print options ... 1272
Settings ... 1307
Job Preset ... 1359
Output Presets ... 1375
Advanced Print Wizard navigation options ... 1401
Settings ... 1422
Designer Script API ... 1479
Standard Script API ... 1481
Control Script API ... 1591
Post Pagination Script API ... 1619

Generating output **1637**

Print output ... 1637
Fax output ... 1638
Email output ... 1638
Web output ... 1639
Generating Print output ... 1639
Generating Print output from the Designer ... 1639
Generating Print output from Workflow ... 1641
Print settings in a template ... 1642
Aborting content creation ... 1642
Print using standard print output settings ... 1643
Print Presets ... 1644
Print using Advanced Printer Wizard ... 1651
Adding print output Models to the Print Wizard ... 1652
Banner pages ... 1653
Splitting printing into more than one file ... 1657
Print output variables ... 1659
Generating Fax output ... 1670
Generating Tags for Image output ... 1672
Generating Email output ... 1673
Before generating Email output ... 1674
Generating Email output from Connect Designer ... 1676

Generating Email output from Workflow ... 1677
Testing Email output for different email clients ... 1677
Aborting content creation ... 1677
Using an ESP with OL Connect ... 1678
Generating Web output ... 1682
Web output settings in the Web context and sections ... 1683
Attaching Web output to an Email template ... 1683
Generating Web output from Workflow ... 1684
Aborting content creation ... 1685
Optimizing a template ... 1685
Scripts ... 1685
Images ... 1686
Runtime parameters ... 1687

Print Manager **1689**

Print Manager Introduction Video ... 1690
Print Manager usage ... 1690
Print Manager ports ... 1690
Print Manager Interface ... 1690
Menu options ... 1691
Printers pane ... 1691
Printer Queue pane ... 1698
Job Queue pane ... 1707
Messages pane ... 1709
Job History pane ... 1710
Printer Capabilities pane ... 1711
Printer Status pane ... 1712
Print Manager Preferences ... 1713

OL Connect Release Notes **1722**

OL Connect Release Notes 2025.1.1 ... 1722
License Update Required for Upgrade to OL Connect 2025.x ... 1723
Backup before Upgrading ... 1723
Overview ... 1724
OL Connect 2025.1.1 Improvements ... 1725
OL Connect 2025.1 Improvements ... 1726
OL Connect 2025.1 Designer Improvements ... 1728
OL Connect 2025.1 DataMapper Improvements ... 1732
OL Connect 2025.1 Output Improvements ... 1734
Workflow 2025.1 Improvements ... 1738
Known Issues ... 1743
Previous Releases ... 1743
OL Connect Release Notes 2024.1 ... 1743
OL Connect Release Notes 2023.2 ... 1759
OL Connect Release Notes 2023.1 ... 1775

OL Connect Release Notes 2022.2 1794
OL Connect Release Notes 2022.1 1808
OL Connect Release Notes 2021.2 1823
OL Connect Release Notes 2021.1 1833
OL ConnectRelease Notes 2020.2 1843
OL Connect Release Notes 2020.1 1855
OL Connect Release Notes 2019.2 1867
OL Connect Release Notes 2019.1 1881
OL Connect Release Notes 2018.2 1893
OL Connect Release Notes 2018.1 1913
OL Connect Release Notes 1.8 1931
OL Connect Release Notes 1.7.1 1950
OL Connect Release Notes 1.6.1 1971
OL Connect Release Notes 1.5 1984
OL Connect Release Notes 1.4 1995

Support **2004**

Additional resources 2004
Report an issue 2004

Legal Notices and Acknowledgments **2005**

End User License Agreement (EULA) 2015
Copyright Information 2015

Welcome to OL Connect 2025.1

OL Connect is a series of tools designed to optimize and automate customer communications management. They work together to improve the creation, distribution, interaction and maintenance of your communications.

Design

The OL Connect **Designer** allows you to design:

- output for print, email and web within a single **template** (see "Designer basics" on page 587)
- **data mapping configurations** to extract data from a data source and convert it into a format that can be shared between different templates (see "DataMapper basics" on page 336)
- **print presets** - job presets and output presets - that are applied outside the design phase, making template printing independent of the device (see "Print Presets" on page 1644).

The Designer's interface is easy to use. More advanced users may use native HTML, CSS and JavaScript.

Automate

OL Connect includes a process automation server, called **Workflow**. Workflow is a tool to automate the processing, distribution and printing of your business documents. It is also capable of servicing response form web pages and email to provide interactive business communications. For a brief description of the most common Workflow processes in OL Connect projects, see "Workflow processes in OL Connect projects" on page 301.
For the user guide of Workflow, see Workflow's Online Help.

Meanwhile, another automation tool is in the making: "OL Connect Automate" on page 317. OL Connect Automate adds a range of nodes to the Node-RED palette, allowing Node-RED to communicate with the OL Connect server and access the OL Connect Database and File Store, or perform common tasks in OL Connect solutions.

Capture OnTheGo

OL Connect can create documents for tablets and mobile devices that run a free **Capture OnTheGo** App. Users with a Capture OnTheGo subscription can then download documents to their own devices, interact with them and send the captured data back to OL Connect for conversion into additional documents or workflows.

For more information about the application refer to these websites: Capture OnTheGo and Capture OnTheGo Help Center.

About this documentation

This online documentation covers **OL Connect** version 2025.1, **Professional** and **Enterprise** editions.

You can download the PDF version of this user guide: OL-Connect-Enterprise-2025-1-User-Guide.pdf.

Note: Since we are always looking for new ways to make your life easier, we welcome your questions and comments about this documentation. Shoot us an email at doc@ca.objectiflune.com.

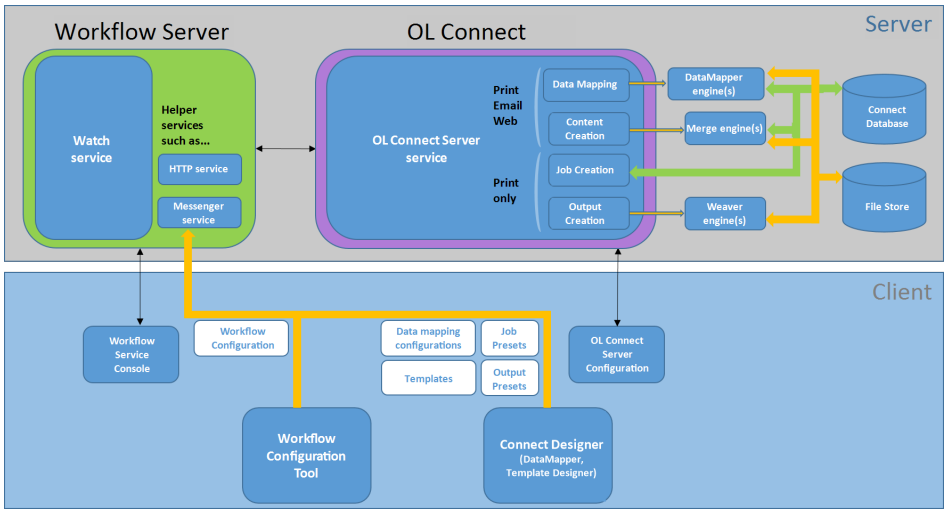
Additional resources

- The OL Resource Center. Read our latest and greatest blog posts, follow hands-on tutorials and visit the user forum for tips and tricks to help you get the most out of OL Connect.
- Step-by-step walkthrough guides introduce you to the functionality of OL Connect at your own pace.
- The step-by-step "Letter" walkthrough guide introduces you to the basic functionality of OL Connect.
- The OL Connect REST API service provides programmatic access to areas such as output creation, data management and file store operations.
- The Upland Support team is dedicated to helping our customers succeed by providing timely resolutions to your issues and questions. You can contact support by phone or via the Community.
- In the Upland Community, you can find information on product releases, submit and track support cases, sign up for notifications, post and reply to forum topics, and learn about training and webinars.

OL Connect: a peek under the hood

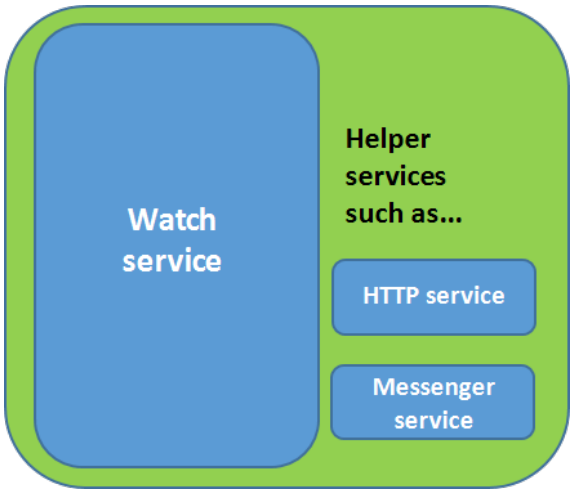
OL Connect consists of visible and invisible parts.
The visible parts are the tools you use to create templates, data mapping configurations, and Print Presets (the Designer/DataMapper), and to create Workflow configurations (the Workflow configuration tool).
The invisible parts process the OL Connect job to provide the actual output. This topic introduces you to those parts.

Here's a simplified, graphical representation of the architecture of OL Connect. The components described below are all located in the 'Server' part.



The Workflow server

The Workflow server (also referred to as the 'Watch service') executes processes independently, after a Workflow configuration has been uploaded and the services have been started. The Workflow server can run only one configuration at a time.



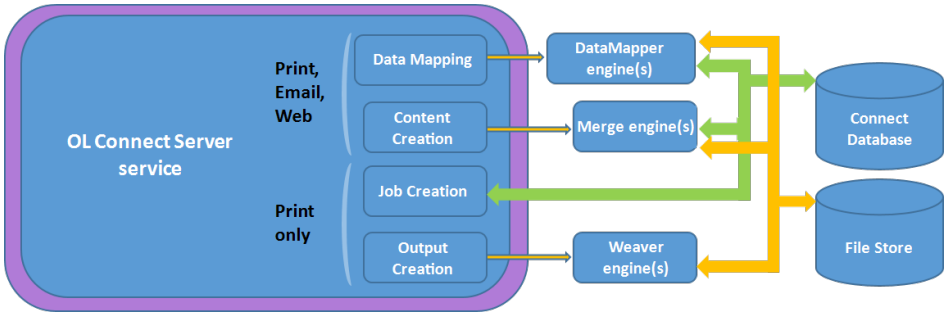
There are a number of services related to Workflow. The OL Connect **Messenger service**, for example, receives the files sent to Workflow from the Designer and the Workflow configuration tool.
The Workflow Service Console lets you start and stop the different services, except the Connect server, and see their log files (see Workflow Service Console).

Note that Workflow isn't limited to Connect functionality. It was originally developed as part of the PlanetPress Suite. Many of the plugins in the Workflow configuration tool are older than Connect. They were left in for compatibility reasons, even though they aren't all useful or usable within Connect. However, the Connect plugins cannot be used with the PlanetPress Suite software.

The OL Connect server

As opposed to the Workflow server, the Connect server was designed to be used only with Connect. The OL Connect server performs several different tasks, all of which are related to Connect content and content management:

It communicates with the Workflow service (with the OL Connect plugins, specifically) and with the Designer when output is generated from the Designer. It creates records (by extracting data from a data source using a data mapping configuration), and jobs. It communicates with the engines (see below) in order to make them create content items and output (spool) files.



The OL Connect server is one of the components that has to be installed with Connect (see "Installation Wizard" on page 42).
In the Workflow Configuration Tool preferences you have to set the OL Connect server settings to enable Workflow to communicate with the server (see Workflow Preferences).

The **OL Connect Server Configuration** tool lets you change the settings for the Connect server, the engines and the service that cleans up the database and the file store. These settings can also be made in the preferences of the Designer.

The OL Connect database

The OL Connect database is the database back-end used by OL Connect itself when processing jobs. It can be either the MariaDB instance provided by the OL Connect installer, or a pre-existing (external) instance (see "Database Considerations" on page 26).

All generated items (records, content items etc.) are stored in this database. They can be

used by the next task in the same process or in a process that runs later, making it possible to commingle Print jobs, for example.

> **Note:** Email content items are not stored in the OL Connect database.

A clean-up of the database is performed at regular intervals in accordance with the settings (see "Clean-up Service preferences" on page 1035).

The File Store

OL Connect has its own File Store which it uses for transient files.
The Clean-up service takes care of removing obsolete files when those files are not marked as permanent (see "Clean-up Service preferences" on page 1035).

> **Tip:** The File Store is accessible for customer implementations. The Workflow configuration tool implements three tasks that allow you to Upload, Download and Delete files in the OL Connect File Store. The files can be accessed through the REST API, which means web portals could potentially access the files directly without having to go through a Workflow process (see The Connect API).

The engines

DataMapper engines. A DataMapper engine extracts data from a data file. The number of DataMapper engines is configurable ("Engines preferences" on page 109).

Merge engine/s. A Merge engine merges data with a template using the scripts in the template, in order to create content items.
The number of merge engines is configurable (see "Engines preferences" on page 109): it can be increased depending on the capacity of the machine that runs the solution (see "Performance considerations" on page 33).

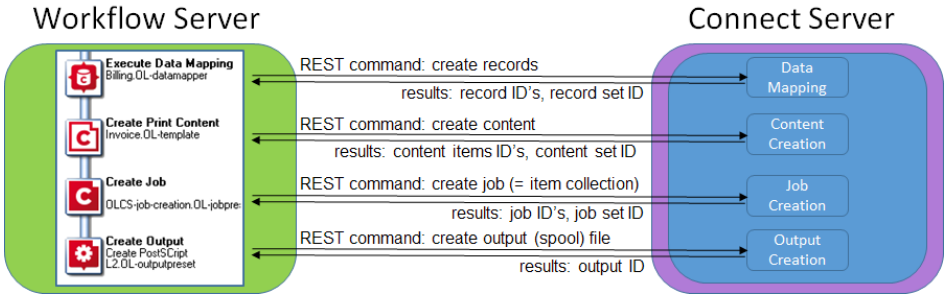
Weaver engines. A Weaver engine creates Print output from Print content items. It takes the settings made in Print Presets or in the Print Wizard into account.
The number of Weaver engines is configurable as well (see "Engines preferences" on page 109).

> **Note:** The Proof Print function doesn't print via the OL Connect Server; it uses the Designer's internal engines.

The REST API

The OL Connect server receives REST commands (see The Connect API), normally either via the Workflow service or from the Designer. This design allows the OL Connect functionality to be used by other applications. The server forwards the commands to the appropriate engine and returns the results to the caller. The results are the id's of the items (records, content items, job etc.) that are stored in the OL Connect database (see below). All OL Connect tasks except the Create Web Content task integrate the results in the Metadata in Workflow.

The figure below shows the communication between OL Connect tasks and the OL Connect server in a Print process.



Printing and emailing from the Designer

To print or send email from within the Designer, the OL Connect service has to be running. The service is started automatically when the Designer starts, but it may not be running if the OL Connect Server and the Designer are installed on different computers. The OL Connect service can be found on the Services tab in the Task Manager.
For a proof print the OL Connect server is not used. Proof printing is always done locally, by the Designer.

Connect file types

This article describes the different file types that are related to OL Connect and its different modules. These are files that are generally transferable between machines, can be sent via email or other means.

- **.OL-template**: A Designer Template file (see "Templates" on page 589). Is linked to a data mapping configuration by default, but not necessarily.
- **.OL-datamapper**: A data mapping configuration file, which can include sample data (excluding database source files such as MariaDB, MySQL, Oracle, etc). See "Data mapping configurations" on page 337.

- **.OL-datamodel**: A data model file which can be imported or exported into either a data mapping configuration or a template. Contains a list of fields and their data type (date, currency, string, etc). See "The Data Model" on page 419.
- **Print Presets** are used when generating a job, ready for output, from Designer (see "Generating Print output from the Designer" on page 1639) or through Workflow (see "Print processes with OL Connect tasks" on page 306).
    - **.OL-jobpreset**: A Job Preset (or Job Creation Preset) file has settings for data filtering and sorting, grouping (groups are used to split the print output), adding metadata fields, and finishing options that override the template's Media and Section settings. (See "Job Presets" on page 1646.)
    - **.OL-outputpreset**: An Output Preset (or Output Creation Preset) file is used to generate the actual print output in the appropriate format (AFP, IPDS, PCL, PDF, PostScript and the like). It includes print settings such as impositioning (N-up, cut & stack), inserter marks, tray settings, separation etc. (See "Output Presets" on page 1648.)
- **.OL-package**: A transfer file used to package one or many of the above files (the data model being part of both the template and the data mapping configuration). Created by using the **File > Package** dialog. (See "Package dialog" on page 1171.)
- **.OL-script**: One or more Designer scripts. Scripts personalize the output of a template. They are either added via wizards (see "Personalizing content" on page 923) or self-written (see "Writing your own scripts" on page 1064). They can also be imported or exported from the Scripts pane in Designer when a template is open.
- **.OL-printerdef**: A Printer Definition File is used by the Output Preset (see "Output Presets" on page 1375) to determine what type of output to produce. These definition files are generated by an internal application that is not currently distributed outside of OL, but the definition files themselves can be provided.
- **.OL-workflow**: A Workflow file used by OL Connect Workflow. Equivalent to .pp7 files (they are, in fact, essentially the same format), containing the processes and such used by Workflow.

Log files

Location

Connect's log files are located in the following folder: `C:\Users\[UserName]\Connect\logs`.
Where `[username]` is replaced by the appropriate Windows user name.

Tip: Actually, the path may not begin with 'C:\Users', as this is language-dependent. On a French system, for example, it would be 'C:\Utilisateurs'.

Type `%userprofile%` in a Windows File Explorer and press Enter to open the actual current user's home directory.

Every time output is generated, the Designer and/or Connect Server and any engines involved in the operation produce their own log files.

Each component writes its log files to a dedicated subfolder of the Log folder. The following folders will likely be of the most benefit.

- Merge engines write to the `logs/Mergeengine` folder
- Weaver engines to the `logs/Weaverengine` folder
- DataMapper engines to the `logs/Datamapperengine` folder
- Server to the `logs/Server` folder

Note that actions of the Cleanup service are only logged in the Server's log file. (See also: "Clean-up Service preferences" on page 1035.)

Note: Workflow services write their logs to an entirely different location. See Accessing the Workflow logs.

Tip: For more information about Connect's architecture, see: "OL Connect: a peek under the hood" on page 16.

Name

The name of a log file consists of the **component**'s name, a **time stamp** and the Windows **process ID**.
The Windows process ID is needed since multiple engines of the same type could be running at the same time, depending on the engine configuration (see "Engine configuration" on page 100) and the size of a job.

Format

The log files are **text** files that can be opened in any text processor. Each log file contains a header section containing a snapshot of the current working environment (Connect version, operating system, system locale settings, disk and RAM information, and such like).

The logging **level** is configurable, as well as the exact **format** - the "logging pattern" - of the individual log messages.

- The Designer's logging preferences are set via the **Designer**; see: "Logging preferences" on page 1055.
- The Connect Server's settings are maintained by the **Connect Server Configuration** utility tool; see "Server configuration settings" on page 94. The logging level that is set applies to the Server as well as the engines.

The Designer's log messages are also displayed in the Messages pane (see "Preflight Results and Messages" on page 1248). The Log Filter allows you to decide what kind of events may be shown in the Messages pane (see "Log Filter" on page 1248). That doesn't change the actual logging level of the Designer, however; it only filters the log file.

Setup And Configuration

This chapter describes the OL Connect installation and the different considerations that are important in regards to the installation and use of OL Connect.

- "System and hardware considerations" below
- "Installation and activation" on page 37
- "Known Issues" on page 126
- "Server configuration settings" on page 94
- "Uninstalling" on page 143

System and hardware considerations

There are a variety of considerations to be aware of. These are documented in the following pages:

- "System requirements" on page 35
- "Database Considerations" on page 26
- "Environment considerations" on page 29
- "Known Issues" on page 126
- "Language and Encoding Considerations" on page 30
- "Antivirus Exclusions" below
- "Performance considerations" on page 33

Antivirus Exclusions

The information on this page is designed to assist IT managers and IT professionals decide what anti-virus strategy to follow with consideration to OL Connect and their internal requirements and needs. This page describes the mode of operation and the files and folders used by OL Connect as well as the files, folders and executables that are recommended to be ignored for best possible performance and to avoid issues caused by antivirus file locks.

IT managers and IT professionals then may decide the anti-virus strategy to follow for their internal requirements and needs depending on the statements outlined herein.

Directories and folders

All Connect applications are installed under an arbitrarily selectable main folder. If the default installation folder options were used, this installation folder would be

"%PROGRAMFILES%\Objectif Lune\OL Connect".

The installation folder will hold all the executable files and other files and folders required for the operation of the whole product suite. All these files and folders remain static after installation. It depends upon the company virus protection strategy, if such files and folders will be monitored or not.

We do, however, recommend that the following file or folders be ***excluded*** from antivirus protection.

Connect Service

The Connect Service is run via the executable file **ServerService.exe**. This file has been reported as causing issues with some antivirus packages, so we recommend adding this file to the antivirus exclusion list, if possible.

The executable will be installed to the installation target "**Connect Server**" sub-folder. The full installation folder can be found by entering "%PROGRAMFILES%\Objectif Lune\OL Connect\Connect Server" in Windows Explorer.

AFP Input (Enterprise edition only)

Performance issues have been reported with the **AFP Input** option under Windows Server versions from Windows Server 2012 onwards. The issues have been specifically associated with Windows Servers running Windows Defender, but the performance degradation might also be encountered when using other Antivirus applications.

Consequently, we recommend that an exclusion be made for the *afp2pdf.exe* executable file in your Antivirus application.

The *afp2pdf.exe* file is stored in a subfolder under the installation folder. If the exact installation folder name is required, enter the standardized system variable %PROGRAMFILES% in Windows Explorer and then search for *afp2pdf.exe*.

Working folders

Working folders for Connect are created and used on a per-user-basis under the respective user's profile folder, accessible on Windows with the standardized system variable %USERPROFILE% in the subfolder "Connect". Working folders are:

- "`%USERPROFILE%\Connect\filestore`": This folder will hold non-intermediate files for the operation of Connect. Files in this folder will be used frequently, but not with a high frequency. Supervising this folder with a virus protection system should not have too much of an impact on the speed of the whole Connect suite.
- "`%USERPROFILE%\Connect\logs`": As the name implies, log files are created and updated here. These log files are plain text files. Virus protection may have an impact on the speed of the whole Connect suite.
- "`%USERPROFILE%\Connect\temp`": Storage folder for temporary data, usually intermittent files in multiple folders. Virus protection on this folder and its subfolders may have a serious impact on the performance of Connect.
- "`%USERPROFILE%\Connect\workspace`": Usually containing settings and helper files and folders. Supervising this folder with a virus protection system should not have too much of an impact on the speed of the whole Connect suite.

Database 1

Depending on the components installed, a database instance is created in a folder called "**connect.database**" under the Windows system temp folder. This folder is accessible via the standardized system variable `%TMP%`. Usually, folders holding such temporary files and folders should be excluded from a virus protection, because this influences the overall performance of the whole system at all. However the responsible person for the computer protection has to decide about the monitoring of such temporary folders following the company guidelines.

Database 2

Another database instance for Connect will be hold and used under the folder, which is intended to hold data, accessible by and for all users. The path to this folder is stored in the standardized system variable `%PROGRAMDATA%`. The Connect database instance is located in the subfolder "**Objectif Lune\OL Connect\MariaDB**".

As this database will be in extremely strong usage, virus protection on this folder and its sub-folders may have a **serious** impact on the performance of Connect.

Database Considerations

This page describes the different considerations and pre-requisites for the database back-end used by OL Connect, whether using the MariaDB instance provided by the installer, or pre-existing (*external*) instance.

Using the MariaDB Instance from the Installer

The MariaDB Instance provided in the "Installation Wizard" on page 42 is already pre-configured with options to provide the most stable back-end setup.

Installing Connect using an existing MySQL instance

If MySQL Server is already present and you wish to use it, the following should be taken into consideration:

- The minimum supported MySQL version is MySQL 5.6.
- The MySQL account must have access to all permissions using the GRANT Command, including creating databases.
- The database configuration must include the following options:
  - **max_connections = 200** : OL Connect uses *a lot* of database connections. This number ensures that even in high volume environments, enough connections will be available.
  - **max_allowed_packet = 500M** : In some implementations, especially when using Capture OnTheGo, large packet sizes are required to allow transferring binary files. This substantial packet size maximum setting ensures that the data received by OL Connect will be able to be stored within the database.
  - **character-set-server = utf8 , collation-server = utf8_unicode_ci , default-character-set=utf8** : These indicate database support for UTF-8/Unicode.
- The database configuration must allow the use of mixed case table names.
  This is particularly an issue on Linux MySQL installations.
- The SQL instance must be open to access from other computers. This means the bind-address option should not be set to 127.0.0.1 or localhost.

Caution: If you chose **not** to install the supplied MariaDB database, and instead opt for using a pre-existing (*External*) database then you yourself must ensure that the *External* database is accessible to Connect.

Upland Software, Inc. will take no responsibility for setting up database connections to any but the supplied MariaDB database.

See "Database Considerations" on the previous page for more information about setting up *external* databases.

Options available within the installer:

- The Configuration page for the local MySQL is displayed.
- MySQL settings are pre-filled with default values if no existing MySQL database configuration is found.
- MySQL settings are pre-filled with existing database configuration settings, if they point to a MySQL database type.

Installing Connect using an existing Microsoft SQL Server instance

If Microsoft SQL Server is already present and you wish to use it, the following should be taken into consideration:

Caution: If you chose **not** to install the supplied MariaDB database, and instead opt for using a pre-existing (*External*) database then you yourself must ensure that the *External* database is accessible to Connect.

Upland Software, Inc. will take no responsibility for setting up database connections to any but the supplied MariaDB database.

See "Database Considerations" on page 26 for more information about setting up *external* databases.

Note: Since OL Connect version 1.6 the minimum required version of the MS SQL Server is **SQL Server 2012**.

- When MS SQL is selected, the default values for root user are **sa** and **1433** for the port.
- If database settings from a previous OL Connect installation are found, the pre-exising settings will be displayed for the matching database type. For MS SQL settings, this will only work if they were created with Server Config Tool 1.5.0 or later, or the Installer for OL Connect 1.6.0 or later.
  If the database type is changed in the Installer configuration page, the default values for this database type will be displayed.
  If the pre-existing database settings are set to Hsqldb, the default database type selection will be MySQL.
- Selected database settings are stored in the preferences, and can be found in this file: *C:\ProgramData\Objectif Lune\OL Connect\.settings\ConnectHostScope\com.objectiflune.repository.eclipselink.generic.prefs*

When modifying Connect

- If the local MariaDB is removed from an installation, the Database Configuration page will offer additionally the **Microsoft SQL Server** database type with respective default values.
- If local MariaDB is added to an installation, the usual MariaDB Configuration page with default values will be displayed.

If the user has installed the Installer Supplied MySQL (2021.2 or earlier) or MariaDB (2022.1 and later) and then switches to an *external* Microsoft SQL by using the Server Configuration Tool, the supplied local database cannot be switched off. By design the installer adds a service dependency between Connect Server and the supplied MariaDB \ MySQL service.

To remove this dependency the user needs to do the following

1. Have a foreign Microsoft SQL running, ready for use with Connect Server.
2. Use the **Server Configuration Tool** "Database Connection preferences" on page 1041 to switch the database to Microsoft SQL.
3. Re-start the Connect Server Service, so that the modifications become active.
4. Counter check that everything is working properly with Microsoft SQL.
5. Open a command-line prompt with full administration rights.
6. Enter the command `sc config OLConnect_Server depend= /`. This removes the dependency.

   Please be aware: The key word `depend` must be followed immediately by the equal sign, but between the equal sign and the forward slash there must be a space.

   Additional information can be found here: http://serverfault.com/questions/24821.
7. After the dependency has been removed, it is possible to stop the supplied MariaDB \ MySQL service (OLConnect_MySQL).

Environment considerations

Remote Desktop Support

Tests have demonstrated that OL Connect can be used through Remote Desktop. It is however possible that certain combination of OS could cause issues. If problems are encountered, please contact OL Support and we will investigate.

OL Connect 1.3 and later have been certified under Remote Desktop.

Antivirus Considerations

- Antivirus software may slow down processing or cause issues if they are scanning in temporary folders or those used by OL Connect. Please see "Antivirus Exclusions" on page 24 for more information.
- Antivirus software might interfere with installation scripts, notably a VBS script to install fonts. McAfee, in particular, should be disabled temporarily during installation in order for MICR fonts to install and the installation to complete successfully.

Windows Search Indexing Service

Tests have concluded that the Windows Search service, used to provide indexing for Windows Search, can interfere with Connect when installing on a virtual machine. If the installation hangs during the last steps, it is necessary to completely disable this service during installation.

- Click on Start, Run.
- Type in **services.msc** and click OK.
- Locate the **Windows Search** service and double-click on it.
- Change the **Startup Type** to **Disable**, and click **Stop** to stop the service.
- Try the installation again.
- Once complete, you may re-enable the service and start it.

Commandline switches and .ini entries

OL Connect is intended to work stably and reliably, based on Java and the Eclipse framework. To ensure this reliability and robustness, many Java and Eclipse parameters have been tested and tuned, which is reflected in the respective .ini entries and the used command line switches. A collection of valuable settings has been elaborated and found its entry in OL Connect “good switches list” (called the “whitelist”).

The protection of the end user’s system is one of our main goals and therefore we have implemented a very strict verification mechanism, which ensures, that only these whitelisted ini entries and command-line switches are accepted, when one of Connect components is started and run. Please be therefore advised, that any non-whitelisted ini entry or command-line switch will be accepted and will - if tried to be used - lead to the respective application’s “sudden death”. If you should encounter such a behavior then please double-check your Connect log file/s for respective entries.

Language and Encoding Considerations

Please note the following considerations:

- **Language**:

  OL Connect is currently offered in several languages. You can switch between these languages via the Preferences dialog. The current languages include:

  - English
  - French
  - German
  - Spanish
  - Italian
  - Korean
  - Portuguese
  - Chinese (Simplified)
  - Chinese (Traditional)
  - Japanese.

  The default language is English.

  The OL Connect help system (this document and the online help) is currently only available in English and (for the biggest part) in French.

- **Encoding**:

  Issues can sometimes be encountered in menus and templates when running OL Connect on a non-English operating system. These are due to encoding issues and will be addressed in a later release.

Network considerations

The following should be taken into consideration in regards to network settings and communications.

- If a local proxy is configured (in the **Internet Explorer Options** dialog), the option **Bypass proxy server for local addresses** must be checked, or some features depending on local communication will not work.

Firewall/Port considerations

The following describes all of the ports that can be used by an OL Connect solution. IT staff may decide the firewall strategy to follow for their internal requirements and needs depending on the statements outlined herein.

	Listens on port #	Destination port #	Type	Comment
FTP Input/Output		**21**	TCP	
Send Email plugin		**25**	TCP	Default SMTP port
SMTP Input plugin	**25**, or **587** when encryption is enabled		TCP	
Email Input plugin		110	TCP	Default POP3 port
Simple Network Management Protocol (SNMP)	161 & 162		UDP	Used by Print Manager, which is only available in OL Connect Enterprise edition.
HTTPS Server	**443**		TCP	
LPR		515	TCP	
LPD	515		TCP	
Secure Email Output plugin		**587**	TCP	
Secure Email Input plugin		**993**	TCP	
Microsoft SQL Server	**1433**		TCP+UDP	
Print Manager	2020		TCP	Used by Print Manager, which is only available in OL Connect Enterprise edition.
MariaDB/MySQL	**3306**		TCP+UDP	
IPDS printer connection	**5001**		TCP	Used by Print Manager, which is only available in OL Connect Enterprise edition.
Messenger	**5863/5864**	**5863/5864**	TCP+UDP	Used for inter-module communication
HTTP Server	**8080**		TCP	
NodeJS HTTPS Server	**8443**		TCP	
HyperSQL	**9001**		TCP	
NodeJS Server	**9090**		TCP	
Telnet	**9100**		TCP	
Connect Server	**9340**		TCP	Used for Connect REST API
Connect Server	**9350**		TCP	Dedicated internal connection for inter-process communication (i.e. between engines)

- Port numbers in **bold** type are user configurable.
- Port numbers in **bold underlined** type are based on the type of database used.
- Some of the ports listed above may also be used by other modules.
- User-configurable modules may use other ports entirely, depending on the settings defined by the end user. A few examples:
  The **Create Email Content** plugin will use the port defined in the Mail host setting in

Workflow's OL Connect Preferences.
The ports used by the **HTTP Client Input** task, **Legacy SOAP Client** and **SOAP Client** plugin depend on the configured URL.

Performance considerations

In order to get the most out of OL Connect, it is important to determine how best to maximize performance. The following guidelines will be helpful in extracting the best performance from OL Connect and they give a rough indication when it would be useful to start looking into hardware upgrades.

Performance analysis details

OL Connect's output speed is limited to a certain number of output items (web pages, emails, or printed pages) per minute. What the **maximum** total output speed will be is determined by your license (see "Speed quota: Pages Per Minute" on page 102).

To get an indication of the **actual** Print output speed, output a Print template to a AFP (Enterprise edition only), PDF or PS file, using one single Weaver engine and the maximum target speed per job (see "Parallel Processing preferences" on page 116).
Next, open the log file of the Weaver engine. By default, the log files are located in this folder: `C:\Users\[username]\Connect\logs\WeaverEngine`, where `[username]` is your own Windows user name.
Search the log file for "PPM" (pages per minute). Repeat this a few times to determine the average output speed.
Likewise, the output speed for an Email or Web template can be found by running it with one Merge engine and the maximum target speed per job. In the Merge engine's log file, search for "PPM".

If your jobs are not running at the licensed speed, there may be several ways to improve performance, as described below. Make sure to address all issues mentioned in this topic before deciding that you need to invest. Note however that it is not guaranteed that the licensed speed can be achieved with *any* job. Creating output for templates with very complex scripts or complex graphics resources will take a certain amount of time, even on high-end hardware.

Improving performance beyond what can possibly be reached by using the methods described below requires purchasing a Performance Pack (see Performance Packs).

For advice please contact your local sales office (see Upland Objectif Lune's Contact page).

Engine configuration

As explained in another topic ("OL Connect: a peek under the hood" on page 16) the Connect Server cooperates with different engines to handle specific tasks.

A **DataMapper engine** extracts data from a data file. A **Merge engine** merges the template and the data to create Email and Web output, or to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. A **Merge engine** merges the template and the data to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output.

Configuring these engines to match both the hardware configuration and the typical usage situation is probably the most effective way to improve Connect's performance.

The number of engines is one of the "Parallel Processing Preferences" that let the Connect Server manage its workload in such a way that the highest possible output speed is achieved.

For an explanation and guidelines to these settings, see "Engine configuration" on page 100 and "Parallel Processing preferences" on page 116.

> **Note:** Connect Server and Connect Designer each have their own distinct scheduling preferences.
>
> Use the **Connect Server Configuration** tool to change the Connect Server settings and **Designer > Windows >Preferences** for changing Designer settings.

Template optimization

When you find that the speed per Merge engine - the Content Creation speed - is low, optimizing a template can make a huge difference. For advice on how to optimize a template see: "Optimizing a template" on page 1685.

Data optimization

One record that produces thousands of pages will take long because only one merge engine can work on it. To produce output faster, try splitting the data into multiple records.

Network and internet connections

Use a fast network and internet connection or avoid loading external or internet resources. Using images, JavaScript or CSS resources located on a slow network or on a slow internet connection will obviously lead to a loss of speed. While we do our best for caching, a document with 5,000 records which queries a page that takes 1 second to return a different image each time will, naturally, slow output generation down by up to 83 minutes.

Hardware configuration

When processing speed is important, the following is suggested after addressing the other issues mentioned in this topic.

- **Antivirus exclusions**. Sometimes, virus scanners, other security software or indexing services can interfere. It can help to disable those kinds of tools for the areas where Connect stores intermediate files. You could exclude the entire `C:\Users\<connectuser>\Connect` folder. See also: "Antivirus Exclusions" on page 24.
- Use a **high-performance, low-latency hard drive**. Connect benefits from fast I/O. This is especially true for DataMapper engines (see "DataMapper engine" on page 104). Preferably use a Solid State Drive (SSD) or similar for storage.
- Use at least **8+ GB High-Quality RAM**. Check memory usage while the Print command is being executed to see if you need more than the minimum of 8GB. Assuming that the Connect Server and the Connect database need 1GB each, and that each engine needs 1GB as well, you can roughly estimate how much memory is needed.
- **Consider using a physical machine** instead of a virtual machine. When running on a Virtual Machine, the machine may report that it has sufficient hardware (cores) available, but in a virtual environment you need to make sure that this hardware is not being shared with lots of other virtual machines.
- Consider using hardware with more **physical cores**. OL Connect doesn't limit the number of Merge engines that is used for a Print job, so if the number of physical cores is low, it makes sense to see if that can be increased. When running on a virtual machine, this is usually easy. When running on a physical machine, it means that you may have to switch hardware.
- For both virtual and non-virtual environments, make sure the machine is not busy with all kinds of other processes.

System requirements

These are the system requirements for OL Connect 2025.1

Operating system

Any 64 bit Windows Desktop or Server version which is in its **mainstream** support period, as defined by https://learn.microsoft.com/en-us/lifecycle/.

New versions of Windows are supported within the first 6 months of their commercial release.

Note: ARM CPUs are not supported. This applies to machines running Windows on ARM and virtualization software running on Apple computers, such as Parallels Desktop.

Supported virtual environments

- VMWare vCenter/vSphere
- Microsoft Hyper-V/Azure
- Amazon Web Services (AWS)
- Google Cloud Platform

Note: Multi-user environments like Terminal Services or VMWare Horizon are not officially supported.

Recommended system requirements for the OL Connect / Workflow servers

As with any software application, minimum hardware requirements represent the most basic hardware on which the software will run. Note however that settling for the minimum specification is unlikely to produce the performance you expect from the system. It can be used when configuring a trial or a development system, however.

- **CPU**: at least 8 64-bit logical processors (aka threads)
- **RAM**: at least 16 GB
- **Storage**: at least 100GB available on fast SSD for the system drive (by default this means the C: drive)
- **Networking**: at least 1Gbps

Note: These values are for the OL Connect **Servers** when used in production. The OL Connect Designer and the Workflow Configuration Tool can be used on systems with lower hardware specifications, so long as they are not used to generate production output.

A word of caution

OL Connect is used for a very wide variety of applications. Consequently, it is difficult to determine which system configuration will produce the best results for any given application. The specifications above are therefore meant as general guidelines that *should* produce the expected results for *most* implementations.

Keep in mind, however, that they may not represent the optimal configuration for your particular application.

Rules of thumb

When installing on multiple PCs, keep the following rules of thumb in mind:

- The more logical processors, the better
- Higher CPU clock speeds are recommended: 24 logical processors @3GHz may perform better than 32 logical processors @2.2 GHz, depending on the workload
- For each additional logical processor, plan to add 2 GBs of RAM
- The number of logical processors is in direct relation with the number of OL Connect engines that can efficiently run concurrently

Installation and activation

This topic provides detailed information about the installation and activation of OL Connect 2025.1.

> **Note:** A PDF version of this guide is available for use in offline installations. Click here to download it.

OL Connect 2025.1 is comprised of 2 different installers: one for the OL Connect software and one for OL Connect Workflow 2025.1.

Where to obtain the installers

The installers for OL Connect 2025.1 and Workflow 2025.1 can be obtained on DVD or downloaded as follows:

- If you are a **Customer**, the installers can be downloaded from the Upland Objectif Lune Web Activation Manager (https://www.objectiflune.com/webactivationmanager/) or through the OL Update Manager if it is activated.
- If you are a **Reseller**, the installers can be downloaded from the Upland Objectif Lune Partner Portal site (https://extranet.objectiflune.com/) or through the Upland OL Update Manager if it is activated.

Installation - important information

For important information about the Installation, including requirements and best practices, please see the following topics:

- "Installation prerequisites" below
- "User accounts and security" on the next page
- "Migrating to a new workstation" on page 67
- "Upgrading from previous OL Connect versions" on page 75

Installation - "How to" guides

For information on how to conduct the installation itself, choose from the following topics:

- "Installation Wizard" on page 42
- "Running OL Connect installer in Silent Mode" on page 55
- "Installing OL Connect on machines without internet access" on page 40

Activation

For information on licensing, please see "Activating a License" on page 64.

Installation prerequisites

- Make sure your system meets the "System requirements" on page 35.
- OL Connect Version 2025.1 can be installed under a regular user account with Administrator privileges., see "User accounts and security" on the next page.
- OL Connect **must** be installed on an NTFS file system.
- OL Connect requires **Microsoft .NET Framework 4.5** (or later) already be installed on the target system.
- Connect 2019.1 requires updated Connect License and/or Update Manager.
  See "Upgrading from previous OL Connect versions" on page 75 for details.
- In order to use the **automation features** in Version 2025.1, **OL Connect Workflow 2025.1** will need to be installed.
  This can be installed on the same machine as an existing PlanetPress® Suite 7.x installation or on a new computer.
  For more information, please see "Information about OL Connect Workflow" on page 74.

  If Workflow installation finds that .NET 4.6.2 (or later) is not already installed, it will

prompt you to install this manually.

User accounts and security

Windows user account

Connect requires local Windows Administrator rights when installing the software and activating the software license. This is to allow read/write access to protected Windows folders and registry entries.

> **Note:** Installing as the *Local System* account is not supported.

Once installed Connect requires only standard Windows user credentials to run.

The following links contain the details as to when and where Windows Administrator rights are required:

- Connect Installation: "Installation Wizard" on page 42
- Activating Connect: "Activating OL Connect" on page 66

Permissions for OL Connect Designer

OL Connect Designer does not require any special permissions to run besides that of a regular program.

It does not require administrative rights and only needs permission to read/write in any folder where templates or data mapping configurations are located .

If generating Print output, OL Connect Designer requires permission on the printer or printer queue to send files.

Permissions for OL Connect Server

The OL Connect Server module, used by the *Automation* module, requires some special permissions to run. These permissions are set during installation, in the *Engine Configuration* portion of the "Installation Wizard" on page 42, but it can also be configured later by modifying permissions for the service. To do this:

- In Windows, open the Control Panel, Administrative Tools, then Services (this may depend on your operating system).
- Locate the service called Serverengine_UUID , where UUID is a series of characters that depend on the machine where the software is installed.
- Right-click on the service and select *Properties*.

- In the *Connection* tab, define the account name and password that the service should use. This can be a local account on the computer or an account on a Windows Domain. The account must have administrative access on the machine. It should also correspond to the user account set up in *Worfklow*.

OL Connect Server user accounts

By default, authentication is enabled on the Connect Server. During a new installation the OL Connect Server's default authentication must be configured by specifying a user name and password. The default username for new installations is **olc-user**.
More user accounts can be configured in the Server Configuration Tool (see "Security and Users Settings" on page 99).
In any Connect Designer that uses a secured Connect Server, an authenticated user account must be specified via the Designer's Preferences (see the "Connect Servers preferences" on page 1061 sub-section of the Designer Preferences dialog).

Note that prior to OL Connect version 2020.2, only one user account could be configured on a Connect Server. The default username was **ol-admin** and the default password was *secret*.

Installing OL Connect on machines without internet access

Installing OL Connect2025.1 in offline mode requires some extra steps. These are listed below.

Updating Connect

Updating to OL Connect 2025.1 from earlier Connect version

In order to update OL Connect to 2025.1 it is first necessary to update the OL Connect license and Update Manager .

For further details on how to upgrade the OL Update Manager to the latest version and update your OL Connect License see the Update Manager 1.9 - Upgrade Guide.

For full details on how update OL Connect see "Upgrading from previous OL Connect versions" on page 75.

Initial OL Connect installation

GoDaddy Root Certificate Authority needs to be installed

In order to install OL Connect, it is necessary for the GoDaddy Root Certificate Authority to be installed (G2 Certificate) on the host machine and for this to be verified online. When a machine hosting the installation does not have access to the Internet, the installation will

fail because the verification cannot be performed. To solve this problem one must first ensure that all Windows updates have been installed on the host machine. Once the Windows updates are confirmed as being up to date, then complete the following steps:

1. Go to https://certs.godaddy.com/repository and download the following two certificates to copy to the offline machine:
    - GoDaddy Class 2 Certification Authority Root Certificate - G2 - the file is gdroot-g2.crt
    - GoDaddy Secure Server Certificate (Intermediate Certificate) - G2 - the file is gdig2.crt
2. Install the certificates: Right mouse click -> Install Certificate, and follow the steps through the subsequent wizard.
3. Now copy the OL Connect installer to the offline machine and start the installation as normal.

Windows certificate validation - Certificate Revocation List retrieval should be switched off

For your security Upland Objectif Lune digitally signs all relevant files with our own name and certificate. The integrity of these files is checked at various times by different, context related, methods. One of these checks, done during the installation process, uses the Windows certificate validation check.

The Windows certificate validation process not only checks the integrity of a file against its signature, but also usually checks if the certificate itself is still valid. That check is done against the current Certificate Revocation List (CRL), which needs to be retrieved from the internet. However, if the machine in question does not have internet access, the retrieval of the CRL must fail, which will lead to subsequent validation issues.

To circumvent such issues it is **highly recommended** to switch off the CRL retrieval prior to installing Connect on machines without internet access. There is no security risk associated with this, as the CRLs would never be retrievable without internet access anyway. Advantage of the switch will not only be found during the installation and operation of Connect, but also in some speed improvements for any application which use signed binaries.

To switch off CRL retrieval on the computer, complete the following steps:

1. Open the “Internet Options” via the Control Panel.
2. Select the “Advanced” tab and scroll down to “Security” node.

3. Uncheck the entry “Check for publisher’s certificate revocation” under that node.
4. Click the OK button to close the dialog.
5. Re-start the computer.

Installation Wizard

Updating from Connect versions predating 2019.1

In order to update OL Connect to 2025.1 from Connect versions prior to 2019.1 it is first necessary to update the Connect License.
For details on how to upgrade the Connect License see "Users of OL Connect prior to 2019.1" on page 75

Starting the OL Connect installer

The OL Connect installer is supplied as an executable file.

Double click on the executable file and after a short pause the Setup Wizard will appear to guide through the installation steps.

Note: OL Connect **requires** prior installation of Microsoft .NET Framework 4.5.
For a full list of other prerequisites, see "Installation prerequisites" on page 38.

Note: It is recommended to install and use OL Connect Designer and Server under the same user account. If you install and run OL Connect Designer and Server under different user accounts (for example, install as *User1* and run Designer as *User2*), account permission issues may occur.

Running the Installation with extra logging

The installer can be run with enhanced logging options, if needed.
To do so, run the OL_Connect_<<Edition>>_2025.1.n.nnnnn_bnnnn.exe (replace <<Edition>> with either Enterprise, Professional or Desktop; and update the build number appropriately) from the command line with one of the following command line options :

- `OL_Connect_<<Edition>>_2025.1.n.nnnnn_bnnnn.exe --verbose`

  This adds extra debugging style logging to the installation process.

- `OL_Connect_<<Edition>>_2025.1.n.nnnnn_bnnnn.exe --trace`

  This adds full trace style logging to the installation process. The log file this produces will be very large, as this option logs everything.

Prerequisites Installation

The installer will check for prerequisite technologies as the first step in the installation process. If this check finds some technologies are missing, it will install those technologies, before continuing with the installation.

Welcome screen

After any prerequisites are installed, the installer Welcome screen appears.
Click **Next** to continue with the installation.

If the installation is an upgrade over a pre-existing OL Connect installation, the installer will first uninstall the earlier version.

If you would like to retain the usage information from that pre-existing OL Connect installation, do not select the **Remove User data** checkbox option.

For information about exactly what data would be saved or deleted, please see "Pre-existing User Data" on page 78.

Switching to a different edition

When switching to a different edition, for example from the Professional to the Enterprise edition, the installer may indicate that the edition to be installed is not the same as the one for which the license is valid. In that case, the license file must first be removed from the following folder:

C:\ProgramData\Objectif Lune\OL Connect\licenses

License Agreement

The next page displays the "End User License Agreement (EULA)" on page 2015, which needs to be read and accepted before clicking **Next**.

Component Selection

After clicking the Next button, the Component Selection page appears, in which the different components of OL Connect can be selected for installation.

The options are:

- **Base**: The installation files required for any OL Connect installation. This component is not optional.

- **Designer**: The Designer module (see "The Designer" on page 587) can be installed standalone (with no other installed modules) on as many machines as you like. It does not require a license to run as a standalone designer tool . This allows any number of people to use the Designer for creating jobs, but without production capabilities such as automation and commingling. The Designer module is optional, but it is recommended that it always be installed.
    - **Messenger**: The Messenger Service that allows connection between OL Connect's Designer and Workflow. Recommended for all production installations.
- **Server**: The Connect Server back-end that provides Connect production capabilities such as production content creation (print output, HTML content for emails and web pages), automation, commingling and picking. It is also referred to as the Connect Master Server in Connect clustered environments.

  Note: (Enterprise edition only) When either the Designer or the Server component is selected for installation, the **AFPmerge** tool is installed as well. This command line tool can embed resources in AFP data files and merge multiple AFP data files into one AFP file. For information about its use, see: .

- **OL Connect Server Extension** (Enterprise edition only): A Client server that is subservient to the main OL Connect Master Server module. When a Client server is installed it communicates with the Master server in order to shares tasks with it. Such an installation assumes that the **Connect Server** (the Connect Master Server) has previously been setup on another machine, and that the database option "***Allow remote client access***" option were selected during that installation.

  Before installing the software, make sure that both TCP/IP ports 3306 and 9340 are open on the Connect Master Server and Connect Extension Server, in both the Inbound and Outbound Firewall Rules. Access within the Private and Domain profiles is sufficient.

  It is recommended your firewall blocks access to port 3306 from external requests as it may represent a security risk if the machine is open to the internet!

  For further information refer to the **Binding and Root access on the Master server instructions** in "Server Clustering" on page 124.

  Note: Only one of the **Server** or **Server Extension** can be installed on a single machine, not both.

- **Print Manager** (Enterprise edition only): The Print Manager module (see "Print Manager" on page 1689) is ***required*** for handling IPDS print output, but may also be optionally installed for managing PCL and PostScript print output.
  The **Print Manager** can be installed standalone (so long as the **MariaDB component** is included in the installation) and can be installed on more than one machine.
- **MariaDB Server**: A supplied MariaDB database used by OL Connect.

  The database is used for referencing temporary Connect files and for sorting temporarily extracted data, and similar.

  > **Note:** When performing an upgrade installation, if the MariaDB version has not significantly changed, then no attempt will be made to upgrade the database content.
  > If there is a significant MariaDB version change, the database content will be upgraded, so that it will continue to work with the new MariaDB version.

  A pre-existing MariaDB, MySQL or Microsoft SQL server (referred to as an ***external*** database, in this documentation) *could* be used instead, for the same purposes. The *external* database could reside on the same computer or on a separate server.
  If you wish to make use of an *external* database, please make sure the **MariaDB** option is not selected.

  > **Caution:** If you chose **not** to install the supplied MariaDB database, and instead opt for using a pre-existing (*External*) database then you yourself must ensure that the *External* database is accessible to Connect.
  >
  > Upland Software, Inc. will take no responsibility for setting up database connections to any but the supplied MariaDB database.
  >
  > See "Database Considerations" on page 26 for more information about setting up *external* databases.

  > **Note:** (Enterprise edition only): All instances of **OL Connect Server** (Master) and **OL Connect Server Extension** (Clients) must use the same instance of MariaDB. Whilst it is *possible* for each separate client installation to have its own MariaDB instance, doing so will prevent the Servers from functioning together within a "Server Clustering" on page 124 environment.
  >
  > The single MariaDB instance can be on any machine, however, whether it be the

Server (master) or one of the Server Extension (client) machines.

- **Destination folder**: This is the location where Connect components are to be installed. Use the Browse button to navigate to a folder other than the default, if required.

  Note: The installation path cannot contain any non ASCII characters (such as Asian language Unicode characters). Nor can it contain characters that Windows disallows in filenames (such as '?', '>' or trailing spaces).
  If an invalid character is entered, the Installation Path entry box will turn red and a description of the error will be displayed in the information area.

The installer calculates how much disk space is required for installing the selected components, along with how much space is available.

- **Total Required Space** : Displays the amount of disk space required for the selected components.
- **Space Remaining**: Displays the amount of space available after installation on the drive currently in the Installation Path.

OL Connect Service Configuration

The **Service Configuration** page is for setting the Microsoft Windows Account that the *Connect Service* component will use.

- **Log on as**: Defines the Windows user account and password that the **Connect Server** services will use.

  Note: The Windows user account must have access rights to all local and network resources required for production, as well as Windows "*Log on as a Service*" rights.

  The Windows user account selection entered here should be recorded for future use, as the "Security and Users Settings" on page 99 dialog can only ever be executed through the user account specified on this page.

  - **Account**: The Windows user account that the service uses to login. If the machine is within a domain, use the format `domain\username`.
    This account must be an existing Windows profile with local Administrator rights.

    Note: Installing as the *Local System* account is not supported.

  - **Password**: The password associated with the selected user.

Use the eye icon to toggle between displaying or masking the password entry.
The password is not validated for password strength, so any entry is acceptable.

- **Validate Account** button: Click to verify that the entered account and password combination is correct and that the service is able to login.

  Note: This button *must* be clicked and the user validated before the **Next** button becomes available.

- **Start service when installation is complete** checkbox: Select this option to have the service start upon installation completion (which is the default).
  If unchecked, the Service will start upon machine reboot.

OL Connect Server Connection

Set the Connect **Server Connection** internal username and password.

The **Server Connection** settings will be slightly different depending upon whether **Connect Server** or **Connect Server Extension** was selected for installation (Enterprise edition only).

The options available are as follows:

- **Host**: This option is only available if **Connect Server Extension** was selected for installation.
  Enter the machine name or IP Address where the Connect Master Server resides.There is no requirement for the **Master** and **Extension** servers to belong to the same IP subnet.

  IP subnetting is beyond the scope of this documentation, but more information can be found here: https://en.wikipedia.org/wiki/Subnetwork.

- **Port**: Enter the port to use to communicate with the Connect Server.
  By default the Connect Server controlled by the *OLConnect_Server* service communicates through port 9340.

- **User**: Enter the internal username for connection to the OL Connect Server.
  The default username for new installations is **olc-user**.

- **Password**: The password associated with the selected user.

  Use the eye icon to toggle between displaying or masking the password entry.
  The password is not validated for password strength, so any entry is acceptable.

Note that prior to OL Connect version 2020.2, only one user account could be configured on a Connect Server. The default username was **ol-admin** and the default password was *secret*.

Database Configuration

The Default **Database Configuration** page appears if the supplied *MariaDB* module was selected for installation in the *Product Selection* screen. It defines the administrative root password for the MariaDB server as well as which port it uses for communication.

The installer will automatically configure the Connect *Server* to use the supplied password and port.

- **Port**: The port on which MariaDB will expect requests to come though, and through which it itself responds.
  A check is run to confirm whether the specified TCP\IP Port Number is available on the local machine. If it is already being used by another service (generally, an existing MySQL or MariaDB installation), the number is highlighted in red and a warning message is displayed.

  > **Note:** The MariaDB database controlled by the *OLConnect_MariaDB* service communicates through port 3306 by default.

- **Root password**: Enter the password for the 'root', or administration account, for the MariaDB server.
  Use the eye icon to toggle between displaying or masking the password entry.
  We recommend that the password be at least 8 characters long and contain at least one of each of the following, even though password selection strength is not enforced by the installer:
  - a lower case character (a, b, c …)
  - an upper case character (A, B, C …)
  - a numeric digit (1, 2, 3 …)
  - a punctuation character (@, $, ~ …)

  For example: "This1s@K"

  > **Note:** When updating from an earlier Connect version, the appropriate MariaDB password **must** be entered or the update will fail.

If the password is subsequently forgotten, then MariaDB must be uninstalled and its database deleted from disk before attempting to reinstall.

- **Allow remote client access** checkbox: Click to enable external access to the MariaDB server.

  Note: This option is required if setting up clustering, or if the MariaDB Server will need to be accessed from any other machine

  It will also be required if the MariaDB database is on a separate machine to this OL Connect installation.

  Tip: This option may represent a security risk if the machine is open to the internet.

  We heavily recommended that your firewall is set to block access to port 3306 from external requests.

- **Username**: Enter the MariaDB user name that will be associated with OL Connect. The default username for new installations is **olconnect**.
- **Password**: The password associated with the selected user.

  Use the eye icon to toggle between displaying or masking the password entry. The password is not validated for password strength, so any entry is acceptable.

Configuring External Database Connection

The **Database Connection** page appears if the supplied MariaDB module was not selected for installation. This page is for setting up the connection to an existing External database.

- **System**: Select the database type to use for the OL Connect Engine. Currently only MariaDB, MySQL and Microsoft SQL Server are supported.
- **Host**: Enter the IP Address or alias of the server where database resides.
- **Database Instance Name**: Enter an existing Microsoft SQL Server's instance name. This option only applies to existing Microsoft SQL Server instances, and not for MariaDB or MySQL.
  Note that the MS SQL Server instance name must meet Microsoft's requirements.
- **Port**: Enter the port on which the database server expects connections.
  For MariaDB and MySQL, this is **3306** by default.
  For Microsoft SQL Server it is **1433** by default.

- **Schema**: Enter the name of the database into which the tables will be created. The standard Connect schema name is "olconnect" by default.
- **Username**: Enter the user account of a user with database administrative rights. Administrative rights are required since tables will need to be created/modified/dropped in the database.
  If accessing a database on a different machine, the server must also be able to accept non-local TCP connections and the user account must also be configured to accept remote connection.
  For example, the "root" MySQL user entered as root@localhost is not allowed to connect from any other machine than the one where MySQL is installed.
- **Password**: Enter the password for the above user account. For MySQL the appropriate password **must** be entered or the Connect installation will fail.
- **Use Windows authentication**: The account of the currently logged in user who launched the installation will be used to perform the validation and configuration of the database.
  This option is only available when Microsoft SQL Server is selected.

  Caution: If this option is enabled, OL Connect cannot be used with any Windows user other than the user who performed the installation.
- **Encrypt Connection** checkbox: Check to enable encrypted connections to the external database.

  The secure connection to MySQL is for the "olconnect" schema.

  Note: It is not in the scope of the Connect installer to configure the MySQL database Server to accept SSL connections. This must be done prior to the installation of Connect.

  By default, the connection will not verify the server certificate (verifyServerCertificate=false), which would allow connecting to a server using a self-signed certificate. If such a certificate is required, then this setting can be changed after installation within the "Database Connection preferences" on page 1041 (which can be accessed from either the "Server configuration settings" on page 94 tool, or the "Preferences" on page 1035 window).

- **Test Connection** button: Click to verify that the information provide into previous fields is valid by connecting to the database.

  > **Note:** This test does not check whether the remote user has READ and WRITE permissions to the tables under the objectiflune schema. It is solely a test of database connectivity.

Ready to install

This page confirms and lists the installation selections made.

If components have been selected which have a shortcut associated with them (Designer, Print Manager, Server) then you will presented with the option to **Create desktop shortcuts**. Select if you wish for desktop icons to be created.

Click **Install** to start the installation itself. This process can take several minutes.

Installation Finished

This screen describes a summary of the components that have been installed.

- **Configure update checks** checkbox: This option is enabled by default. It causes the **Product Update Manager** to run after the installation is complete. This allows configuring OL Connect to regularly check for entitled updates.
  Note: this checkbox may not be available in the event that an issue was encountered during the installation.

When ready, click the **Finish** button to close the installation wizard, and initialize the Product Update Manager, if it was selected.

The Product Update Manager

If the **Configure Update Check** option has been selected, a message will be displayed after clicking "*Finish*" in the setup. The message details the information that needs to be sent back to Upland Software, Inc. in order to determine when/if the software needs updating.

Click "Yes" to install or open the Product Update Manager where the frequency with which the updates can be checked and a proxy server (if required) can be specified.

> **Note:** If the Product Update Manager was already installed by another Upland Software, Inc. application, it will be updated to the latest version and will retain the settings previously specified.

Select the desired options and then click **OK** to query the server and obtain a list of any updates that are available for your software.

- Note that the Product Update Manager can also be called from the "***Objectif Lune Update Manager***" option in the Start menu.
- It can be uninstalled via Control Panel | Programs | Programs and Features.

Product Activation

After installation, it is necessary to activate the software. See "Activating a License" on page 64 for more information.

Before activating the software, please wait 5 minutes for the database to initialize. If the software is activated and the services rebooted too quickly, the database can become corrupted and require a re-installation.

Pre-existing User Data

The following scenarios display what happens to pre-existing User Data in a Connect upgrade.

> **Note:** In regards to the MySQL (Connect version 2021.2 and earlier) or MariaDB (Connect version 2022.1 onwards) entries: these are only applicable if the OL Connect database component was installed in a previous Connect installation.

Scenario 1: Upgrading from OL Connect 2021.2 or earlier with Remove User Data CHECKED

- In all cases:
  - Files and folders are removed from the user data folder *C:\Users\<connectUser>\Connect* , where *<connectUser>* is the user that installed OL Connect.
  - Files and folders are removed from the following data folders:
    - *C:\ProgramData\Objectif Lune\OL Connect\.settings*
    - *C:\ProgramData\Objectif Lune\OL Connect\CloudLicense*
    - *C:\ProgramData\Objectif Lune\OL Connect\ErrorLogs*
    - *C:\ProgramData\Objectif Lune\OL Connect\LiquibaseUpdate*
  - Files are removed from the root of the data folder *C:\ProgramData\Objectif Lune\OL Connect*.
    If the folder is empty following this (i.e. no license or user folders were present)

then the *C:\ProgramData\Objectif Lune\OL Connect* folder itself is removed.

- License files as well as any content not listed above but found in the *C:\ProgramData\Objectif Lune\OL Connect* folder remain untouched.

- Additional cases:

1. If MySQL was previously installed as an OL Connect component AND the database contains some user-defined schemas:
    - The native OL Connect schema is removed from the MySQL database.
    - The MySQL database files (*C:\ProgramData\Objectif Lune\OL Connect\MySQL*) are kept intact, as user-defined schemas mean that the user did not have only the OL Connect native schema content in their database.
        - A message at the end of the upgrade will advise the user that some non-OL schemas were found in the database, so the database files were not removed.
2. If MySQL was previously installed as an OL Connect component AND the database does not contain any user-defined schemas:
    - The MySQL database files (*C:\ProgramData\Objectif Lune\OL Connect\MySQL*) are removed entirely.

Scenario 2: Upgrading from OL Connect 2021.2 or earlier with Remove User Data UNCHECKED

- In all cases:
    - The user data folder *C:\Users\<connectUser>\Connect* (where *<connectUser>* is the user that installed OL Connect) is retained, untouched.
    - All the files and folders under the data folder *C:\ProgramData\Objectif Lune\OL Connect* remain untouched.

- If MySQL was previously installed as an OL Connect component:
    - All schemas from the MySQL database are migrated to MariaDB, allowing the user to continue using their database content normally.
      NOTE: This might take some time during the installation, depending upon the size of the existing databases.
    - The MySQL database files (*C:\ProgramData\Objectif Lune\OL Connect\MySQL*) are also kept intact.

Scenario 3: Uninstalling OL Connect 2022.1 or later with Remove User Data CHECKED

- In all cases:
- Files and folders are removed from the user data folder *C:\Users\<connectUser>\Connect* , where *<connectUser>* is the user that installed OL Connect.
- Additional cases:
    1. If MariaDB was previously installed as an OL Connect component AND the database contains some user-defined schemas:
        - The native OL Connect schema is removed from the MariaDB database.
        - The MariaDB database files (*C:\ProgramData\Objectif Lune\OL Connect\MariaDB*) are kept intact, as user-defined schemas mean that the user did not have only the OL Connect native schema content in their database.
            - A message at the end of the upgrade will be displayed, stating some non-OL schemas were found in the database, so the database files were not removed.
    2. If MariaDB was previously installed as an OL Connect component AND the database does not contain some user-defined schemas:
        - The MariaDB database files (*C:\ProgramData\Objectif Lune\OL Connect\MariaDB*) are removed entirely.

Scenario 4: Uninstalling OL Connect 2022.1 or later with Remove User Data UNCHECKED

- In all cases:
    - Files and folders are removed from the user data folder *C:\Users\<connectUser>\Connect* , where *<connectUser>* is the user that installed OL Connect.
    - Files and folders are removed from the following data folders:
        - *C:\ProgramData\Objectif Lune\OL Connect\.settings*
        - *C:\ProgramData\Objectif Lune\OL Connect\Address Blocks* (introduced OL Connect 2024.2)
        - *C:\ProgramData\Objectif Lune\OL Connect\CloudLicense*

        - *C:\ProgramData\Objectif Lune\OL Connect\ErrorLogs*
        - *C:\ProgramData\Objectif Lune\OL Connect\LiquibaseUpdate*
    - Files are removed from the root of the data folder *C:\ProgramData\Objectif Lune\OL Connect*.
      If the folder is empty following this (i.e. no license or user folders were present) then the *C:\ProgramData\Objectif Lune\OL Connect* folder itself is removed.
    - License files as well as any content not listed above but found in the *C:\ProgramData\Objectif Lune\OL Connect* folder remain untouched.
- If MariaDB was previously installed as an OL Connect component:
    - All schemas from the MariaDB database are kept, allowing the user to use those database files if they reinstall the software.

Running OL Connect installer in Silent Mode

Updating from Connect versions predating 2019.1

In order to update OL Connect to 2025.1 from Connect versions prior to 2019.1 it is first necessary to update the Connect License.
For details on how to upgrade the Connect License see "Users of OL Connect prior to 2019.1" on page 75

General information

OL Connect can be installed from the command line in "*silent mode*" to allow for scenarios such automated installations during company wide roll-outs, or to allow for unattended out-of-hours updates. The trigger for the Connect Installer to run in silent mode is a text file with the fixed name **installProperties.ini** located in the same folder as the OL Connect installation executable file.

How to prepare the **installProperties.ini** file is detailed in the following sections.

Installation Properties file

The basic rules for the **installProperties.ini** file are:

- Comment Lines start with the semi-colon character ';'
  Example: `; The options to configure an external database`
- The properties are listed in **Key = Value** pair format.
  Example: `product.ServerExtension = false`
- Quotes are used for path strings, but not for other string types

Example: `path = "c:\Program Files\Objectif Lune\OL Connect"`

- Property values are case insensitive

The installation settings fall into three distinct categories:

- "[Logging]" below for setting the installation logging options.
- "[Installation]" on the next page for selecting what gets installed.
- "[Uninstall]" on page 60 handles properties related to product Uninstallation. These also impact Maintenance mode.

[Logging]

This section handles **Silent Installer** logging options.

The logging Key pairs are as follows:

- **verbose**: Boolean (Default: `false`)
  A basic log is always created by the installer, with or without the verbose option. The verbose option is more suitable for debugging purpose.
  If set to true, then a verbose log file is created in the logging path specified in the INI file.
  If no logging path is specified in the INI file, then the default one is used.
  If set to false, standard logging is done.
- **path**: String (Default: `%PROGRAMDATA%\Objectif Lune\Installation Logs`)
  Sets the folder to which the installation log will be written.
  Only the log folder should be specified here, not the log file name.

  Note: The log file name's format is set automatically and uses the format *Installer-YYYY-MM-DD-####.#.#.log*, where:

  - *YYYY-MM-DD* = The date the log was created
  - *####.#.#* = The OL Connect version number

  The file name for a maintenance installation begins with "Maintenance" rather than "Installer"

Example:

```
; Logging properties
[Logging]
verbose = false
path = "c:\temp\Silent Install"
```

[Installation]

This section handles various installation parameters as well as product selection.

The logging Key pairs are as follows:

- **product.<name>**: Boolean (Default: `false`)
  Each OL Connect product has its own entry, which can be set to true (to install) or false (to omit from installation).
  What products are available for installation is determined by which OL Connect branding is being installed. The products for OL Connect branding are as follows:
  - **product.Designer**
  - **product.PrintManager** (Enterprise edition only)
  - **product.Server**
  - **product.ServerExtension** (Enterprise edition only)
  - **product.MariaDB**
  - **product.Messenger**
- **path**: String (Default: `%PROGRAMFILES%\Objectif Lune\OL Connect`)
  Sets the installation root folder for the OL Connect applications.
- **RegisterService.connectServer**: Boolean (Default: `true`)
  Register the Server (or Server Extension in the Enterprise edition) services or not (such as in the case of a container).
- **server.username:** String (Default: the current user/domain installing the service)
  Determines the domain and username to be used when configuring the Server (or Server Extension in the Enterprise edition) service.

  The username can use the following syntax formats:
  - username
  - domain\username
    (Note: the backslash between the domain and user names needs to be escaped by another backslash. For example: `server.username = ourcom-pany\\pbrown01`)
  - username@domain
- **server.password**: String (Default: there is no default for this setting)
  Password to use when registering the Server (or Server Extension in the Enterprise edition) service.

- **database.configure**: Boolean (Default: `true`)
  If set to "false", then the database configuration is skipped.

  > **Note:** If database configuration is skipped (`database.configure = false`), then none of the `database.xxx` properties below are required, and these properties will be ignored, even in they are included in the INI file.

- **database.system**: String, Optional (Default: there is no default for this setting)
  Entry needs to be from one of the following options: **mariadb**, **mysql**, **mssqlserver**

  > **Note:** If `product.MariaDB = true` has been set:
  >
  > - This setting becomes optional. Otherwise it is required.
  > - The value is *required* to be `mariadb`, if the value is to be provided.

- **database.host**: String, Optional (Default: there is no default for this setting)
  If `product.MariaDB = True` has been set, this value is required to be set to `localhost`, if provided.
- **database.port**: Numeric, Optional (Default: MariaDB's default port, 3306)
  The database engine port.
  The required entry depends upon the selection made in `database.system`.
  If `product.MariaDB = True`, then the port should be set to 3306.
- **database.rootpassword**: String (Default: there is no default for this setting)
  Database root password.
  There is no default value, and if this is left unspecified the installation will fail.
- **database.remoteaccess** (Enterprise edition only): Boolean (Default: depends upon whether the Connect Server is installed.
  If the property `product.Server = true`, then the default is false.
  If `product.Server = false` or is left unspecified, then the default is `true`)
  When set to `true` the database will be configured to accept connections from systems other than localhost.
- **database.username**: String (Default: `olconnect`)
  The username that OL Connect will use to connect to the database.
- **database.password**: String (Default: there is no default for this setting)
  The password that OL Connect will use to connect to the database.
  There is no default value, so if this is left unspecified, then the installation will fail.
- **database.instance**: String, Optional (Default: there is no default for this setting)
  Only valid if `database.system = mssqlserver`.

- **database.schema**: String (Default: there is no default for this setting)
  Specifies the database schema to use. Required. (Optional if `product.MariaDB = true`)
- **database.encryptedconnection**: Boolean (Default: `False`)
  Specifies the database schema to use.
  (Optional if if `product.MariaDB = true`)
- **server.connection.configure**: Boolean, Optional (Default: `True`)
  If set to `False`, then the server connection configuration is skipped.
- **server.connection.user**: String (Default: `olc-user`)
  The server connection username.
- **server.connection.password**: String (Default: there is no default for this setting)
  The server password. Required if `server.connection.configure = true`.
- **server.connection.port**: Numeric (Default: 9340)
  The server port number.
- **server.connection.host** (Enterprise edition only): String (Default: there is no default for this setting)
  The server host. Required only for the Server Extension if `server.connection.configure = true`.
- **desktopShortcuts**: Boolean (Default: `False`)
  Specifies whether desktop shortcuts are to be added or not.
  This flag only takes effect if components were selected which have a shortcut associated with then (Designer, Print Manager (Enterprise edition only), Server). If no such were selected, then this flag will have no effect.
- **Language**: String, Optional
  Sets the OL Connect application language settings.
  Supported user locales (as language code) are as follows:
  - English: en
  - French: fr
  - German: de
  - Italian: it
  - Japanese: ja
  - Spanish: es
  - Portuguese (Brazil): pt

- Chinese (PRC/TW): zh
- Korean: ko

- **Locale**: String, Optional
  Sets the OL Connect application's region using a country code. For example: US, UK, DE, CH.
  If this property is not set, the locale defaults to the system locale, and if that is not found, the selected installation language will determine a default locale.

Example:

```
; Installation settings
[Installation]
product.Designer = true
product.Server = true
product.PrintManager = true
product.ServerExtension = false
product.MariaDB = true
product.Messenger = true
RegisterService.connectServer = true
server.username = Administrator
server.password = ObjLune
server.connection.user = olc-user
server.connection.password = secret
database.rootpassword = @Admin2022
database.username = olconnect
database.password = @Admin2022
database.remoteaccess = true
desktopShortcuts = true
path = "c:\Program Files\Objectif Lune\OL Connect"
Language = fr
Locale = FR
```

[Uninstall]

This section handles properties related to Uninstallation, but also Maintenance mode installations.
These options have no effect at all if OL Connect is not present on the system.

The logging Key pairs are as follows:

- **remove**: Boolean (Default: `False`)
  If the present version of the installer is already installed, this property defines the installer behaviour.
  The installer will perform a *Remove MSI* action on OL Connect if `remove = true`, whilst a *Modify MSI* action will be performed if `remove = false`.

  Note: This option has no effect if the product is not installed on the current system

> **Note:** If the *Remove* action is taken, then the equivalent of an uninstall is done, while a *Modify* action will change the components installed on the system based on the ones defined in the INI file [Installation] section, allowing removal or addition of components in the current installation.

- **keepdata**: Boolean (Default: `True`)
  Allows the user to specify if they wish to keep or remove user data (located under *%PROGRAMDATA%\Objectif Lune\OL Connect* when performing a product uninstall.

Example:

```
; Uninstallation/Repair properties
[Uninstall]
remove = true
keepdata = true
```

Properties file examples

Simple Connect installation example

Here is an example of a complete **installProperties.ini** file for a relatively simple OL Connect installation.

```
; OL Connect silent installer properties

; Logging properties
[Logging]
verbose = false
path = "c:\ProgramData\Objectif Lune\Installation Logs"
;

Installation settings
[Installation]
product.Designer = true
product.Server = true
product.PrintManager = true
product.ServerExtension = false
product.MariaDB = true
product.Messenger = true
RegisterService.connectServer = true
server.username = Administrator
server.password = ObjLune
server.connection.user = olc-user
server.connection.password = secret
database.rootpassword = @Admin2022
database.username = olconnect
database.password = @Admin2022
database.remoteaccess = true
desktopShortcuts = true
path = "c:\Program Files\Objectif Lune\OL Connect"

;Optional uninstallation settings
[Uninstall]
remove = true
keepdata = false
```

Server Extension installation example

The following is an example of a complete **installProperties.ini** file for a Server Extension installation (Enterprise edition only).

```
; OL Connect silent installer properties for Server Extension installation

; Logging properties
[Logging]verbose = false
path = "c:\ProgramData\Objectif Lune\Installation Logs"

; Installation settings
[Installation]
product.Designer = false
product.Server = false
product.PrintManager = false
product.ServerExtension = true
product.MariaDB = false
product.Messenger = true
RegisterService.connectServer = true
server.username = Server EX2
server.password = objlune
database.system = MariaDB
database.host = 192.168.108.50
database.username = root
database.password = Admin@123
database.schema = olconnect
database.encryptedconnection = false
database.rootpassword = Admin@123
database.remoteaccess = true
desktopShortcuts = true
server.connection.configure = true
server.connection.host = 192.168.108.50
server.connection.port = 9340
server.connection.user = ol-admin
server.connection.password = secret
path = "c:\Program Files\Objectif Lune\OL Connect"

; Optional uninstallation settings
[Uninstall]
remove = true
keepdata = false
```

Exit Codes

Success

- 0 = Installation completed successfully / no specific error code was returned.

Pre-Installation Check (200s)

- 201: Operating system is not 64-bit
- 202: Windows Operating System is too old (pre-Windows 7)
- 203: Minimal UAC prerequisites were not met
- 204: Installation user did not have administrator rights

Silent installation properties (300s)

- 301: MariaDB product was selected, but no root password was supplied
- 302: MariaDB product was selected, but no user password was supplied

- 303: User provided database is to be used, but `database.system` property was missing
- 304: User provided database is to be used, but `database.system` property had invalid value
- 305: User provided database is to be used, but `database.host` property was missing.
- 306: database.instance property was used with a database system other than MS SQL Server
- 307: User provided database is to be used, but `database.username` property was missing.
- 308: User provided database is to be used, but `database.password` property was missing.
- 309: Connect server / server extension product was selected, but `server.connection.password` property was missing
- 310: Connect server extension product was selected, but `server.connection.host` property was missing

Upgrade errors (400s)

- 401: Some Connect applications were running and need to be closed before installation can proceed.
- 402: The installer brand does not match the brand of the OL Connect version currently installed. (PlanetPress)
- 403: The installer brand does not match the brand of the OL Connect version currently installed. (PReS)
- 404: The installer brand does not match the brand of the OL Connect version currently installed. (Printshop Mail)

License file validation (500s)

- 501: OL Connect license file is in older format
- 502: License Care Date does not allow installation of product
- 503: License brand mismatch with installer brand

Destination and selected product check (600s)

- 601: Server and Server Extension both were selected to be installed.
  Only one of the two may be installed on any one system.

- 602: No component was selected to be installed
- 603: Destination folder is invalid or there is too little disk space available

Installation aborted (700s)

- 701 - 725: Installation was aborted due to user cancellation

Service stop error (800s)

- 801: OLConnect_MySQL/OLConnect_MariaDB service could not be stopped during uninstallation

Activating a License

OL Connect and OL Connect Workflow both come with individual 30 day trial license periods during which time it is not necessary to have a commercial license to run the applications.

This allows time for reviewing the applications and for organizing a commercial license. If a modification to the trial license is required, such as to allow an extension to the trial period, or for extra functionality, then a new activation code will need to be requested.

Obtaining the OL Connect Magic Number

To obtain an activation file the OL™ **Magic Number** must first be retrieved. The Magic Number is a machine-specific code that is generated based on the computer's hardware and software using a top-secret Upland Objectif Lune family recipe. Each physical computer or virtual computer has a different Magic Number, and each requires a separate license file to be functional.

To get the OL Connect **Magic Number** open the **OL Connect Software Activation** application.

- Open the **Start Menu**
- Click on **All Programs** and browse to the **Objectif Lune** folder.
- Open the **OL Connect Software Activation** shortcut.
- The **OL Connect Software Activation** application consists of the following:

- **License Information** subsection:
  - **Magic Number**: Displays the OL Connect Magic Number.
  - **Copy the magic number to the clipboard**: Click to copy the Magic Number to the clipboard. It can then be pasted in the activation request email using the Windows CTRL+V keyboard shortcut.
- **Licensed Products** subsection:
  - **Name**: Displays the name of the application or module relevant to this activation.

    The Information button provides detailed information about the application or module license.
  - **Serial Number**: Displays the trial license serial number or the activation serial number if the product has been activated in the past.
  - **Expiration Date**: Displays the date when the activation will expire, or the current date if the product is not activated.
- **End-User License Agreement** - *Appears only when loading a license file*:
  - **License**: This box displays the EULA. Please note that this agreement is legally binding.
  - **I agree**: Select to accept the EULA. This option **must** be selected to install the license.
  - **I don't agree**: Select if you do not accept the EULA. You cannot install the license if this option is selected.
- **Load License File**: Click to browse to the Connect license file (**.olconnectlicense**), once it has been received.
- **Install License** - *Active only when a license file is Loaded*: Click to install the license and activate the software.
- **Close**: Click to cancel this dialog.
  Even if a license file has been Loaded, it will not be installed if this dialog is Cancelled before the **Install License** button was clicked.

Requesting a license

After getting the Magic Number, a license request must be done for both OL Connect (with the Enterprise version: for both Master and Extension Servers, if using a Clustered environment) and Workflow:

- **Customers** must submit their Magic Number and serial number to Upland Objectif Lune via the Web Activations page: http://www.objectiflune.com/activations. The OL Customer Care team will then send the OL Connect license file via email.
- **Resellers** can create an evaluation license via the Upland Objectif Lune Partner Portal by following the instructions there: http://extranet.objectiflune.com/

Note that if you do not have a serial number, one will be issued to you by the OL Activations team.

Accepting the license will activate it, after which the OL Connect services will need to be restarted. Note that in some case the service may not restart on its own. To resolve this issue, restart the computer, or start the service manually from the computer's Control Panel.

Activating OL Connect Workflow

OL Connect Workflow uses the same licensing scheme as OL Connect. There are two ways of activating the license for Workflow after saving it to a suitable location:

- If only OL Connect Workflow is installed, double-click on the license for the OL Connect Workflow License Activation dialog to open. Applying the license here activates all of the Workflow components.
- If you have both OL Connect Workflow and OL Connect installed, it will not be possible to double-click on the license file as this will always open the OL Connect Activations Tool. Instead, open OL Connect Workflow manually and apply the license through the activations dialog within.

Activating OL Connect

To activate OL Connect, simply save the license file somewhere on your computer where you can easily find it, such as on your desktop. You can then load the license by double-clicking on it, or through the **OL Connect Software Activation** tool.

Activating the OL Connect license requires the user to have local Windows Administration rights.

- Using a user profile that has local Windows Administration rights, open the **Start Menu**
- Click on **All Programs**, then browse to the **Objectif Lune** folder.
- Run the "**OL Connect Software Activation**" tool.
- Click the **Load License File** button, and browse for the .olconnectlicense file you received from Upland Software, Inc..
- Read the EULA and click the *I agree* option to accept it.

- Click **Install License** to activate the license. The license will then be registered on the computer and you will be able to start using the software.

Caution: After installation message will appear warning that the Server services will need to be restarted. Just click OK to proceed.

Reactivating your license

You will need to request a new license if:

- The **magic number has changed** on the workstation the license is activated. The magic number can change following hardware updates or updates of the operating systems. When that happens, the software prompts you to enter a valid activation code.
- You are transferring the license to a **new workstation**.

As a customer, log in at the Web Activation Manager with your email address, customer ID and password to reactivate your license. For further instructions, read the Web Activation Guide (available for download from the Web Activation Manager).

If you are a reseller/partner:

1. Login at https://extranet.objectiflune.com/index.aspx.
2. Click “Reseller Licenses”.
3. Go to the “Request an NFR” section; select the product from the drop-down list, click “Select license”.
4. Enter the new Magic Number and click Submit. You will receive an email shortly containing your activations file.

Migrating to a new workstation

The purpose of this document is to provide a strategy for transferring a OL Connect (and/or Workflow) installation to a new workstation.

Before installing the software

Before upgrading to a new version, even on a new workstation, consult the product's release notes to find out about new features, bug fixes, system requirements, known issues and much more.

- "OL Connect Release Notes " on page 1722

You should also consult the following pages for some technical considerations before installing:

- "Network considerations" on page 31
- "Database Considerations" on page 26
- "Environment considerations" on page 29
- "Installation prerequisites" on page 38
- "Antivirus Exclusions" on page 24

Downloading and installing the software

In order to migrate to a new workstation, the software must already be installed on the new workstation. Follow the "Installation and activation" on page 37 guide to download and install the newest version of OL Connect on the new workstation.

If using Clustering in OL Connect Enterprise, please read the "Server Clustering" on page 124 page of this documentation for more details relevant to the installation.
Note that in a Clustering setup, both the Master and all Extensions must be upgraded to the same version.

Backing up files from the current workstation

The first step in migrating to a new workstation would be to make sure all necessary production files and resources are backed up and copied over to the new system.

> **Note:** Although it is not necessary to convert all of your documents when upgrading to the latest version, we strongly recommended doing so.
> It is considered "*Best Practice*" to convert the documents to the version installed and then re-send them to the Workflow Tools.

Backing up Workflow files

To save all Workflow-related files, backup the entire working directory:

```
C:\ProgramData\Objectif Lune\PlanetPress Workflow 8
```

Here are a few important points when transferring these files:

- If you are upgrading to the latest version of Connect, it is recommended to open each template in Designer, produce a proof making sure the output is correct. Then send the template with its data mapping configuration, Job and Output preset files to Workflow by clicking on **File > Send to Workflow...**

- If you still use PlanetPress 7 legacy documents, PTK files can be imported by clicking on the Workflow tool button at the top left corner of the Workflow tool interface. If copying the `PlanetPress Workflow 8` folder directly, it's important to delete any file with the .ps7 extension so as to refresh the PostScript file for the new workstation. Deleting the .ps7 files will make Workflow recreate them.
- The Workflow configuration file itself is named ppwatch.cfg, and is backed up with the folders. However, it needs to be re-sent to the Service to be used. To do this, rename the file to .OL-Workflow, open the file with the Workflow tool, and send the configuration.
- Locate Custom Plugins (.dll) from the below folder on the old workstation and import them onto the new workstation:
  `C:\Program Files (x86)\Common Files\Objectif Lune\PlanetPress Workflow 8\Plugins`
  To import the plugins:
  1. Start the Workflow Configuration Tool.
  2. Click on the Plug-in Bar.
  3. Click on the down pointing triangle under the **Uncategorized** group.
  4. Select **Import Plug-in** and select the .dll file.
- Import any external scripts used by the **Run Script** plugin, making sure they reflect the same paths as on the previous workstation
- Install any external application, executable and configuration files used by the External Program plugin, making sure they reflect the same paths as on the previous workstation
- Reconfigure local ODBC connections (i.e. create local copies of databases or recreate required DSN entries).
- Backup and import other custom configuration files, Microsoft Excel Lookup files, making sure they reflect the same paths as previously.
- Reinstall required external printer drivers and recreate all Windows printer queues and TCPIP ports.
- On the new workstation if the "TCP/IP Print Server" service is running in Windows, it is requested to disable that service so that it does not interfere with the Workflow LPD/LPR services.
- Configure the Workflow services account as in the previous installation. If accessing, reading and writing to network shares, it is recommended to use a domain user

account and make it a member of the local Administrators group on the new workstation. Once the user account has been chosen:

1. Click on Tools in the Workflow Configuration menu bar.
2. Click **Configure Services**.
3. Select the user account.

- If required, grant permissions to other machines (Designer clients and other servers) to send documents and jobs to the new server.
    - Click on Tools in the Workflow Configuration menu bar.
    - Click on **Access Manager**
    - Grant necessary permissions to remote machines.
    - Restart the Workflow Messenger service.
- Reconfigure the Workflow Preferences as previously by clicking on the Workflow button at the top left corner and clicking on Preferences:
    - Reconfigure the **Server Connection Settings** under **Behavior > OL Connect**.
    - Reconfigure each of the plugins, where necessary, under **Plug-in** as previously. Capture OnTheGo users may want to enable the **Use PHP Arrays** option under **Plug-in > HTTP Server Input 1**.
    - Send the configuration to the local Workflow service.

Backing up Connect Resources

The following resources are used by Connect and can be backed up from their respective folders:

- **Job Presets** (.OL-jobpreset):
  `C:\Users\[UserName]\Connect\workspace\configurations\JobCreationConfig`
- **Output Presets** (.OL-outputpreset):
  `C:\Users\[User-Name]\Connect\workspace\configurations\PrinterDefinitionConfig`
- **OL Connect Print Manager Configuration files** (.OL-ipdsprinter):
  `C:\Users\[UserName]\Connect\workspace\configurations\PrinterConfig`
- **OL Printer Definition Files** (.OL-printerdef):
  `C:\Users\[User-Name]\Connect\workspace\configurations\PrinterDefinitionConfig`
- **OMR Marks Configuration Files** (.hcf):
  `C:\Users\[UserName]\Connect\workspace\configurations\HCFFiles`

Where `[username]` is replaced by the appropriate Windows user name.

> **Tip:** Actually, the path may not begin with 'C:\Users', as this is language-dependent. On a French system, for example, it would be 'C:\Utilisateurs'.
>
> Type `%userprofile%` in a Windows File Explorer and press Enter to open the actual current user's home directory.

Other Resources

- **OL Connect Designer Templates**, **DataMapper** or **Package** files, copied from the folder where they reside.
- All PostScript, TrueType, Open Type and other **host based fonts** used in templates must be reinstalled on the new workstation.
- Import all **dynamic images** and make sure their paths match those in the old server.
- Make sure the new workstation can also access network or remote images, JavaScript, CSS, JSON, and HTML resources referenced in the Connect templates.

Secondary software and licenses

The following only applies to specific secondary products and licenses that interact or are integrated into the main product.

Image, Fax and Search Modules

- Reconfigure the Image and Fax outputs with the new host information.
- Import the Search Profile and rebuild the database in order to generate the database structure required by the Workflow.

OL Connect Send

- As of version 8.6 the Connect Send plugins are installed automatically with Workflow. If you are using an older version, run the OL Connect Send Plug-in Installer on the new Workstation to re-install the Connect Send plugins.
- Reconfigure the Server URL and port during the OL Connect Send Printer Driver setup.
- Re-run the OL Connect Send printer driver setup on client system and select the Repair option to point the clients to the new Server URL.

Configuring the Connect Engines

Any changes made to the Server preferences require the OLConnect_Server service to be restarted to take effect.

1. Stop the OLConnect_Server service from **Control Panel > Administrative Tools > Services > OLConnect_Server > Stop**.
2. Configure the Merge and Weaver Engines scheduling preferences as in the previous installation
    - Open the Server Configuration from: `C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.exe`
    - Configure the DataMapper, Merge and Weaver engines preferences (see "Parallel Processing preferences" on page 116). As of version 2018.1 these preferences include the minimum (Xms) and maximum (Xmx) memory utilization for the Server, Merge and Weaver engines.
    - Configure any other options for the Clean-up Service.
3. Now start the **OLConnect_Server** service

Configuring the Server Extensions

In the case where the OLConnect MySQL is installed on the new Master Server, it is important to reconnect all Server Extension systems to the new Master Server.

Perform the following action on each Server Extension:

1. Stop the OLConnect_ServerExtension service from **Control Panel > Administrative Tools > Services > OLConnect_ServerExtension > Stop**.
2. Open the Server Extension Configuration from: `C:\Program Files\Objectif Lune\OL Connect\Connect Server Extension\ServerExtension.exe`
3. Click on Database Connection and configure the JDBC Database connection settings so that the hostname points to the new Master Server.
4. Click on Scheduling and type in the location of the new Master Server.
5. Start the **OLConnect_ServerExtension** service.

Transferring software licenses

Once all the above resources have been transferred over to the new server, it is recommended to thoroughly test the new system - in **demo mode** - with sample files under

normal production load to identify points of improvement and make sure the output matches the user's expectation.
Output generated at this point will normally bear a watermark which can be removed by transferring licenses from the old server to the new one.

- To transfer Connect and Workflow licenses, the user is usually required to complete a **License Transfer Agreement** which can be obtained from their local Customer Care department.
- If you want to transfer your licenses to the new machine right away, you may ask your local Customer Care department for a **30 day Transition activation** code for your *old* machine.
- Upgrades cannot be activated using the automated Activation Manager. Contact your local Customer Care department.

To apply the license file received from the Activation Team:

1. Ensure that all services are stopped on your old machine before activating and starting the services on the new machine. Attempting to run the software with the same license simultaneously will not only run into errors but it is a breach of our EULA.
2. Start the OL Connect Software Activation module:
   `C:\Program Files\Objectif Lune\OL Connect\Connect Software Activation\SoftwareActivation.exe`
3. Click on Load License File to import the license.OLConnectLicense.
4. Start the Software Activation module on the Extension servers, where applicable.
5. Click on Load License File to import the above same license.OLConnectLicense.
6. Restart the OLConnect_Server service and restart the OLConnectServer_Extension service on the Extension servers, where applicable.
7. The number of Expected Remote Merge and Weaver engines should now be configurable in the Connect Server Configuration module (`C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.exe`)

Uninstalling OL Connect from the previous workstation

It is recommended to keep the previous install for a few days until everything is completed. However, once your transition is successful and complete, the OL Connect software must be uninstalled from the original server. See "Uninstalling" on page 143.

Information about OL Connect Workflow

If you wish to use OL Connect Workflow (automation), you will need to install OL Connect Workflow 2025.1 as well. Workflow 2025.1 is provided through a separate installer which is available on CD or for download as follows:

- If you are a **Customer**, the installer can be downloaded from the Upland Objectif Lune Web Activations page: http://www.objectiflune.com/activations
- If you are a **Reseller**, the installer can be downloaded from the Upland Objectif Lune Partner Portal: http://extranet.objectiflune.com/

OL Connect Workflow can be installed in parallel on the same machine as an existing PlanetPress® Suite 7.x installation.

Note however:

- If both versions need to be hosted on the same machine, OL Connect Workflow 2025.1 must always be installed after the legacy PlanetPress® Suite 7.x installation.
- When uninstalling OL Connect Workflow 2025.1, you may be prompted to repair your legacy PlanetPress® Suite 7.x installation.
- If OL Connect Workflow 2025.1 has been installed alongside PlanetPress® Suite 7, Capture can no longer be used with Workflow 7.
- OL Connect Workflow 2025.1 and PlanetPress® Suite Workflow 7 cannot run simultaneously, since only one version of the Messenger service can run at a time. In fact, no two versions of Workflow can be run simultaneously on the same machine, regardless of versions.
- It is possible to switch between different versions running by shutting down one version's services and then starting the other. However, this is not recommended. There are no technical limitations that prevent processes from previous PlanetPress Suite Workflow versions (as far back as Version 4) to run on OL Connect Workflow 2025.1, removing the need to run both versions.

For more information on the licensing of Workflow 2025.1, please see "Activating a License" on page 64.

Upgrading

This page provides information about Upgrading to OL Connect version 2025.1.

Upgrade information is detailed in the following pages:

- "Upgrading from previous OL Connect versions" below
- "Upgrading from PReS Classic" on page 81

Upgrading from previous OL Connect versions

Users of OL Connect prior to 2022

Users of any version of OL Connect prior to 2022 should see the page "Pre-existing User Data" on page 78 for information about exactly what data is saved or deleted.

Users of OL Connect prior to 2019.1

Users of OL Connect versions prior to 2019.1 should note that **Update Client 1.2.40** is a prerequisite for both OL Connect 2019.1 and Connect Workflow 2019.1 installations. Only Update Client 1.2.40 has the capacity to upgrade the OL Connect license to the newer format that is required by the installers of those products.

If you do not have Update Client version 1.2.40 installed already, then the next time you run your Update Client it will show that there is an update available of itself to Version 1.2.40 (or later).

Simply click on the "Install" icon to initiate the upgrade.

For further details on how to upgrade the Update Client and update your Connect License see the Update Client 1.2.40 Upgrade Guide.

Note: An incomplete uninstall of OL Connect before a reinstall or upgrade to a newer version can lead to issues. See "Issue after erroneous or incomplete update or re-installation" on page 134.

Before you upgrade

Always backup before upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see "Backup existing OL Connect version" on the facing page.

Note: The scheduling settings were changed significantly in version 2019.2. Please make sure to record your current scheduling settings for reference before proceeding with an upgrade..

Recommendations

Before embarking on an upgrade, also follow these recommendations:

- Prior to updating your production environment, all updates to OL Connect/Workflow should be performed in a **development & test environment**. This is both to test the upgrade process and to test that your solution is still working as expected. Having a development & test environment minimizes the risk of failure and business impact.
- **Planning** the upgrade:
    - Perform the upgrade of your production server during off-peak hours when it least impacts business.
    - Prepare a rollback plan appropriate to your organization, which includes provisions for reverting in the case of catastrophic errors. This can be as simple as reverting to a snapshot or may be more involved on physical hardware.
    - Anticipate at least 1 hour of downtime to provide enough time for the installation and any rollback plan.
- Consult the "System requirements" on page 35 and "Installation prerequisites" on page 38 before the upgrade to ensure that your environment is supported. If not, upgrading is not recommended.
- When possible, disable your **antivirus and anti-malware** software during the upgrade process. If it is not allowed by the security policies, please consider configuring the proper exceptions:
    - "Antivirus Exclusions" on page 24 (OL Connect
    - Workflow exceptions
- Read the "OL Connect Release Notes " on page 1722. They provide information regarding enhancements in that release version (known issues, bug fixes, enhancements, new features, etc.)

Backup existing OL Connect version

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing

production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

Backing up a virtual machine

Backing up a virtual machine installation is relatively straight forward. Simply take a snapshot of the virtual machine instance, prior to upgrading. This would save all the localized preferences and configurations.

Backing up a real machine

Backup these folders

- `C:\ProgramData\Objectif Lune\OL Connect\.settings\ConnectHostScope`
- `C:\Users\[UserName]\Connect\filestore`
- `C:\Users\[UserName]\Connect\workspace\configurations`
- `C:\Users\[UserName]\Connect\workspace\Designer\.metadata\.plugins\org.eclipse.core.runtime\.settings`
- `C:\Users\[UserName]\Connect\workspace\Server\.metadata\.plugins\org.eclipse.core.runtime\.settings`

Where `[username]` is replaced by the appropriate Windows user name.

Tip: Actually, the path may not begin with 'C:\Users', as this is language-dependent. On a French system, for example, it would be 'C:\Utilisateurs'.

Type `%userprofile%` in a Windows File Explorer and press Enter to open the actual current user's home directory.

Note that the installer is designed to keep these folders/files intact. Backing them up is simply a precautionary measure.

Additionally, these are only the folders that are natively managed by the installer. Back up every custom location/resource that may be important for the functionality of your solution.

Backup your database

If you want to be completely thorough and be able to exactly replicate your existing system, you should also backup your existing Connect database.

If the default (pre Connect 2022.1) MySQL database were being used as the Connect back-end database, we would recommend the MySQLDump tool be used for this. See for details on this utility program: mysqldump (https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html).

Pre-existing User Data

The following scenarios display what happens to pre-existing User Data in a Connect upgrade.

> **Note:** In regards to the MySQL (Connect version 2021.2 and earlier) or MariaDB (Connect version 2022.1 onwards) entries: these are only applicable if the OL Connect database component was installed in a previous Connect installation.

Scenario 1: Upgrading from OL Connect 2021.2 or earlier with **Remove User Data** CHECKED

- In all cases:
  - Files and folders are removed from the user data folder *C:\Users\<connectUser>\Connect* , where *<connectUser>* is the user that installed OL Connect.
  - Files and folders are removed from the following data folders:
    - *C:\ProgramData\Objectif Lune\OL Connect\.settings*
    - *C:\ProgramData\Objectif Lune\OL Connect\CloudLicense*
    - *C:\ProgramData\Objectif Lune\OL Connect\ErrorLogs*
    - *C:\ProgramData\Objectif Lune\OL Connect\LiquibaseUpdate*
  - Files are removed from the root of the data folder *C:\ProgramData\Objectif Lune\OL Connect*.
    If the folder is empty following this (i.e. no license or user folders were present) then the *C:\ProgramData\Objectif Lune\OL Connect* folder itself is removed.
  - License files as well as any content not listed above but found in the *C:\ProgramData\Objectif Lune\OL Connect* folder remain untouched.
- Additional cases:

1. If MySQL was previously installed as an OL Connect component AND the database contains some user-defined schemas:
    - The native OL Connect schema is removed from the MySQL database.
    - The MySQL database files (*C:\ProgramData\Objectif Lune\OL Connect\MySQL*) are kept intact, as user-defined schemas mean that the user did not have only the OL Connect native schema content in their database.
        - A message at the end of the upgrade will advise the user that some non-OL schemas were found in the database, so the database files were not removed.
2. If MySQL was previously installed as an OL Connect component AND the database does not contain any user-defined schemas:
    - The MySQL database files (*C:\ProgramData\Objectif Lune\OL Connect\MySQL*) are removed entirely.

Scenario 2: Upgrading from OL Connect 2021.2 or earlier with **Remove User Data** UNCHECKED

- In all cases:
    - The user data folder *C:\Users\<connectUser>\Connect* (where *<connectUser>* is the user that installed OL Connect) is retained, untouched.
    - All the files and folders under the data folder *C:\ProgramData\Objectif Lune\OL Connect* remain untouched.
- If MySQL was previously installed as an OL Connect component:
    - All schemas from the MySQL database are migrated to MariaDB, allowing the user to continue using their database content normally.
      NOTE: This might take some time during the installation, depending upon the size of the existing databases.
    - The MySQL database files (*C:\ProgramData\Objectif Lune\OL Connect\MySQL*) are also kept intact.

Scenario 3: Uninstalling OL Connect 2022.1 or later with **Remove User Data** CHECKED

- In all cases:
- Files and folders are removed from the user data folder *C:\Users\<connectUser>\Connect* , where *<connectUser>* is the user that installed OL Connect.
- Additional cases:
    1. If MariaDB was previously installed as an OL Connect component AND the database contains some user-defined schemas:
        - The native OL Connect schema is removed from the MariaDB database.
        - The MariaDB database files (*C:\ProgramData\Objectif Lune\OL Connect\MariaDB*) are kept intact, as user-defined schemas mean that the user did not have only the OL Connect native schema content in their database.
            - A message at the end of the upgrade will be displayed, stating some non-OL schemas were found in the database, so the database files were not removed.
    2. If MariaDB was previously installed as an OL Connect component AND the database does not contain some user-defined schemas:
        - The MariaDB database files (*C:\ProgramData\Objectif Lune\OL Connect\MariaDB*) are removed entirely.

Scenario 4: Uninstalling OL Connect 2022.1 or later with **Remove User Data** UNCHECKED

- In all cases:
    - Files and folders are removed from the user data folder *C:\Users\<connectUser>\Connect* , where *<connectUser>* is the user that installed OL Connect.
    - Files and folders are removed from the following data folders:
        - *C:\ProgramData\Objectif Lune\OL Connect\.settings*
        - *C:\ProgramData\Objectif Lune\OL Connect\Address Blocks* (introduced OL Connect 2024.2)
        - *C:\ProgramData\Objectif Lune\OL Connect\CloudLicense*

        - *C:\ProgramData\Objectif Lune\OL Connect\ErrorLogs*
        - *C:\ProgramData\Objectif Lune\OL Connect\LiquibaseUpdate*
    - Files are removed from the root of the data folder *C:\ProgramData\Objectif Lune\OL Connect*.
      If the folder is empty following this (i.e. no license or user folders were present) then the *C:\ProgramData\Objectif Lune\OL Connect* folder itself is removed.
    - License files as well as any content not listed above but found in the *C:\ProgramData\Objectif Lune\OL Connect* folder remain untouched.
- If MariaDB was previously installed as an OL Connect component:
    - All schemas from the MariaDB database are kept, allowing the user to use those database files if they reinstall the software.

Upgrading from PReS Classic

PReS Classic and OL Connect are very different products.

Whilst OL Connect provides considerably more options for email and web output, one need not abandon existing PReS Classic print jobs. They can still be run through Connect Workflow, using the PReS Print Controls task.

Upgrading from PlanetPress Suite 6/7

> **Note:** This document is intended for people that purchased OL Connect and were previously users of PlanetPress Suite version 6 or 7. They should already have their new serial number(s) in hand and the OL Connect installers.

OL Connect embodies a true Production Print as well as an Interactive Business Communication Solution.

This document provides information on the migration process and the requirements and considerations for existing PlanetPress Suite users to upgrade to the latest generation of our products.

What does OL Connect contain?

OL Connect is comprised of the following modules:

- **OL Connect Workflow** 2025.1. This is the natural evolution of PlanetPress Suite Workflow 7 (Watch, Office or Production). OL Connect Workflow 2025.1 is very similar to the PlanetPress Suite Workflow 7 version but contains a number of new features and has

the ability to run OL Connect jobs, as well as PlanetPress Suite, PrintShop Mail Suite and PReS Classic documents.

- **Imaging** for OL Connect is available as an option. It contains:
  - OL Connect Fax
  - OL Connect Image
  - OL Connect Search

  IMPORTANT: If you owned them, you must also upgrade your Imaging modules to use the new version.

- OL Connect **Designer**. This is a design tool based on completely new technology. It is not backwards compatible and therefore cannot open PlanetPress Suite Design 7 documents. *If you want to continue editing those documents you can keep doing so in PlanetPress Suite Design 7.*
- OL Connect **Server**. This is the core of the Connect technology (see "OL Connect: a peek under the hood" on page 16 in the Online Help: https://help.upplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html). This module automates the merging of data with your new templates and generates the output. It is required for OL Connect Workflow 2025.1 to handle templates created with the OL Connect Designer. It can be installed on the same or a different machine as OL Connect Workflow 2025.1.

IMPORTANT: OL Connect does **not** contain the PlanetPress Design 7.

OL Connect does not need any *printer licenses* to print from OL Connect or PlanetPress Suite. It can also print PrintShop Mail 7 and PReS Classic documents if these programs are licensed.

You can keep everything you have

The first thing to know is that you can keep your current PlanetPress Suite Workflow 7 configuration and your PlanetPress Suite Design documents. When upgrading to OL Connect, they will remain functional.

Please note that PlanetPress Suite Workflow 7 and OL Connect Workflow 8 cannot run at the same time. See "Information about OL Connect Workflow" on page 74 for information about these limitations. The only exception is the PlanetPress Suite Design tool that you can continue to use as it is not part of OL Connect.

OL Connect installation considerations

The PlanetPress Suite could run on a computer with a minimum of only 1GB of RAM available. The OL Connect Server with OL Connect Workflow 2025.1, by default, requires 8GB of RAM, but if you intend on using the new OL Connect Designer on the same computer, you should consider having at least 16GB of RAM available. See "System requirements" on page 35.

Distributed installation or not

You can decide to install OL Connect modules all on the same computer or have each module on a different computer. Reasons for this could be:

- There is insufficient memory in the computer currently running OL Connect Workflow 2025.1 to also run OL Connect Server.
- You want to use a more powerful computer with more RAM and more cores to run the Server to achieve maximum performance (see "Performance considerations" on page 33).

What do I gain by upgrading to OL Connect?

When upgrading to OL Connect, PlanetPress Watch users receive key features of PlanetPress Office such as the following:

- Ability to input data from PDF
- Ability to print your PlanetPress Suite documents on any Windows printer (no need for printer licenses)
- Ability to create standard PDF output from your PlanetPress Suite documents
- Even if you don't recreate your existing PlanetPress Suite documents, you can easily change your workflow to convert your output to PDF, then output them in PCL to any device supporting it.
- The full version of PlanetPress Connect can open your company to the digital world by enabling you to send HTML responsive emails as well as creating dynamic responses and interactive web pages.
  You can reuse the content of your existing documents and map it onto responsive documents that can be sent by email in full HTML glory and/or make them available as native HTML web pages using the latest CSS/JavaScript features.

Note: If you were a PlanetPress Production user, you retain all functionalities within OL Connect Workflow 2025.1. These are automatically imported during the activation (see below).

Create new documents and integrate them into your workflow at your own pace

You can start benefiting from the innovative technology of the new OL Connect Designer right away by designing new documents, or re-doing existing ones at your own pace. You can also now:

- Use the new DataMapper to easily map any input data into a clean data model that any designer person can use.
- Easily create documents with tables that spread over multiple print pages, respecting widow and orphan rules, displaying sub-totals and totals properly.
- Have text that wraps around images.

Upgrade steps

1. To upgrade to OL Connect, the first step is to stop your PlanetPress Workflow services. You can do so from the PlanetPress Workflow configuration tool or from the Windows Service Management console.
2. Then, using the OL Connect setup, install the Designer and/or Server on the appropriate computers.
3. Then, using the OL Connect Workflow 2025.1 setup, install OL Connect Workflow and/or OL Connect Image on the appropriate computers. (See "Installation and activation" on page 37 for more details.)

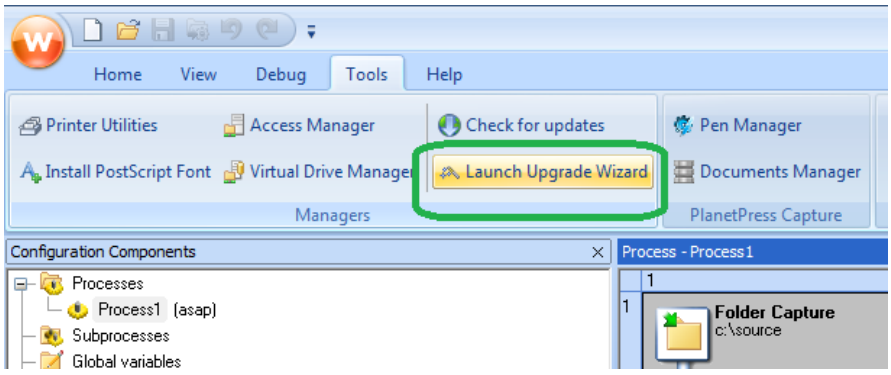
   Note: If Workflow installation finds that .NET 4.6.2 (or later) is not already installed, it will prompt you to install this manually.

4. If you installed OL Connect Workflow 2025.1 on the same computer where you had PlanetPress Suite Workflow 6 or 7, you can use the Upgrade Wizard to import your:
   - PlanetPress Workflow:
     - Processes configuration
     - PlanetPress Suite compiled documents
     - Service configuration

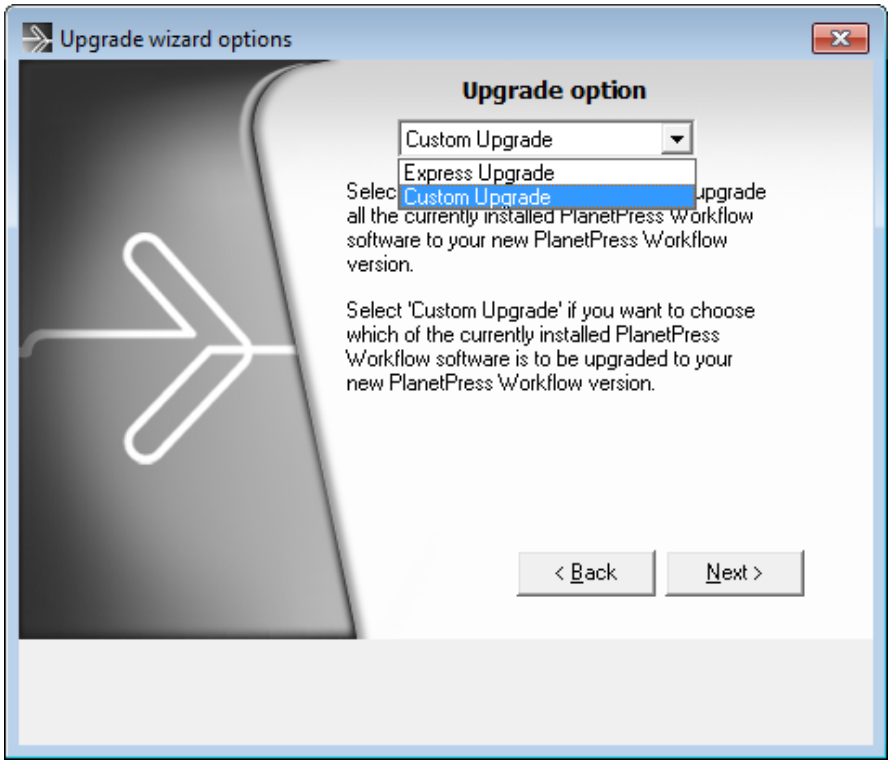
        - Access manager configuration
        - Custom plug-ins
    - PlanetPress Fax settings
    - PlanetPress Image settings
    - PlanetPress Search profiles
    - Printer activation codes
    - PlanetPress Capture database
    - PlanetPress Capture pen licenses
    - Custom scripts
    - Content of your virtual drive
    - PlanetPress Messenger configuration

5. If you installed OL Connect Workflow 2025.1 on a different computer, please see "How to perform a Workflow migration" on page 90 for help importing all those settings, if you wish to import them.
6. To launch the Upgrade wizard, open the OL Connect Workflow 8 configuration tool and, from the Tools menu, launch the Upgrade Wizard.

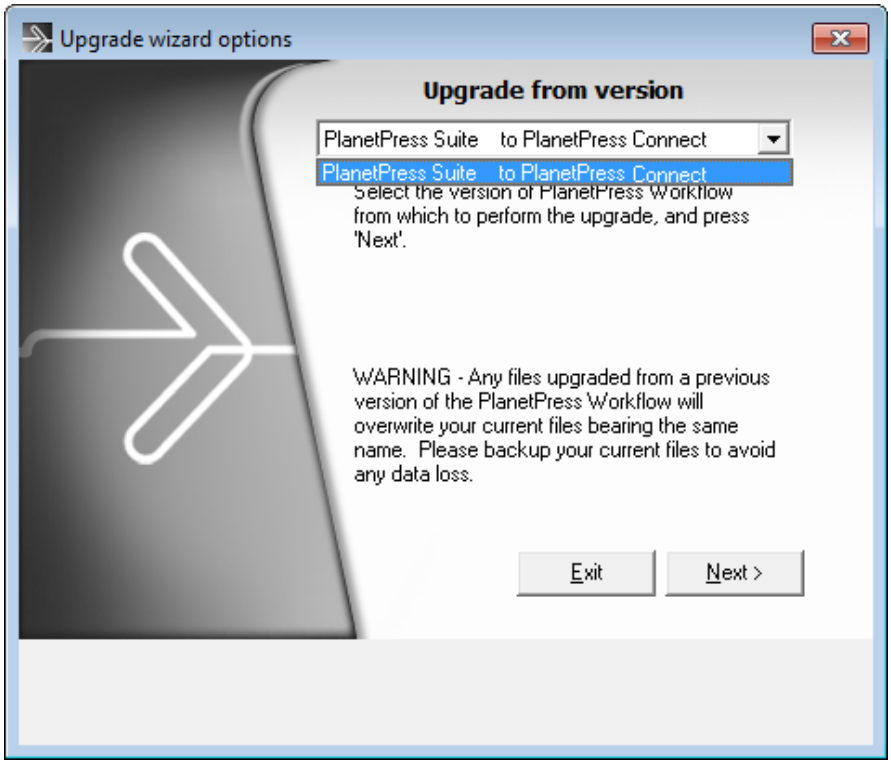
    IMPORTANT: Before you start this process, make sure you have a backup of your current installation/computer.



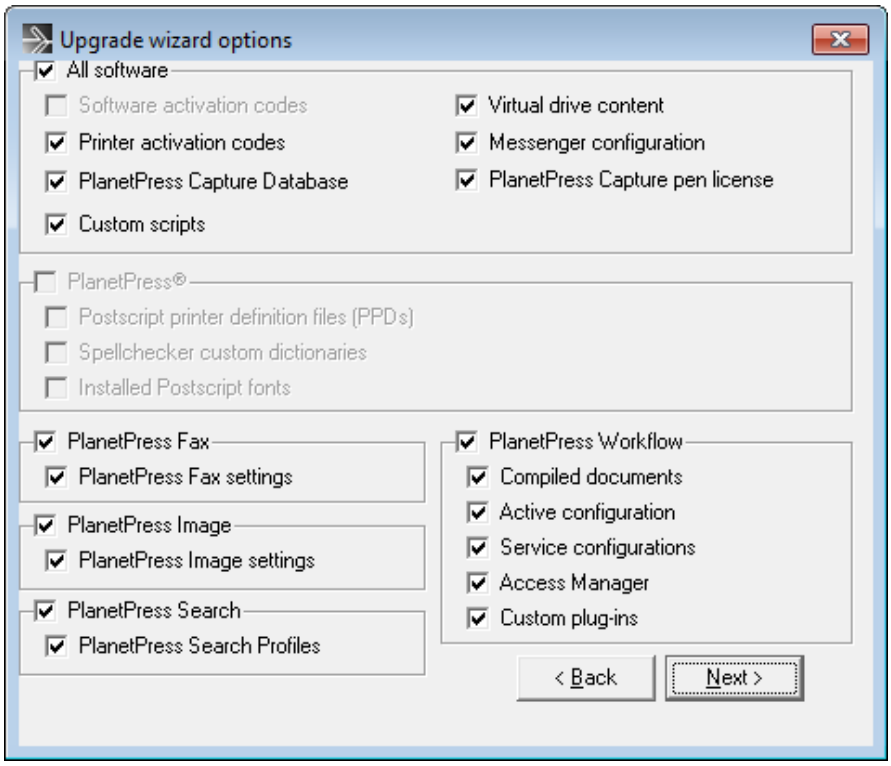
7. Then select your upgrade type:



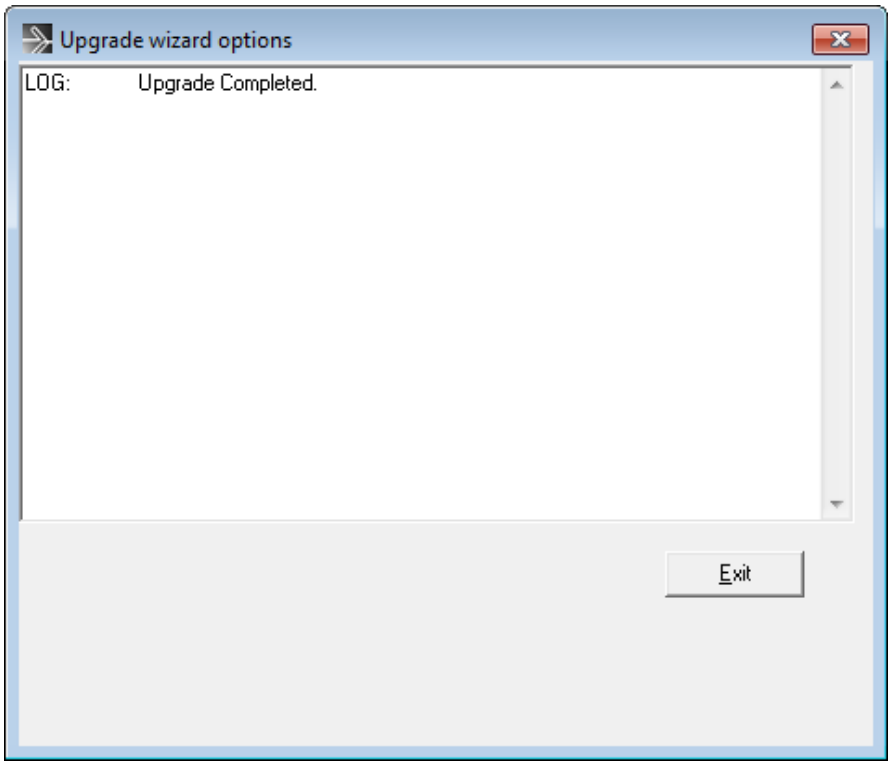
8. Then select the product from which you wish to upgrade:



9. If you selected to do a Custom upgrade, select the required options:



10. Then finally review the log in the final dialog for details on how it went:



11. After that you will need to get the activation file for your product.
    To obtain your activation file, download the OL Connect installer from the Web Activation Manager (http://www.objectiflune.com/webactivationmanager/), follow the instructions for the installation using the serial number provided to you. You can activate your license through the Web Activation Manager.
12. From now on, if you need to modify your PlanetPress Design documents, simply open PlanetPress Design 6 or 7, edit your document and send the updated version to OL Connect Workflow. In order to do that:
    - If you have PlanetPress Design on the same computer as OL Connect Workflow 2025.1, you need to save the documents to PTK by using the “Send to” menu, then "OL Connect Workflow”, and there use the “Save to file” button. Then, from the OL Connect Workflow configuration tool, in the “Import” menu, select “Import a PlanetPress Document” and select the previously saved file.
    - If you have PlanetPress Design on a computer and OL Connect Workflow 2025.1 on another, you can simply use the “Send to” menu in the Designer and select the OL Connect Workflow tool to which you want to send the PlanetPress Design document.

How to perform a Workflow migration

What do you need to consider when upgrading from PlanetPress Suite 7 to OL Connect Workflow 2025.1 on a new computer?

Installing and Activating Workflow 2025.1 on a new computer

Points to consider:

- Before installing, be sure to read "Installation and activation" on page 37. There you will find detailed Connect Workflow installation steps as well as system requirements, notes on license activation and much more.
- It is recommended you retain your existing PlanetPress Suite installation for a period of time after the OL Connect Workflow 2025.1 installation. We recommend this particularly when undertaking migration from one to the other. Once the migration has completed, you should uninstall PlanetPress Suite from your original installation. In the meantime, a fresh installation of OL Connect will run for 30 days without requiring an activation code, to simplify the migration process.
- Request new activation codes for your software licenses (License Transfer agreement needs to be filled out and signed). Contact your local Activations Department. www.objectiflune.com/activations
- Please note that PlanetPress Suite Workflow 7 and OL Connect Workflow 8 cannot run at the same time. See "Information about OL Connect Workflow" on page 74 for information about these limitations. The only exception is the PlanetPress Suite Design tool that you can continue to use as it is not part of OL Connect.

Printer Licences

If you are currently using Printer Licenses under PlanetPress Suite 7 and wish to continue doing so in OL Connect Workflow 2025.1, there are a few ways in which you can reinstall those printer activation codes onto OL Connect Workflow 2025.1. They are as follows:

- If you retained the .pac file (printer activation codes) from your previous installation, then double click on that file from within your new computer, and the printers will get activated.

  If you did not retain the pac file, you can export a new printer activation code. This is done from the PlanetPress Suite Designer **Help > Printer Activation** menu option. When the "*Activate a printer*" dialog is launched, right click within it and select the *Export* context menu option, then save the file on the new computer. Double clicking

on the .pac file will then activate all of your printers on the new computer.

- Login to our Web Activation Manager (www.objectiflune.com/activations) using your customer number and password to get your Printer Activation Codes.
- If you do not have access to the computer in which PlanetPress Suite was previously installed, print a Status Page for each printer from your Connect Workflow 8 Configuration. Do this via the **Tools > Printer Utilities** menu option. Select "*Print Status Page*" and then select your printers from the list.

  Email the Status Page(s) to activations@ca.objectiflune.com and you will receive a .pac file in return, with which you can activate your printer(s).

Documents and Resources

PlanetPress Suite Documents and Resources

- Backup all your PlanetPress Suite Design documents from your old computer and copy them onto the new computer. The files use the extension .ppX, where X is the version number of the PlanetPress Suite that created the files.
  The documents do not have to be in any specific folder.
- Back up the entire directory of: "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PlanetPress Watch\Documents*".
  This folder contains all the PlanetPress Design documents and compiled forms (*.ptk and *.ptz).
  Paste the files onto the new computer in the following folder:
  "*C:\ProgramData\Objectif Lune\PlanetPress Workflow 8\PlanetPress Watch\Documents*"
- Back up the latest .pwX (PlanetPress Workflow Tools Configuration) file, found here:
  "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PlanetPress Watch*".
  Paste onto the new computer in the following folder:
  "*C:\ProgramData\Objectif Lune\PlanetPress Workflow 8\PlanetPress Watch*"

There are several ways you can import Documents into PlanetPress Workflow. They are as follows.

1. In Connect Workflow go to **File > Import > PlanetPress Document ...** and select the .ptk document you wish to import.
   These files will most likely be found in the Documents folder on the PlanetPress Suite computer:
   "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PlanetPress Watch\Documents*"

2. Copy all the PlanetPress Suite 7 Documents and Compiled forms (*.ptk and *.ptz) from the Documents folder on the PlanetPress Suite computer and paste them into the equivalent folder on the Connect Workflow Computer.
   The PlanetPress Suite 7 folder would be "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PlanetPress Watch\Documents*".
   The OL Connect Workflow 8 folder will be "*C:\ProgramData\Objectif Lune\PlanetPress Workflow 8\PlanetPress Watch\Documents*"
3. Use the **File > Send To** menu option in PlanetPress Suite Designer and select the OL Connect Workflow 8 to which you want to send the PlanetPress Suite Designer document.
   This should work with PlanetPress Suite versions 6 and 7.

   Make sure that ports 5863 and 5864 are not blocked by firewall on either machine.
   Also make sure you add the PlanetPress Suite machine's IP address to the permissions list in Connect Workflow 8 from **Tools > Access Manager**.
   Further information about Workflow Access Manager can be found here: Access Manager.

Windows Operating System Steps:

- Install all the Windows printer queues from the old computer, making sure they are named the same.
- If your existing documents referenced any local dynamic image resources in a folder or in Local Host, make sure that you import them onto the new computer as well, or make them available on a network accessible drive.
- Any special PostScript or TrueType fonts used will also need be installed on the new computer.
- Verify that you have access to any other resources that the PlanetPress Suite used. This includes network folders, printers, third party software and the like.

Workflow Plug-ins

Back up any custom PlanetPress Suite Workflow configuration Plug-ins (.dll) and copy them onto the new computer.

The PlanetPress Suite Workflow plug-ins folder can be found here: "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PlanetPress Watch\Plugins*".

Make sure that you copy only the custom plug-ins.

Alternatively, you can download custom plug-ins from http://planetpress.objectiflune.com/en/suite/resources/support onto the new computer.

Once you've copied your PlanetPress Suite Workflow configurations to Connect Workflow, you can confirm their availability through the Plug-in Bar **Uncategorized** category. There you will find all the Custom plug-ins that have been installed.

Missing plug-ins will be represented in Workflow steps through the use of a "?" icon. Such as in the following image, which shows that the "*TelescopingSortPlugin*" is not installed.

To import a plugin:

1. Click on the popup control ( ) in the Plug-in bar.
2. Select **Import Plugin**
3. Browse to the location of the plug-in DLL file
4. Click on Open.
5. The new plug-in should appear in the Plug-in Bar **Uncategorized** category.

Configuring OL Connect Workflow

- Reconfigure any settings that may need to be applied to the PlanetPress Suite Messenger and PlanetPress Workflow Tools LPD services using the Access Manager (see https://help.uplandsoftware.com/objectiflune/en/olconnect-workflow/2025.1/Workflow/Interface/Access-Manager.html).

- All PostScript and TrueType host based fonts must be reinstalled. Make sure you restart the computer after this step.
- If necessary, reconfigure local ODBC connections. (i.e. create local copies of databases or recreate required DSN entries)
- Manually install all external executables that will be referenced by the Connect Workflow processes in the configuration file. If possible, retain the local path structure as used on the older installation.
- If the Windows "TCP/IP Print Server" service is running on the new computer, it is recommended that you disable the Server so that it does not interfere with the PlanetPress LPD/LPR services.
- If you are using images from a virtual drive, copy the entire contents of "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PSRIP*" and paste them onto the new computer here: "*C:\ProgramData\Objectif Lune\PlanetPress Workflow 8\PSRIP*".
- Make sure to set the user who will run the PlanetPress Services. This is done by going into Tools/Configure services. The user will need to have local administration rights in order to be able to run the services.
  For more information, see Users and Configurations (https://help.uplandsoftware.com/objectiflune/en/olconnect-workflow/2025.1/Workflow/WorkflowServices/Users_and_Configurations.html).
- Once all these steps have been completed, you will need to import your configuration file. Find the latest pwX file located on the old computer, if it is not already copied across to the new computer. The default location on the old computer is "*C:\ProgramData\Objectif Lune\PlanetPress Suite 7\PlanetPress Watch*".

  On the new computer you will need to go to **File > Import > Configuration Components**. Browse and find your file. If the file is not visible change the file type to *.pw7

PlanetPress Image, Fax and Search

- Reconfigure the PlanetPress Image and PlanetPress Fax outputs with the new host information.
- You must import the Search Profile and rebuild the database in order to generate the required database structure.

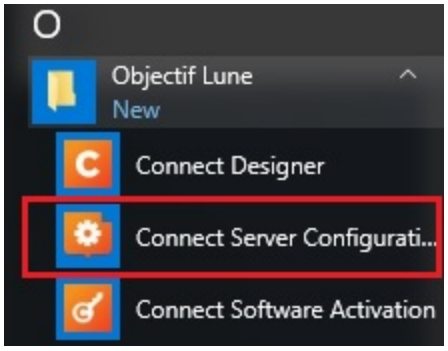
Server configuration settings

This chapter describes configuring the OL Connect Server.

In the Enterprise edition of OL Connect you can install Servers on multiple machines, in a Master/Client relationship. To see an overview of these Master/Client setups, please see the Enterprise only "Server Clustering" on page 124 topic.

The Connect Server settings are maintained by the **Connect Server Configuration** utility tool which is installed alongside OL Connect.

Connect Server Configuration can be launched from the Start Menu, as seen in the following screenshot:



The **Connect Server Configuration** dialog is separated into individual pages, where each page controls certain aspects of the software.

The following pages are available:

- "Clean-up Service preferences" on page 1035
- "Connection preferences" on the facing page
    - "Security and Users Settings" on page 99 (only available in the Enterprise edition, but not for Enterprise edition **Server Extension (Client)** installations)
- "Database Connection preferences" on page 1041
- "Engines preferences" on page 109
    - "Automatic Restart settings" on page 108 (only available in the Enterprise edition, but not for Enterprise edition **Server Extension (Client)** installations)

      Note: Automatic Restarts are controlled by the main server. Server Extension (Clients) are treated the same as local engines in regards to restart timing.
- "Hardware for Digital Signing preferences" on page 1053
- "Language preferences" on page 116

- "Logging preferences" on page 1055
- "Parallel Processing preferences" on page 116

Connection preferences

Background

The Connection preferences are a way to control precisely how to connect to the OL Connect Server.

This preference page was added in Connect 2018.2 to simplify management of HTTP communication with OL Connect. HTTPS communication options were then added in OL Connect 2021.1.

These preferences allow Connect inter-engine communication to occur on an alternate port, reducing the chances of inter-engine communication being starved of connections when running on a system that processes very large numbers of HTTP/HTTPS connections.

In OL Connect Enterprise, the **Connections** choices available to you will be dependent upon whether the installation is a **Master** (or **standalone**) installation or a **Server Extension (Client)** installation.
For **standalone** and **Master** installations, see "Connection settings (standalone/Master)" below.
For **Server Extension (Client)** installation see "Connection settings (Client)" on page 98.

Connection settings (standalone/Master)

- **REST Services** group. Use this to adjust the HTTP or HTTPS communication settings for Connect.
    - **Port**: Set the primary HTTP or HTTPS Server connection port number for Connect.
    - **Protocol**: Select whether to use HTTP or HTTPS.
    - **Restrict access to localhost (127.0.0.1)** checkbox:

      This option allows you to specify whether the server restricts access to localhost (ON) or allows requests from remote locations (OFF).

      By default on new installations, the option shall be set to ON (the server restricts remote access). This applies to both the interactive and the silent installer.

On existing installations, the option is set to OFF, to maintain backward compatibility.

If HTTPS is selected, the following HTTPS specific options become available:

- **Root certificate**: This is an optional selection. Use the Browse button to locate the appropriate *root certificate* file, if one is available.
- **Server certificate**: Use the Browse button to locate the appropriate *server certificate* file. This is a mandatory selection.
- **Private key**: Use the Browse button to locate the appropriate *Private key* file. This is a mandatory selection.
- **Private key Password**: Enter the *Private key* password.
- **Confirm password**: Re-enter the password, to confirm it was entered correctly.

> **Note:** Enabling HTTPS for the primary REST Services connection requires different ports be used for the internal connections (IPC and REST) between Connect Server and its engines.

- **Internal communication on separate ports** checkbox:
  Set the internal connection communication ports. These settings are purely for Connect inter-engine communication.
  It is recommended that this option be selected.

  > **Note:** If HTTPS were selected as the **REST Services** protocol, these internal Port settings must be entered.

  > **Note:** Please note that local security settings (including firewalls) must be taken into consideration when setting Port entries.

  - **IPC Port**: Set the internal connection port number. This cannot be the same at the primary REST Services port number.
  - **REST Port**: Set the internal REST connection port number. This cannot be the same at the primary REST Services port, or the **IPC Port**.
- **Advanced** group
  - **Maximum threads**: Sets the maximum number of HTTP threads for processing requests.
    This entry should only be changed in consultation with OL.

- **Maximum asynchronous tasks**: Set the maximum number of requests that can be executed simultaneously.
  The default asynchronous task setting is twice the number of processing cores available to the computer. This can be expanded, if the limitation is deemed a bottleneck.

  Note: This does not limit the number of requests that can be received, just how many are processed in parallel. Additional requests are buffered and are processed as capabilities allow.

Connection settings (Client)

- **Master Server** group. Use this to set connection to OL Connect **Master** server. (OL Connect **Enterprise edition only**.)
  - **Hostname**: Set the device name for the OL Connect **Master** server.
  - **Port**: Set the port number for the OL Connect **Master** server.
  - **Username**: The account that the services use to login to the OL Connect **Master** server. If the machine is on a domain, use the format domain\username.
  - **Password**: Enter the password associated with selected username.
  - **Initial retry interval (secs)**: The retry interval when initially trying to connect (or reconnect) to the OL Connect **Master** server.
  - **Prolonged retry interval (secs)**: The retry interval for trying to connect (or reconnect) to the OL Connect **Master** server after the initial retry limit period has been exceeded.
  - **Initial retry limit (mins)**: How long the initial connection (or reconnection) attempts should run for.

Buttons

The **Connections** preferences also provides you with buttons to :

- **Test Connection**. This button is available only on OL Connect (Enterprise edition) **Server Extension (Client)** installations .
  When the button is pressed an attempt is made to connect to the OL Connect **Master** server, using the location, user name and password set in this preference page.
  If the connection is successful, a confirmation message is displayed in the top area of the preferences page.
  If the test fails, a warning message is displayed in the top area of the preferences

page. Set logs for further information.

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Security and Users Settings

Note: This option is only available in the Enterprise edition of OL Connect.

This dialog controls the security settings for external applications connecting to the OL Connect Server, such as OL Connect Designer, OL Connect Workflow plugins, or scripts communicating through the REST API.

Caution: It is **highly recommended** to keep security enabled and change the password on any server accessible from the web.

If these precautions are not taken, data saved in the server may be accessible from the outside!

- **Allow cross-origin resource sharing**: Chose whether or not to allow cross-origin resource sharing.
- **Enable server security**: Enable to add authentication to the REST server.
  - When enabled, the username and password (which cannot be blank) of an authorized user must be entered in any remote OL Connect Designer that links to this Server. See the "Connect Servers preferences" on page 1061 sub-section of the Designer Preferences dialog.
  - When disabled, a username and password is not required to make REST request, and tasks in OL Connect Workflow do not require them in the Proxy tab. Nor would a username and password be required on any remote OL Connect Designer that links to this Server.
- **Session expiration time (mins)**: Select a session time (in minutes, between one minute and one year) that the authentication stays valid for the requested process. This can reduce the number of requests to the server since an authentication request is not necessary during the session.
- **Authorized Users** table: The Table contains a list of all currently Authorized Users. All required information is summarized in the table, which has buttons to the right which allow one to add ( ), edit ( ) or delete ( ) individual entries. Double clicking

on any entry in the table also launches the Edit dialog.

The options available in the Add/Edit dialogs are as follows:

- **Username**: Enter the username for the server security.
- **Roles**: Check the roles that apply to the user.
  - **Data Handler**: The user can start and stop operations, get results of operations, and view and handle data (except resources). This appears as ROLE_DATA in the table.
  - **Resource Handler**: The user can manage resources like data mapping configurations, templates, and print presets. This appears as ROLE_RESOURCE in the table.
  - **Monitoring**: Can monitor the server, but cannot see data, change server settings, or modify resources.This appears as ROLE_MONITOR in the table.
- **Password**: Enter a password for this user for the server security. There are no specific password requirements or restrictions.
- **Confirm password**: Re-enter the password assigned to this user.

Note: Existing client sessions are not affected by changes to authorized users. Clients may only see the effect after their session expires.

Engine configuration

Note: The options available will differ, based upon whether your edition of OL Connect is the Enterprise edition or Professional edition. Where the options differ, this is reflected in the text.

The Connect Server cooperates with different engines to handle specific tasks. A **DataMapper engine** extracts data from a data file. A **Merge engine** merges the template and the data to create Email and Web output, or to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. A **Merge engine** merges the template and the data to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. (For more information see: "OL Connect: a peek under the hood" on page 16).
Settings for these engines are made in the **Connect Server Configuration** tool (see "Server configuration settings" on page 94).

Connect allows for the parallelization of jobs. This means you can allocate 1 or more engines to process jobs.

OL Connect Enterprise edition only: The number of each type of engine - on each machine in a Server Clustering setup - is configurable, as well as the amount of Merge engines than can work together on the same job (determined by job size: small, medium or large) and at what maximum speed.

OL Connect Professional edition: The number of each type of engine is configurable, as well as the amount of Merge engines than can work together on the same job (determined by job size: small, medium or large) and at what maximum speed.

The "Parallel Processing preferences" on page 116 allow you to control precisely how the OL Connect Server handles jobs.

This gives you, as solution developer or application manager, full control of how to apply a machine's power. For example, you can share the available resources to process multiple jobs at once or allocate all resources to process one single job as fast as possible, or anything in between.

OL Connect distinguishes 5 type of jobs:

1. Small Print
2. Medium Print
3. Large Print
4. Email
5. Web

OL Connect categorizes print jobs on the number of pages they will produce. What constitutes a **Small**, **Medium** or **Large** job can be configured per server (see below, in "Allocating processing power to jobs" on page 105).
There is no distinction between small, medium and large jobs for Email and Web output.

This topic explains all of these settings and the principles behind them, and it provides guidelines for letting the Server manage the workload in such a way as to achieve the highest possible output speeds.

Factors to take into account are:

- Your license, which imposes a **speed quota** (see "Speed quota: Pages Per Minute" on the facing page).

- The **processing power** of your machine. How many cores it has determines how many engines can be launched (see "Launching multiple engines" on the next page).
- The size and number of jobs of one kind that need to be handled, sequentially or simultaneously. In other words, your **use case**. By allocating processing power to jobs of different sizes you can make the setup match your usage situation (see "Allocating processing power to jobs" on page 105).

Tip: Other ways to enhance performance are described in another topic: "Performance considerations" on page 33.

Speed quota: Pages Per Minute

The highest possible output speed depends first and foremost upon your license.
The maximum is 10000 Pages Per Minute (PPM) with the Professional edition, and 80000 PPM with the Enterprise edition.

Note: PPM in this case stands for Pages *and* emails *and* web pages Per Minute.

The number of engines that are allowed to operate in parallel to create the same type of output are referred to as the **Licensed task limit**.
OL Connect Enterprise edition provides 64 Licensed tasks.
OL Connect Professional edition provides 8 Licensed tasks.

Several engines can create the same type of output in parallel.
One engine needs *at least* one free speed unit to be able to create output.

It is important to note that only **output operations** are limited by this quota.

- **Weaver** engines **always** require a Licensed task to run.
- **Merge** engines **only** require a Licensed task when creating Email or Web output. Merge engines involved in a Print process don't need a Licensed task in order to run.
- **DataMapper** engines **don't** need Licensed tasks.

In situations where Print *and* Email and/or Web output are created at the same time, only the Merge engines that create Email/Web output count towards the maximum number of Licensed tasks for that type of output.

Spare speed units are distributed proportionally

Since the number of engines is configurable, and jobs may run concurrently, the number of engines in use may not match the exact number of available Licensed tasks.
When there are more Licensed tasks than there are engines in use, the OL Connect server

distributes the speed units and the maximum 'pages' per minute to all running jobs in proportion to the number of engines they are using.

> **Note: Output speed** is the speed at which the output is created by the engine in question. Data mapping and other steps in a production process are not taken into account. The **throughput speed** is the speed of the entire production process. This will always be lower than the output speed.

Launching multiple engines

One single engine can only process a single job at a time and will run mostly single-threaded.
In order to benefit from multi-core systems it is recommended that several engines run in parallel.

As a rule of thumb, you will want to run one less engine in total on a machine than the system has cores, leaving one CPU core free for the Connect Server and the operating system to use.

Modern hardware typically has both full cores and **hyper-threading** or logical cores. The logical cores should not be counted as a full core when determining how many engines to use. As a guide, count logical cores for only 25%-50% of a full core.
For example: on an Intel i7 CPU that comes with 4 cores and 4 additional hyper-threading cores, Windows Task Manager will show 4 cores and 8 logical processors on its performance tab. On a CPU like this, 5 or 6 engines can be configured to run in parallel.

To configure the number of engines:

1. Open the **Connect Server Configuration** utility tool (see "Server configuration settings" on page 94).
2. Under **Parallel Processing**, go to the *Content Creation* tab and set the number of Merge engines for the various tasks.
3. Go to the *Output Creation* tab and set the Reserved Weaver (Output) engines.
   See "Deciding how many engines of each type to launch" below.
4. Click **Apply** or **Apply and Close**.

It is advised that you *do not configure more engines than can be backed by actual processing power*. This adds overhead while not adding processing power.

Deciding how many engines of each type to launch

When jobs run in parallel, **different types** of engines may run at the same time. It depends on the usage situation which type of engines has the biggest impact on performance.

The more and the larger operations of a kind need to be performed simultaneously with smaller operations, the sooner you will see a performance increase when using multiple engines.

Merge engine

Generally, launching a relatively high number of **Merge** engines results in better performance, as Merge engines are involved in the creation of output of all kinds (Print, Email and Web) and because content creation is relatively time-consuming.

DataMapper engine

Adding DataMapper engines might be useful in the following circumstances:

- When large data mapping operations have to be run simultaneously for many jobs.
- When frequently using PDF or XML based input. Particularly in the case of XML input with large individual records.
- When the *All In One* plugin is used often in Workflow configurations and there are more than two Merge engines running.

The OL Connect MariaDB database needs a fast storage system (SSD or other fast devices) to be able to keep up with two or more DataMapper engines.
When the database is installed on a system with a slow hard drive, adding a DataMapper engine may not increase the overall performance.

Weaver engine

Adding extra Weaver (Output) engine(s) might be useful when large Print jobs are to be run simultaneously with smaller Print jobs.

Memory per engine

By default, each engine is set to use up to a predetermined amount of RAM. To make optimum use of a machine's capabilities it might be useful to increase the amount of memory that the various engines can use.

- DataMapper engines may perform better with greater memory when running jobs containing a lot of data.
- For complex templates with a lot of pages per document, there is a chance that Merge engines will run better with more memory.
- The maximum memory usage of a Weaver engine can be relevant for jobs with heavy

graphics; or for jobs that use Cut & Stack impositioning; or for jobs using particular variables that entail page buffering (see "Content variables" on page 1662).

The **Maximum memory per engine** setting is found in the "Engines preferences" on page 109.

These settings only control the maximum size of the *Java heap* memory that an engine can use; the total amount of memory that will used by an engine is actually a bit higher.

Also keep in mind that the Connect Server and the operating system itself will need memory to keep running.

Allocating processing power to jobs

Which engine configuration is most efficient in your case depends on how Connect is used. What kind of output is needed: Print, Email, and/or Web? How often? How big are those jobs? Do they have to be handled at the same time or in sequence? Would it be useful to give priority to small, medium or large jobs, and/or to jobs of a certain kind?

Depending on the answers to these questions, you can allocate processing power to jobs in order to run them as fast as possible, and/or in the order of your preference.

The first step in this process is to define the size of small, medium and large jobs.

Job size

Connect lets you define job sizes by setting the maximum number of pages a job can have and still be considered a **small** job, and what the minimum number of pages a job can have in order to be considered **large**. Jobs that fall between the small and large jobs are **medium** jobs.
Defining **small**, **medium** and **large** jobs is important, as you can assign additional resources to jobs that are considered either medium or large, via the settings for Merge and Weaver engines.

There is no recommendation regarding what number of pages constitute a small, medium or large job. Job size is a relative concept: in a small service company a job may be considered large when it outputs 1,000 pages, whereas that same job in a large insurance company might be seen as small. This setting needs to be based on an assessment of the actual (or expected) workload of Connect.

To set the job sizes:

1. Open the **Connect Server Configuration** utility tool (see "Server configuration settings" on page 94).
2. Under **Parallel Processing**, go to the *Output Creation* tab and enter the maximum number of pages in a small job.
3. Enter the minimum number of pages in a large job.
4. Click **Apply** or **Apply and Close**.

Medium jobs will be those that fall between the maximum pages of a small job, and the minimum pages of a large job.

The number of engines used for **small**, **medium** and **large** jobs is configurable (see below).

Running a job as fast as possible

Number of parallel engines per Print job

Two or more engines of a kind can be combined to work on the same Print job. Generally jobs will run faster with more than one engine, because sharing the workload saves time. However, running one job with multiple engines reduces the number of jobs that can be handled at the same time by that kind of engine, because there are only so many engines (and speed units) available.

> **Note:** When each individual **record** in a job is composed of a very large number of pages, the Memory per engine setting and the machine's hard drive speed are probably more important than the number of Merge engines, since one record cannot be split over multiple cores, or machines (Enterprise edition only) (see "Memory per engine" on page 104).

Target speed per Print job

If a Print job of a specific size has more than one parallel speed unit assigned to it, that *multiplies its speed*, however it *reduces the number of Print jobs* that can be run simultaneously.

When no other Print output operations run at the same time, a single job will use all available speed, or the maximum target speed reserved for jobs of that size (see "Dividing processing power over jobs" on the next page).

To set a number of speed units per Print job:

1. Open the **Connect Server Configuration** utility tool (see "Server configuration settings" on page 94).

2. Under **Parallel Processing**, go to the *Output Creation* tab.
3. Set the **Target speed when running simultaneous jobs** for small, medium and large Print jobs.
4. Click **OK** or **Apply**.

Dividing processing power over jobs

There is a number of ways in which you can divide processing power over output operations of a certain kind and/or size.

- By **reserving engines** for jobs of a certain **kind** (and **size**, in the case of a Print job). Note that reserved engines *cannot be used* by any other type of job. This means there will be fewer engines to handle other jobs. Consequently, the other jobs may take more time and may have to wait (or wait longer). However, if the server receives many web requests then having engines reserved for HTML output can help performance.
- By reserving a **number of parallel engines** for Print jobs of a certain size (see "Number of parallel engines per Print job" on the previous page). More parallel engines will make them run faster, but they will have to wait (longer) if the required number of engines isn't available when they come in.
- By **specifying target speeds for simultaneous Print jobs** of a certain size.

All of these engine configuration settings are found in the Parallel Processing Preferences:

1. Open the **Connect Server Configuration** utility tool (see "Server configuration settings" on page 94).
2. Under **Parallel Processing**, check out the information contained in both *Content Creation* and *Output Creation* tabs.

How the Server decides if a job can be handled

In summary, this is how jobs are handled when they can run in parallel.

- Whenever a job comes in, the number of engines to use is determined. (For Print jobs, this is based on whether the operation is small, medium or large; see "Job size" on page 105.)
- If there are enough **reserved** Merge engines for that type of job available then those engines will be used.
- If there are not enough reserved Merge engines available, then any unreserved Merge engine that is available will be used.

- If no, or not enough, Merge engines are available then the job will have to wait until the required number of appropriate Merge engines becomes available.

The following limitations apply at all times:

- The maximum number of concurrent Merge engines working on jobs of the same kind or size may not be exceeded.
- If no - or not enough - speed units are available for that type of output, the job must wait.

Examples

Here are a few examples of use cases and settings that would be appropriate in such cases.

Batch processing. In a batch processing situation, jobs don't have to be handled simultaneously. All jobs - whether they are big and small - are processed one after another. Every job should be handled as quickly as possible. It is therefor recommended to assign the maximum number of engines and target speeds to all jobs. Do not reserve engines for certain jobs.

Web requests. In online communication, response times are critical. If the Server receives a lot of Web requests, it should handle as many as possible, as quickly as possible, at the same time. It is recommended to launch as many Merge engines as possible and to reserve most of them for HTML output. The jobs will generally be small and can do with just one Merge engine.

Mixed jobs that are processed in parallel. In a situation where small, medium and large jobs can come in at any time and should be handled in parallel, the challenge is to find a balance between how much power can be allocated to jobs (to minimize the time they cost) and how long they can wait. No single job should require all of the processing power, *unless* it is acceptable for it to have to wait until the maximum number of engines finally comes available - and then all other jobs will have to wait.

Automatic Restart settings

> **Note:** This option is only available in the Enterprise edition of OL Connect.

> **Note:** This option is not available for Enterprise edition **Server Extension (Client)** installations

It is considered good management to restart the OL Connect Engines periodically. This dialog provides the controls for scheduling OL Connect Engines restarts.

Automatically restart engines to safeguard system availability

- **Time Limit** checkbox: This enables engine restarts to be scheduled to occur either after a specified period of time, or within a daily time window.
  This means daily Engine restarts can be scheduled to run outside of production hours.
- **Rolling restart** checkbox: This sets the engines restarts to occur in a "rolling" fashion, whereby each engine is stopped and restarted individually, one after the other.
  Each engine is only stopped when the previously restarted engine is once again ready to start processing.
- **Restart method**: Chose between restarting after a specified number of minutes (**Restart after interval**) or within a daily time window (**Daily restart in period**).
- **Restart after (minutes)**: Only available if **Restart after interval** selected.
  Specify the amount of minutes that the restarts are to occur after.
- **Daily restart period begin**: Only available if **Daily restart in period** selected.
  Enter the daily start time for the time window in which automatic restarts will be scheduled to occur.
- **Daily restart period end**: Only available if **Daily restart in period** selected.
  The end of the daily time window in which the automatic restarts are scheduled to occur.

Memory limit

Enter the memory limit for individual Engines. If any of these memory limits are exceeded, an automatic restart of those Engine types will be triggered.

- **Data Mapper Engine (MB)**: Enter the memory limit for Data Mapper Engine.
- **Merge Engine (MB)**: Enter the memory limit for Merge Engine.
- **Weaver Engine (MB)**: Enter the memory limit for Weaver (Output) Engine.

Engines preferences

Background

A **DataMapper engine** extracts data from a data file. A **Merge engine** merges the template and the data to create Email and Web output, or to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. A **Merge engine** merges the template and the data to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output.

This configuration page allows control over those Connect engine settings.

This page appears in a both the Designer Preferences and the Connect Server Configuration tool preferences, but with some different options between each.
There are also differences between OL Connect Master/Standalone and Client installations (OL Connect Enterprise edition, only).

For further background on Engine Configuration, see this page: "Engine configuration" on page 100.

Settings

Configure engine options

- **Maximum Data Converters** selection. This setting only appears in the OL Connect Server Configuration tool (Master/Standalone) Preferences, but not in the OL Connect Server Configuration tool (Client) Preferences or the Designer Preferences.

  AFP, PCL and PostScript input have to be converted to PDF before data mapping can be performed. For this, a DataMapper engine will request a conversion engine.

  > **Note:** AFP and IPDS are only available in OL Connect Enterprise edition.

  The selection here manages the maximum number of AFP/PCL/PostScript conversion requests that can be handled in parallel.

  When printing from the Designer, the Designer offloads AFP/PCL conversions to the Server but locally handles PostScript conversions.

  > **Note:** The option to use AFP input is included in the Enterprise edition of OL Connect.
  >
  > AFP input is dependent upon a third party library (CDP), which only allows OL Connect to run up to 4 AFP input processes on a given machine at a given time. In a Server Clustering setup these processes can only currently be run on the Main Server.

- **Tasks handled by**:
  This setting only appears in the OL Connect Server Configuration tool (Master \ Standalone) configuration page, but not in the Server Configuration tool (Client) configuration page.
  Select how Connect jobs are to be processed. The choices are between:

- **Engines** (separate processes): Select to have a number of separate engines (for Data Mapping, Merging and Weaver, as appropriate) to process jobs.

  > **Note:** This is the best choice for production or clustered environments.

- **Connect Designer** (uses less memory): /
  Connect Server (not scalable, use for low memory non-production situations):
  This setting differs between the Connect Server Configuration and the Connect Designer preferences.
  If selected in the Connect Designer preferences, this runs the engines as part of the Designer.
  If selected in the Connect Server Configuration preferences, this makes the engines run inside the Server.
  These options effectively mean the same thing: Do not run the selected engines as separate executables.

  > **Note:** This option is not scalable and should only really ever be used for low memory, non-production (testing) environments.

Number of engines

This group is only available if **Engines** were selected as the processing architecture , except in the OL Connect Server Configuration tool (Client) configuration page, where it is always available.

- **DataMapper Engines** selection: This selection is only available in the **Connect Server Configuration** utility. It is not available in the Connect Designer Preferences.

  A **DataMapper engine** extracts data from a data file. A **Merge engine** merges the template and the data to create Email and Web output, or to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. A **Merge engine** merges the template and the data to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output.

  Select the desired amount of DataMapper Engines.

  Adding DataMapper engines might be useful in the following circumstances:

  - When large data mapping operations have to be run simultaneously for many jobs.

- When frequently using PDF or XML based input. Particularly in the case of XML input with large individual records.
- When the *All In One* plugin is used often in Workflow configurations and there are more than two Merge engines running.

The OL Connect MariaDB database needs a fast storage system (SSD or other fast devices) to be able to keep up with two or more DataMapper engines.
When the database is installed on a system with a slow hard drive, adding a DataMapper engine may not increase the overall performance.

- **Merge Engines** selection: Select the desired amount of Merge Engines.

A **DataMapper engine** extracts data from a data file. A **Merge engine** merges the template and the data to create Email and Web output, or to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. A **Merge engine** merges the template and the data to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output.

Generally, launching a relatively high number of **Merge** engines results in better performance, as Merge engines are involved in the creation of output of all kinds (Print, Email and Web) and because content creation is relatively time-consuming.

> **Note:** The OL Connect installer automatically sets the number of Merge Engines based upon the machine CPUs/Processors (the number of cores) and available Memory (RAM). The default setting will likely be enough for most users.
>
> However, virtual machines can provide an exception to this. In some virtual environments the OL Connect installer cannot accurately determine the number of logical cores actually available to the virtual machine, and will thus use a lesser number than that which could actually be supported.

You can confirm how many Merge engines are running on your machine by running Task Manager and locating the mergeengine.exe entries in your process list.
Note that there will be an extra Merge and Weaver engine in your task manager if you are running Designer.

The Merge engine is responsible for the plugins Create Print Content, Create Email Content and Create Web Content.

License restrictions only apply to the Merge engine when creating Email or Web content. They do not apply for Create Print Content. This means you are allowed to start an infinite amount of Merge engines on any given Connect Sever to run print jobs, but you will be restricted to a set number of Merge engines for your Email and HTML jobs.

- **Weaver Engines** selection: This option is *read-only* in OL Connect Designer Preferences.
  Select the desired amount of Weaver (Output) Engines.

  A **DataMapper engine** extracts data from a data file. A **Merge engine** merges the template and the data to create Email and Web output, or to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output. A **Merge engine** merges the template and the data to create an intermediary file for Printed output. The intermediary file is in turn used by a **Weaver engine** to prepare the Print output.

  Adding extra Weaver (Output) engine(s) might be useful when large Print jobs are to be run simultaneously with smaller Print jobs. However, too many running engines will waste precious RAM and CPU cycles to idle processes, whilst too few could create a bottleneck.

Memory

This group is only available if **Engines** were selected as the processing architecture, except in the OL Connect Server Configuration tool (Client) configuration page, where it is always available.

Specify the maximum amount of random access memory (RAM) in megabytes that will be used per engine in order to make optimal use of the machine's memory.

The selections made here will apply to all Engines of that type.

Note: These settings only control the maximum size of the *Java heap* memory that an engine can use; the total amount of memory that will used by an engine is actually a bit higher.

Also keep in mind that the Connect Server and the operating system itself will need memory to keep running.

By default, each engine is set to use up to a predetermined amount of RAM. To make optimum use of a machine's capabilities it might be useful to increase the amount of memory that the various engines can use.

Here are some simple scenarios where you might want to increase an Engine's memory usage:

- DataMapper engines may perform better with greater memory when running jobs containing a lot of data.
- For complex templates with a lot of pages per document, there is a chance that Merge engines will run better with more memory.
- The maximum memory usage of a Weaver engine can be relevant for jobs with heavy graphics; or for jobs that use Cut & Stack impositioning; or for jobs using particular variables that entail page buffering (see "Content variables" on page 1662).

The options available in the **Memory** group are:

- **DataMapper Engine (MB)** selection: This selection is only available in the **Connect Server Configuration** utility. It is not available in the Connect Designer Preferences. Select the desired amount of DataMapper Engine memory.

  DataMapper engines may perform better with greater memory when running jobs containing a lot of data.
- **Merge Engines (MB)** selection: Select the desired amount of Merge Engine memory.

  For complex templates with a lot of pages per document, there is a chance that Merge engines will run better with more memory.
- **Weaver Engine (MB)** selection:
  Select the desired amount of Weaver (Output) Engine memory.

  When running really large jobs, it often pays to increase Weaver memory allocation, even if only for the duration of the production job(s).

  The maximum memory usage of a Weaver engine can be relevant for jobs with heavy graphics; or for jobs that use Cut & Stack impositioning; or for jobs using particular variables that entail page buffering (see "Content variables" on page 1662).

Multi-threading settings (Enterprise edition only)

This group allows you to set Multi-threading rasterization options for form ripping when generating AFP or IPDS output.

> **Note:** AFP and IPDS are only available in OL Connect Enterprise edition.

- **Enable multi-threaded reading** check box: This setting controls how the engine reads its input, with multi-threading reading being the default.

  > **Note:** This setting is primarily for trouble shooting. We recommended always leaving multi-threading on, unless otherwise advised by Support.

- **Number of multi-threaded form rips** selection: Set the amount of individual threads to assign to form ripping.

  This setting is useful for when graphics content triggers rasterization when writing AFP or IPDS output. Typical examples of content triggering rasterization are: transparency; fill patterns; gradient fills; or text rotated at a non-right angle.

  Increasing the number of threads should speed up Output Creation under such circumstances.

  > **Note:** How effective this setting is at speeding up the output will depend upon the number of cores available on the processing machine and how many other tasks the machine is running simultaneously.
  >
  > If only one Output Creation task is running, setting this value as equal to the number of (logical) cores will likely give the best performance.

Configure merge engine options

- On the **Merge engine tab** you can set the **Template reuse limit**. Each merge engine keeps the resources of the most recently used templates open in memory, for performance reasons. The template reuse limit determines how many times those resources get reused before they are recreated. The default value is 100. A low value is more appropriate for large jobs, while a high value is more appropriate for small jobs.

Buttons

The **Engines** preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Language preferences

- **Display language**: Select a language from the drop-down list to be used as the language of the Connect Server Configuration tool, and all log files created by OL Connect Server and its components (after the software/service is restarted).

Parallel Processing preferences

The parallel processing preferences (previously referred to as Scheduling preferences, prior to 2019.2) page provides the means to control precisely how the OL Connect and Connect Server (Master or Client, with the Enterprise version of OL Connect) handles jobs that operate in parallel.

There is considerable difference between the preferences that are available in the Designer Parallel Processing page and the Server Configuration Parallel Processing page.

- For the **Designer** specific preferences, see "Parallel Processing properties (Designer Preferences)" below.
- For the **Server Configuration** specific preferences, see "Parallel Processing properties (Server Configuration)" on page 118.

For additional information on how these preferences can enhance performance, see "Engine configuration" on page 100 and "Performance considerations" on page 33.

Parallel Processing properties (**Designer** Preferences)

Preset selection (Designer Preferences)

Only the **Custom** setting is applicable to the Designer Preferences, so this option is always selected and the field made read-only.

Content Creation Tab (Designer Preferences)

A Tab with data that relates solely to Content Creation.

The options are:

- **Total Merge engines configured** read only display: This is a read only entry that shows the total number of Merge engines available. To change this value, you must update the Merge Engines in the "Engines preferences" on page 109 page.
- **Multi tasking** group:
  When starting a new Content Creation task, the task will immediately commence if there is a Merge engine available. How many Merge engines to use is based on the number of records in the input data.

  Select from the following options:

  - **Optimize per task**: This runs each task with as many Merge engines as needed (until engines are exhausted).
    Using this option means that Merge engines will not be reassigned when new tasks come in.

    This option is better suited for **batch processing**.
  - **Maximize simultaneous tasks**: Merge engines will be reassigned from a running task to new tasks when they arrive.
    To accommodate as many tasks as possible, the server can dynamically reassign Merge engines to new tasks as they arrive. Thus a running Content Creation task need not block other tasks.
    If multiple Merge engines are processing a task, an engine can be taken from that task and reassigned to a new task.
    As each task finishes, any freed up Merge engines get re-assigned back to still running tasks, if no new tasks were waiting.

    This option is better for **on demand (ad hoc) and simultaneous job processing**.
  - **Additional engine every (records)** entry: This controls how many Merge engines are used for a Content Creation task. It means that for every additional 'x' records in the task, an additional Merge engine will be used.
    For example, with the default 100 record threshold, tasks with 1-100 records will be assigned 1 Merge engine, tasks with 101-200 get assigned 2 merge engines, tasks with 201-300 get assigned 3 merge engines, and so on.

    > **Note:** These entries aren't applied instantaneously. There is often a lag. That is why you can reserve a specific number of engines for new jobs, in the options below. Those reservations operate in real time.

    The default of 100 records was chosen purely because it is an easily multiplied

number, not because it has been proven to have any significant value. It means that on an average system (i.e., less than 10 Merge engines) any decently sized task is allowed to use all Merge engines. It also assumes that using more than one Merge engine for less than 100 records will probably not make a big enough difference to throughput speed. Obviously, there are situations where these assumptions will not apply.

> **Note:** Currently, it's only the print and PDF content creation tasks that use multiple Merge engines.

Parallel Processing properties (**Server** Configuration)

These are the options that you will be presented with if the OL Connect Server was installed.

Whether options are available for selection on this page or not is entirely dependent upon the **Number of engines** selection made in the "Engines preferences" on page 109 page. If either of the Merge Engines or Weaver Engines were set to a value beyond one, then options will become available in this Parallel Processing properties page.

Preset selection (Server Configuration)

Choose from some common usage scenarios. The preset scenarios are:

- *Default* - Basic settings that are good for running most things. Single jobs have preference over multi-tasking, however.
- *Batch Print* - Best settings for processing jobs, one by one, in a sequential, first in first out (FIFO) order.
- *On demand Print* - Best settings for processing many small print jobs simultaneously.
- *On demand* - Use when serving web pages, sending emails, and printing many on demand jobs simultaneously.
- *Connect Send* - Settings optimized for use with *Connect Send*.
  Connect Send needs a Merge engine available for on demand web pages, to be able to process on demand print jobs (especially content creation), and have a Weaver engine available for creating the production output.
- *Capture on the Go* - Settings optimized for use with *Capture on the Go* applications.
  Capture on the Go needs on demand content creation for web pages (the forms), emails (notifications), and PDFs (persistent version of forms).
- *Custom* - where you can chose exactly where and how the engines are to be assigned.

Note: We would recommend basing any Custom settings on one of the preset scenarios.
Select the preset that most closely matches your day to day needs, then tweak those settings.

Only the **Custom** setting allows you to manually set where and how the engines are to be assigned. If selected, then options for *Content Creation* and *Output Creation* will become available under the two Tabs named thus.

Content Creation Tab (Server Configuration)

A Tab with data that relates solely to Content Creation.

The options are:

- **Total Merge engines configured** read only display: This is a read only entry that shows the total number of Merge engines available. To change this value, you must update the Merge Engines in the "Engines preferences" on page 109 page.
- **Multi tasking** group:
  When starting a new Content Creation task, the task will immediately commence if there is a Merge engine available. How many Merge engines to use is based on the number of records in the input data.

  Select from the following options:

  - **Optimize per task**: This runs each task with as many Merge engines as needed (until engines are exhausted).
    Using this option means that Merge engines will not be reassigned when new tasks come in.

    This option is better suited for **batch processing**.
  - **Maximize simultaneous tasks**: Merge engines will be reassigned from a running task to new tasks when they arrive.
    To accommodate as many tasks as possible, the server can dynamically reassign Merge engines to new tasks as they arrive. Thus a running Content Creation task need not block other tasks.
    If multiple Merge engines are processing a task, an engine can be taken from that task and reassigned to a new task.
    As each task finishes, any freed up Merge engines get re-assigned back to still running tasks, if no new tasks were waiting.

This option is better for **on demand (ad hoc) and simultaneous job processing**.

- **Additional engine every (records)** entry: This controls how many Merge engines are used for a Content Creation task. It means that for every additional 'x' records in the task, an additional Merge engine will be used.
  For example, with the default 100 record threshold, tasks with 1-100 records will be assigned 1 Merge engine, tasks with 101-200 get assigned 2 merge engines, tasks with 201-300 get assigned 3 merge engines, and so on.

  > **Note:** These entries aren't applied instantaneously. There is often a lag. That is why you can reserve a specific number of engines for new jobs, in the options below. Those reservations operate in real time.

  The default of 100 records was chosen purely because it is an easily multiplied number, not because it has been proven to have any significant value. It means that on an average system (i.e., less than 10 Merge engines) any decently sized task is allowed to use all Merge engines. It also assumes that using more than one Merge engine for less than 100 records will probably not make a big enough difference to throughput speed. Obviously, there are situations where these assumptions will not apply.

  > **Note:** Currently, it's only the print and PDF content creation tasks that use multiple Merge engines.

- **Reserve engines for on demand tasks** group checkbox:
  Reassigning engines is not instantaneous when a new task arrives. To avoid inefficiencies, Merge engines will first finish work on their current selection of records, before being reassigned. Reserving engines better ensures that on demand tasks get picked up right away, but it also means that less engines will be available for large tasks.

  The total amount of Merge Engines available for selection here must be set in the "Engines preferences" on page 109 page.

  Select from the following options:

  - **Email (engines)**: Set the amount of Merge engines to reserve for Email jobs.
  - **Web (engines)**: Set the amount of Merge engines to reserve for Web based jobs.

- **Print and PDF (engines)**: Set the amount of Merge engines to reserve for Print output jobs.
- **Merge engines available for any task**: A read only value that shows how many Merge engines remain available for selection.
- **On demand task size limit (records)**: An *on demand task* is a task that has someone (or something) waiting for it to finish, so these typically need to finish "as soon as possible". Often these are single documents but not necessarily always.
  This option allows you to designate what a Content Creation *on demand task* is, based upon the number of records.

Output Creation Tab (Server Configuration)

A Tab with data that relates solely to job Output Creation.

If only the single Weaver Engine is configured in the "Engines preferences" on page 109 page, then this whole tab will be disabled.

- **Licensed speed limit (pages per minute)**: This read only entry shows the current license speed limitations, in pages per minute. The speed limitations are determined by your OL Connect license.
  This information is to help you choose what settings would make sense when assigning the "*Target speed*" values later in the Tab.
- **Licensed tasks limit**: This read only entry shows the current license task (or job) limitations.

  Note: The terms "job" and "task" can be used interchangeably.
- **Total Weaver engines configured**: This read only entry shows the total number of Weaver engines available. To change this value, you must update the amount of Weaver Engines in the "Engines preferences" on page 109 page.

Job sizes group:
These two settings allow you to define what type of jobs are to be considered **Small** jobs and what types are to be considered **Large**. Any job that falls between these two settings (if there *is* a gap between the two) will be considered a **Medium** job.

Small job max (pages): Enter the maximum number of pages a job can have and still be considered **Small**.

Large job max (pages): Enter the minimum number of pages a job must have before it is considered to be a **Large** job.

Reserved engines group:
These settings allow you to reserve some Weaver (Output) engines for small and medium Output Creation tasks. This it to prevent large jobs from using all available engines and blocking small or medium jobs from running. Since Weaver engines cannot switch tasks, such behaviour can only be achieved through reserving engines for small and/or medium jobs.

Engines may be reserved both for small, and for medium sized jobs. Engine reservations are not *required* for either though.

- **Small job (engines)**: Optionally enter the number of Weaver engines you wish to reserve for **Small** jobs.
  To make sure large batch jobs get sufficient speed during Output Creation, set a lower target speed for small jobs, this will automatically allow more for the large and medium jobs.
- **Medium job (engines)**: Optionally enter the number of Weaver engines to reserve for **Medium** jobs.
- **Total Weaver engines configured**: This read only entry shows the number of Weaver engines still available. This is the Total engine count, minus the number of engines assigned to both **Small** and **Medium** jobs.
  To change this value, you must update the total amount of Weaver Engines in the "Engines preferences" on page 109 page.

Target speed when running simultaneous jobs group:
If a single Output Creation task is running, it will be run at the full speed in the license. But when multiple tasks run in parallel, this speed has to be divided between them. By default, the maximum speed will be divided equally between tasks.

Use these settings to override what speed (in Pages Per Minute) should apply for each type of job. This allows you to prioritize one or more type of job above the others. For example, if your production process normally handles lots of small jobs, you might want to provide the smaller jobs greater speed (throughput) than the less frequent Medium and Large jobs.

But if there are so many small jobs that they start limiting the throughput of larger jobs? If there is always at least one small job running, then the maximum speed for the larger job will stay at half of the licensed limit. In this case, you might want to increase the target speed for Large jobs.

There are no hard and fast answers as to what settings will work best. It will likely be a

matter of trial and error. But many sites will not need to change speed settings at all.

The options are:

- **Small job (PPM)**: Enter the target speed for **Small** jobs, in Pages Per Minute (PPM).
- **Medium job (PPM)**: Enter the target speed for **Medium** jobs, in Pages Per Minute (PPM).
- **Large job (PPM)**: Enter the target speed for **Large** jobs, in Pages Per Minute (PPM)

The entire licensed speed limit will always be distributed among jobs when running jobs simultaneously.

After assigning a target speed, any remaining licensed speed will be distributed throughout any simultaneous jobs by a ration of the target speed.

Some general rules of thumb to apply when distributing target speed:

- Do you need to change speeds? In many cases there will likely be no need to change the target speed.
- The target speed is not a guaranteed actual speed, but a speed limit that the engine is allowed to exceed in order to utilize the licensed speed.
- When changing the target speed, don't be overly precise, you are unlikely to get that exact value anyway. It will likely be a matter of trial and error.
- As long as you don't overdo it, the actual speed limit for a task will usually be higher than the target speed.
- If there is a chance of not getting the target speed, you will see a warning in the preference page. This is just a warning, and nothing will break if you choose to ignore it.

Buttons

The Parallel Processing preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Server Clustering

Note: This option is only available in the Enterprise edition of OL Connect.

Note: This option is not available for Enterprise edition **Server Extension (Client)** installations

Server Clustering, only available in the Enterprise edition of OL Connect , enhances the processing capabilities of OL Connect Server by load-balancing jobs between the main Server module (master) and one or more Server Extension installations.

Setting up Server Clustering requires two or more installations of OL Connect on separate machines.

The Master server is setup by installing the OL Connect Server module during the "Installation Wizard" on page 42, while any Server Extension (Client) Servers are setup by installing the OL Connect Server Extension module instead.

Quick HowTo

1. Install the Master server (OL Connect Server module), making sure to select the MariaDB module.
2. Disable the **Restrict access to localhost (127.0.0.1)** option in the "Connection settings (standalone/Master)" on page 96.
3. Grant access to the MariaDB root user for the appropriate IP range on the Master server.
   See the "Binding and Root access on the Master server" on the next page section for details.
4. Restart the MariaDB Service on the Master server.
5. Install Server Extension (Client) servers (OL Connect Server Extension module).
6. Install the license on the Master server.
7. Set the preferences for the engines (see "Engine configuration" on page 100) on both the Master and Server Extensions (Clients).
8. Install the license on the Server Extension (Client) servers.
9. Restart the Master server then, once restarted, restart the Server Extension (Client) servers.

What if MariaDB is not on the Master server?

It is possible to setup clustering with a MariaDB instance that is on a Server Extension (Client) server instead of on the master. In this case, the Server Extension (Client) must be installed with the MariaDB modules, the MariaDB instance configured (steps 2-4 above) then the master and other Server Extensions can be installed. The remainder of the instructions remain the same.

It is also possible to setup clustering with MariaDB/MySQL being installed completely separately from OL Connect, such as using an existing MariaDB/MySQL instance. For such cases, the instructions for the bind address must be followed, but the user does not have to be root. A user for the database must be created and have full access (GRANT ALL PRIVILEGES) to a database called "objectiflune" that can be created before Connect is installed.

Binding and Root access on the Master server

- The MariaDB server's binding must be set to accept connections from the Server Extension (Client) servers.
  To do this, open `C:\ProgramData\Objectif Lune\OL Connect\MariaDB\my.ini` in a text editor and change the `bind-address=127.0.0.1` line to `bind-address=*`

  Once the changes have been made and saved you need to restart the MariaDB services.
- Access must be granted to the `root` user on the IPs from which the Server Extension (Client) server will connect.>
  - Open the (Windows) Command Prompt application and execute the command: `C:\Program Files\Objectif Lune\OL Connect\MariaDB\bin`

    Or open in PowerShell or Terminal. Whichever you are more comfortable with.
  - Type in the following command to connect to the database, where <password> is your MariaDB password:

    ```
    MariaDB --user=root --password=<password> olconnect
    ```

    Note: This password is the MariaDB password, entered in the OL Connect installation process. The default OL Connect MariaDB schema is `olconnect`.

    You should see the prompt become `MariaDB [olconnect]>`

- Here, type the command to allow the "root" user to be accessed from a specific IP subnet range.

  For example, to accept communication on 192.168.*.*, use:
  `GRANT ALL PRIVILEGES ON objectiflune.* TO 'root'@'192.168.0.0/255.255.0.0' IDENTIFIED BY 'password';`
  The `password` is the one provided during installation.
  (ref: http://dev.mysql.com/doc/mysql-security-excerpt/5.5/en/adding-users.html)

Tip: IP Subnets understanding is beyond the scope of this documentation. If you want to learn more, please see the Subnetwork article on Wikipedia (http://en.wikipedia.org/wiki/Subnetwork).

Clustering Preferences and Setup

When Server extensions are installed and connected to a Master, the following options and settings change in availability or behavior:

- The "Clean-up Service preferences" on page 1035 requires special configuration on Clustering setups:
  - Clean-up service should not run simultaneously on all machines (staggered clean-up).
    Doing so may cause jobs not to be processed since all servers are busy.
  - Only the machine where the MariaDB/MySQL Server product is installed should attempt to clean-up database items. Essentially servers that do not have MariaDB/MySQL should only run Orphan File Clean-up.

Known Issues

This page lists important information about issues that apply to OL Connect 2025.1.

Issues with User profile names containing Unicode characters

Connect has issues with user profile names containing Unicode characters. Particularly when locating job resources contained under the user's profile folder.

This issue has been considerably improved in 2025.1, but the following problems still persist when the working folder contains Unicode characters:

- Connect Merge Engine fails to execute JavaScript files.
- Issues converting PostScript to PDF.

- Issues with saving a PDF element in non-PDF output types.

DataMapper regular expression issue in 2025.1

An issue was identified that affects existing data mapping configurations in OL Connect 2025.1. A regular expression (such as \d\d\d\d\d\d, or \d{6} or [0-9]{6}) applied to a **Goto** step placed both outside and inside a **Repeat** step of a DataMapper configuration no longer provides the expected result. This will be fixed in a later release.

In the meantime, the workaround is to apply the following change:

1. Launch the OL Connect Designer application
2. Open the DataMapper configuration file in question
3. Go to **Settings** pane -> **Input Data** section
4. Change the **Processing method** option to *Original*
5. Save the modified DataMapper configuration file

Script API setting changes in 2025.1

In OL Connect 2025.1 stricter checking was introduced for the Uniform Resource Identifier (URI) passed into the `loadhtml`, `loadjson` and `loadtext` scripting functions.
Special characters such as spaces in a path used to work but now need to be properly URI encoded.

Example:

```
 // Worked in 2024.2 but not in 2025.1
 var result = loadhtml("file:///C:/PPWorkDir/Hello
World/Data/Example.html");
 // 2025.1 onwards
 var result = loadhtml
("file:///C:/PPWorkDir/Hello%20World/Data/Example.html");
```

This applies to more than just the space character. Refer to RFC 2396 for the rules on URI encoding.

Email setting change in 2025.1

In OL Connect 2025.1 we fixed an issue where the "*Do not inline styles*" option would still inline CSS style rules on elements. This might affect existing jobs. Particularly if you were inadvertently using the "Do not inline styles" option, but actually wanted the 'Apply CSS properties to elements' behaviour.

Dynamic email attachments issue (2024.2.1 specific)

Issues with Dynamic email attachments, which were added using custom scripts or via the Dynamic Attachment wizard.
The issue was that dynamic email attachments would accumulate across multiple messages. Additionally, when testing in the Designer application, this could cause attachments to be unintentionally added to the template.

As of 2024.2.2 we now ensure dynamic email attachments are handled correctly, with each email containing only the intended attachments and templates being saved properly.

If you are using dynamic attachments in your templates, we recommend testing them after updating to confirm that everything is functioning as expected.

Adding entries to manually added detail tables (2023.2 onwards)

The *addRow()* functionality did not work for detail tables that were manually added to the Data Model, prior to OL Connect version 2023.2. The fix for this issue allowed the use of *addRow()* for detail tables that are manually added to the data model or which have no extracted rows. As a result, any detail table without extracted data is now recognized as an existing but empty table, with a length of 0.

Remote translation files issue

Changes in a remote PO file (.rpo) are not shown in Preview mode or when double-clicking the .rpo file.
This will be addressed in a future release.

Performance not scaling as expected

When you have a machine with a lot of CPU cores and you configure Connect to run many **Merge Engines** (8+) simultaneously, the performance can be less than expected.
It has been reported that the CPU is not fully utilized and the performance per Merge Engines actually goes down, the more Merge Engines you add.

This will be addressed in a future release.

Unicode Variation Selectors issue

The Designer and Merge Engine can crash when a document or input data contains a Unicode Variation Selector. A variation selector is a special Unicode character in the range of U+FE00 - U+FE0F that indicates which variant of the previous character should be used.
The issue has so far only occurred on Windows 11 systems and will be fixed in a later release.

ODBC drivers do not all work with OL Connect

ODBC drivers do not all work with Java applications.
Since OL Connect is a Java application, it is strongly recommended you use JDBC instead.

Incorrect colors in print preview

Colors are not displayed correctly in previews generated by the Advanced Print Wizard.

Account permission issues

If you install and run OL Connect Designer and Server under different user accounts (for example, install under *User1* and run Designer as *User2*), account permission issues may occur.

It is recommended that you install and use OL Connect Designer and Server under the same user account.

Issues converting some images from PCL to PDF

Problems have been reported with the conversion of some images, from PCL files to PDF output.
In some, the image is replaced with a grey box. In others, the images convert, but with a vertical line added to the edge of the image.

This is due to an issue with a third party library which will be addressed in a future release.

Incorrect warning when sending email from Designer

When sending email from the Designer with a license that allows this, an incorrect warning is displayed stating that sending email is not covered by the license. The warning does not prevent the emails from being sent.

Master Page not affected by style rules for <body>

In the output, content in Master Pages is affected by style rules targeting <html>, but not by style rules targeting the <body> element. This is because in the output, they are placed outside the <body> of the section to which they are applied. When editing a Master Page, style rules targeting the <body> element will work, because the Master Page is not currently applied to a section, but they will have no effect in the output.

JDBC connection issue

In OL Connect version 2022.2, the SQL Server driver has been updated to version 10.2.0.jre11. As opposed to preceding versions, this driver by default attempts to connect to a JDBC database with encryption enabled if the *encrypt* parameter is missing from the connection string. This might break existing JDBC connections, particularly those defined in scripts and in the Database Wizard's 'Advanced Mode'.

To instruct this driver to not use encryption, the "`;encrypt=false`" parameter needs to be present in the connection string.

Installer issues

The new 2022.1 installer has some minor issues that will be fixed in a subsequent release. The issues are:

- After installation, the "*recent files*" list is cleared and the measurement units are reset from 'cm' to 'inch'.
- When updating from earlier 2022.1.x versions the bundled MariaDB connection settings are reset.

Dynamic tables causing issues

In some specific circumstances the dynamic tables feature does not behave correctly. The following scenarios should be avoided to prevent subsequent issues with the template:

- Using `page-break-before:always` on the top TR of a TBODY causes a problem and results in a pagination error
- It is possible to specify conflicting attributes on a TR element:
  - `page-break-before:always` or `page-break-after:always` combined with data-repeat
  - `page-break-after:always` combined with `page-break-before:avoid` on the next sibling TR (or similar combinations)

These issues will be addressed in a later OL Connect release.

DataMapper: Automatic Date/Time does not work with certain negative UTC time zone offsets

The new "*Automatic*" date parsing option in the DataMapper cannot parse dates with negative UTC time zone offsets of non-zero minutes.

For example: *2021-03-01T10:00-03:**00*** will work, but *2021-03-01T10:00-03:**30*** will not.

Positive time zones with non-zero minutes (such as *+10:30*) are unaffected.

The negative time zone offsets affected are: **UTC -02:30**; **-03:30**; **-04:30** and **-09:30**

This issue will be fixed in a later release. In the meantime, the workaround is to use "*Custom*" or "*ISO8601*" date parsing options in DataMapper.

CSS inlining colour values now converted to RGB

As of OL Connect 2021.2 when using the CSS inlining mode "*Apply CSS properties on elements*" for emails, all colour values are now converted to RGB, rather than to HEX.

Issues running Connect on Hyper-V 9.0

Some customers have reported difficulties running OL Connect on Hyper-V version 9.0. In some instances OL Connect cannot install and in others the OL Connect Server service sometime stops with a signature error.
To resolve these issue we recommend downgrading to Hyper-V version 8.0 where these issues are not reported.

Minor font changes

As of Connect 2021.1 we no longer round fonts to pixel size. This can lead to tiny differences in the output (of 1-2 pixels) when compared to earlier versions.

Changed Omit Master Page Back behaviour

In versions of Connect prior to 2020.2, if a page had no content except for a linked DataMapper background, then Connect would consider it an "empty" page when determining whether or not to "*Omit Master Page Back in case of an empty back page*" (available as an option in the sheet configuration of a section). This has now been fixed in Connect 2020.2, and such pages are no longer considered empty.

This could impact on the output from existing templates.

Print Wizard Preview showing some Landscape mode jobs in Portrait mode

Print jobs featuring associated Connect "Output Presets" on page 1648 which have "Imposition options" on page 1456 set to Landscape will not **Preview** in Landscape, but rather in Portrait mode. The printout is unaffected, however.

Issues associating PDF files with Connect

Under certain circumstances, Connect Setups prior to 2019.2 would fail when attempting to add the "Enhance with Connect" association with PDF files. This would then cause the setup to appear to fail.

Whilst this issue has been fixed in the Connect 2019.2 installer, if a user had previously experienced the issue and temporarily worked around it to complete the installation, then the Connect installer will fail on upgrade or uninstallation.

To get around this, a manual uninstall is required, or a modification to registry entries.

Issues when loading some Workflow plugins for the first time

Under some circumstances, certain of the new Workflow plugins introduced with Connect 2019.2 will fail to load correctly when run for the first time. Instead of opening the plugin dialog as expected, the plugin hangs and displays a message that it is "Loading UI .... ". This is due to an issue with a third party library which will be addressed in a future release.

To get around the problem, please close and reopen the plugin. The problem only occurs on the initial opening, and should work fine thereafter.

The license update introduced in OL Connect 2019.1 does not cater for existing AFP input licenses

AFP Input is an add on option for OL Connect licenses. Unfortunately, the update to the 2019.1 version of the OL Connect license does not cater for existing AFP input licenses.

If you have an existing AFP input license we ask that you contact your local Customer Care team after the initial license update is complete and have them add the AFP input option back into your license. (See (https://www.objectiflune.com/WebActivationManager/CareInfo.aspx.)

The Connect Print Manager is a standalone Connect tool that is used in production sites to manage print jobs. It can be installed and run on standalone machines without a Connect license.

The Update Client for Connect 2019.1 looks for a Connect license as part of the update process, but it will not find one on standalone Connect Print Manager installations. As a result the Update Client will not recognize that there is a Connect update available to the Print Manager machine.

In order to update Connect Print Manager to version 2019.1 you will need to download the Connect 2019.1 installer outside of the Update Client. The Connect 2019.1 installer can be downloaded from the Web Activation Manager site. See https://www.objectiflune.com/webactivationmanager/. Or you could ask your local Administration for the installer, as it would likely already have been downloaded for installation by the document designers.

Page break changes in 2019.1

Improved page breaking in Connect 2019.1 might impact upon some existing templates. It is recommended that you check page breaking in existing jobs, where page breaks at a specific location are a known criteria.

Minor differences in AFP and IPDS output introduced in 2019.1

Connect 2019.1 now defaults to "*Scale to Fit*" for both **IPDS** and **AFP** output.
Please note that this may have an impact when printing pre-existing Connect jobs (made in Connect versions predating 2019.1) on older printer models that do not support "*Scale to Fit*".

Issue with image placement in 2019.1 when using some customized AFP and IPDS Printer Definitions

An issue with image placement in Connect 2019.1 AFP and IPDS output was discovered just prior to the release. The issue is specific to AFP and IPDS 600 DPI Printer Definitions that are not set to "*Scale to Fit*".

As "*Scale to Fit*" is now switched on by default in all standard Connect 2019.1 AFP and IPDS Printer Definitions, none of those should encounter the issue.

This issue will be fixed in a subsequent patch release.

Issue after erroneous or incomplete update or re-installation

If one or more products (the OL Connect Designer, Connect Server, Software Activation, Print Manager, or Server Configuration Tool) or engines exit within a second of starting, this may be caused by a recent erroneous or incomplete uninstall before a reinstall or upgrade to a newer version of OL Connect.

This may be solved by deleting the `%UserProfile%\Connect\.eclipse` directory.

Backend database might require periodic maintenance

Databases maintain a variety of statistics in order to optimize performance. When high levels of inserts and/or deletions occur, the statistical data keeping can struggle to keep up. Over a period of prolonged and intensive processing this can result in a degradation in performance, with the whole database slowing down as it struggles to clean itself up.

In Connect terms the effect can be felt as the Data Mapper and/or Job Creation progressively slowing down.

To cure this issue, it is recommended that you periodically run manual maintenance on the backend database.

If using **MySQL**, the following script should be run in a query window:

```
set @a=null,@c=null,@b=concat("show tables where",
ifnull(concat(" `Tables_in_",database(),"` like '",@c,"' and"),''),
" (@a:=concat_ws(',',@a,`Tables_in_",database(),"`))");

Prepare `bd` from @b;
EXECUTE `bd`;
DEALLOCATE PREPARE `bd`;

set @a:=concat('optimize table ',@a);
PREPARE `sql` FROM @a;
EXECUTE `sql`;
DEALLOCATE PREPARE `sql`;

set @a=null,@b=null,@c=null;
```

If using **Microsoft SQL Server** run the following command in a query window:

```
sp_updatestats
```

Windows 10 Search service impacting Connect

The Windows 10 Search service runs as a background task, indexing files and folders. It has been noted that this background task is sometimes preventing files being added to the Connect temporary files folder when large amounts of files are being output and copied.

If this is an issue for you, we suggest disabling Search Indexing on the *C:\Users\<username>\Connect* folder.

This issue will be fixed in a later release.

Job Presets: External Sorting change introduced in 2018.2

Versions prior to 2018.2 did not correctly save the line end characters for external sort configurations in Job Presets, which meant the job could not be externally sorted. This issue has been fixed in version 2018.2. However, Job Presets created with an earlier version may still have the wrong line end character for external sorting. To fix this, open the Job Preset in the new version, reset the line end setting in the sorting options and then save the preset.

Engine Preferences: Backward Compatibility Issues introduced in 2018.2

- Prior to version 2018.2 Connect allowed a mixture of internal and external engines. As of OL Connect 2018.2 this is no longer allowed.

  When upgrading to OL Connect 2018.2 from such installations, the pre-existing settings will not only no longer apply, but can cause scheduling preference conflicts for the Merge and Weaver engines.

  To fix this, any pre-existing Connect installation that was running a mixture of internal and external Merge and Weaver Engines must first restore their scheduling preferences to the default values. This can be done by clicking on the **Restore Defaults** button in the *Scheduling* pages of the Server Preference or the Designer Preference dialogs.
- The Designer's scheduling settings are only updated correctly for the user actually performing the update.

Business Graphics: Backward Compatibility Issues introduced in 2018.1

As a consequence of changes in both the user interface and the underlying technology, Business Graphics made with a version prior to OL Connect 2018.1 may not display correctly when opened in version 2025.1.

The currently known backward compatibility issues are listed here:

All charts

- **Legend position**: The position of the legend is not converted. It defaults to 'left' in a converted chart.
- **Rows are series/Columns are series**: Only one type of data structure for detail tables is supported in the new version: the one that corresponds to the former *Columns are series* setting (with which charts display one series per record, one bar/point per data field). After conversion to 2018.1, charts that used the *Rows are series* setting will be displayed as if the *Columns are series* setting were used. Pie charts will thus only show data from the first record in the detail table. If the number of records in the detail table remains consistent, then the charts can be corrected by modifying the data mapping configuration (see "Preparing a data table" on page 831 in the Online Help: https://help.uplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html). Otherwise the data needs to be transposed via script.
- **NOTE:** Expanded custom chart scripts cannot be converted.

Pie charts

- **Default colors**: The default colors (used when no pie chart colors are specified) have changed.

Line and Bar charts

- **Legend label**: In previous versions, the name for a values series (needed for the legend) could only be taken from a field outside the detail table. The value of the selected field would be used. Setting a different label required expanding the chart script. After conversion, the name of the field is used instead of the value. Setting a different label can now be done in the Edit Chart Script dialog (see "Selecting data for a Business Graphic" on page 830 in the Online Help: https://help.objectiflandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html).

Known Font issues

The following font(s) are known to have issues in OL Connect 2025.1:

- **Benton Sans CFF** font

Minor differences in PCL, AFP and IPDS output introduced in 2018.1

The browser component (Mozilla Gecko) used in the WYSIWYG editor of the Designer was updated for Connect 2018.1. This allows use of new CSS properties, such as flexbox.

However this update could lead to increased output file sizes for some PCL, AFP and IPDS jobs. This is generally not a cause for concern, however there might be some associated increase in processing times, as well as some minor differences in the output. For example, table line widths and font spacings might differ slightly (particularly for SMALL CAPS text), which could lead to slightly different word-wrapping in some circumstances.

Windows Server 2016 issue

As of OL Connect 2018.1 Connect is officially supported under Windows Server 2016.

Please note, however, that the **Update Client** application might be blocked by the enhanced security settings in Windows Server 2016.

To fix this, add http://updates.ca.objectiflune.com to the list of trusted web sites on that machine, or lower the internet access rules.

Limit of 100MB of image files within a single job

The browser component (Mozilla Gecko) used in the WYSIWYG editor of the Designer was updated for Connect 2018.1. This allows use of new CSS properties, such as flexbox.

However this update also introduced a limit of 100MBs for image files included within a single job. The limit is set at 100MB deliberately, as this allows *most* jobs to run faster. However, if a job requires more than 100MBs of image files, then the Connect image cache size can be increased to cater for such.

Please contact **OL Support** for instructions on how to modify the image memory cache value, if needed.

Print Output: Booklet Impositioning changes introduced in 2018.1

When Booklet Impositioning is enabled, all pages within a document need to be changed to duplex prior to Impositioning . The method for duplexing jobs has been changed to now always combine existing pages into the front and backsides of sheets, rather than adding empty backsides to any simplex pages.

The result is that now every document in the job becomes a booklet without any empty pages between the first page and the last page.

With some exceptions. Booklet Impositionings that require a multiple of 4 pages (*Saddle binding* and *Perfect binding*) will still get empty pages added, when needed.

Memory usage in clustered environments

If Database Petitioning is not selected, then OL Connect can fill the database faster than the Clean-up service can clear it (the ratio is approximately 3:1), in high speed clustered environments. Therefore if the system needs to run for some time under a considerable load, a larger database should be used to prevent production from being interrupted. Or set the clean-up service to use Database Partitioning.

We also recommend only the Master Server performing the database clean-up, or the local Server Extension if the database is not installed with the Master, to maximize performance. For **all** Designers, Server and Server Extensions except one, set the clean-up of the database entities to run every 100 months and disable the clean-up for any Designer which also has a Server or Server Extension on that machine.
Note that if the Designer is a stand-alone then it is not connected to the central database for clean-up purposes.

The Database Petitioning option can be found in the "Clean-up Service preferences" on page 1035, see https://help.uplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html in the Online Help.

Issues with Microsoft Edge browser

The Microsoft Edge browser fails to display web pages when the Workflow's CORS option (in the HTTP Server Input 2 section) is set to "*". This issue will be resolved in a future release.

Installation paths with multi-byte characters

When installing the Chinese (Traditional or Simplified) or Japanese versions of Connect, if the user specifies an alternative installation path containing multi-byte/wide-char characters it can break some of the links to the Connect-related shortcuts in the Start Menu and cause an error to appear at the end of the installer. The workaround for the moment is to use the default installation path. The problem will be addressed in a later release.

Switching languages

Changing the language using the **Window > Preferences > Language Setting** menu option does not currently change all of the strings in the application to the selected language. This is a known issue and will be fixed in a later release.

In the meantime we offer the following workaround for anyone who needs to change the language:

1. Go to the .ini files for the Designer and Server Config:
    - `C:\Program Files\Objectif Lune\OL Connect\Connect Designer\Designer.ini`
    - `C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.ini`
2. Change the language parameter to the required one under Duser.language=en | es | de | fr | it | ja | ko | pt | tw | zh

Only one of the above language tags should be selected. Once saved, Connect will appear in the selected language at next start-up.

GoDaddy certificates

When installing Connect offline, dialogs allow installing the GoDaddy certificates. Most users should use the default settings and click **Next**. In some cases, however, this may not work correctly. For this reason those users should activate **Place all certificates in the following store** and then select the **Trusted Root Certification Authorities** as the target certificate store.

MySQL Compatibility

The minimum supported MySQL version is MySQL 5.6.

PostScript print presets

The print presets for PostScript were changed from Version 1.1 onwards meaning that some presets created in Version 1.0 or 1.0.1 may no longer work.

Any PostScript print preset from Version 1.0 that contains the following will not work in Version 2025.1: *.all[0].*

Any preset containing this code will need to be recreated in Version 2025.1.

Available Printer Models

Note that only the single Printer Model (Generic PDF) will appear on the **Advanced** page of the **Print Wizard** by default.

To add additional printer models click on the settings button next to the Model selection entry box.

External resources in Connect

There are certain limitations on how external resources can be used in Connect. For example if you want to link a file (e.g., CSS, image, JavaScript etc.) from a location on the network but you do not want to have a copy of the file saved with the template you need to do the following:

1. The resource must be located where it can be accessed by all Servers/Clients run as users. Failure to do this will cause the image to appear as a Red X in the output for all documents which were merged by engines which could not access the file. The job will terminate normally and the error will be logged.

2. The file must be referenced via a UNC path e.g.,

   file:////w2k8r2envan/z%20images/Picture/Supported/JPG/AB004763.jpg

   - UNC paths are required because the services will be unable to access mapped network drives (Windows security feature).
   - The engine processing the job will look on the local file system for the direct file path leading to the “resource not found” issue mentioned above.

Caution: The Designer itself and Proof Print do not use processes that run as services and they may find local files with non-UNC paths which can lead to the false impression that the resources are correct.

Using Capture after installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, Capture can no longer be used within Workflow 7. The plugins are now registered uniquely to Workflow 8 and the Messenger for Workflow 7 is taken offline. It is only possible to use Capture from OL Connect Workflow 8 thereafter.

Capturing spool files after installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, the PlanetPress Suite 7 option to capture spool files from printer queues will no longer function. The solution is to use OL Connect Workflow 8 to capture spool files from printer queues.

Color Model in Style Sheets

The color model of colors defined in a style sheet can sometimes change after editing the style sheet. This is a known issue and will be addressed in a subsequent release.

Image preview in Designer

If in the Windows Internet settings (**Connection Settings > LAN configuration**) a proxy is enabled, but "Bypass proxy settings for local addresses" is not checked, the image preview service, conversion service and live preview tab in the Designer will not work and exhibit the following issues:

- Images will be shown as 0 size boxes (no red 'X' is displayed).
- Live preview does not progress, and when re-activated reports "browsers is busy".

To fix the issue you must check the "Bypass proxy settings for local addresses" option.

Merge/Weaver engines when printing

The print operation in the Designer will automatically detect whether the Merge\Weaver engines are available and display a message for the user to retry or cancel if not. Once the Merge/Weaver engine becomes available and the user presses retry the print operation will proceed as normal. This message can also occur in the following circumstances:

- If the server is offline and you are not using Proof Print.
- On some occasions before the Print Wizard opens.

REST Calls for Remote Services

The Server will accept REST calls for all remote services and will make commands wait indefinitely until the required engines become available. The Server will log when it is waiting for an engine and when it becomes available. Note that there is no way to cancel any commands other than stopping the Server.

Print Content and Email Content in OL Connect Workflow

In OL Connect Workflow's Print Content and Email Content tasks, the option to Update Records from Metadata will only work for fields whose data type is set to String in the data model. Fields of other types will not be updated in the database and no error will be raised. This will be fixed in a later release.

Print Limitations when the Output Server is located on a different machine

The following limitation may occur when using the Print options from a Designer located on a different machine to the Output Server:

- The file path for the prompt and directory output modes is evaluated on both the client AND server side. When printing to a network share it must be available to BOTH the Designer and Server for the job to terminate successfully.
- The Windows printer must be installed on both the Server and Designer machines.

- When printing via the Server from a remote Designer, the output file remains on the Server machine. This is remedied by selecting “Output Local” in the Output Creation configuration.

VIPP Output

Some templates set up with landscape orientation are being produced as portrait in VIPP. It can also sometimes be the case that text and images can be slightly displaced. These are known issues and will be addressed in a later release of Connect.

Magic Number changes when installing Docker

Installing Docker on a system where Connect has already been installed may impact Connect's licensing mechanism and require reactivation.

> **Note:** Installing Connect after Docker has already been installed will not cause issues.

Uninstalling

This topic provides some important information about uninstalling (removing) OL Connect 2025.1.

To uninstall OL Connect select the application from within the Add/Remove programs option under the Control Panel. This will start the **OL Connect Setup Wizard** in uninstall mode.

> **Note:** The **OL Connect Setup Wizard** might take some seconds to appear.

Important: Stop any active Anti-Virus software before uninstalling Connect back-end database.

Some anti-virus systems are known to block the uninstallation of MariaDB datafiles, as well as blocking the uninstallation of the MariaDB database application itself. If you wish to uninstall the Connect back-end database it is **highly recommended** that any anti-virus application be stopped prior to uninstalling OL Connect , as otherwise the Connect uninstallation might not work correctly.

Impacts upon other applications and services

- The Uninstall will terminate the installed Server / MariaDB service(s).
- The following applications / services should be stopped in a controlled fashion, before running the OL Connect Uninstall:
    1. OL Connect
    2. Connect products on remote systems which refer to this MariaDB database.
    3. Any Connect Workflow using OL Connect plugins which connect to this server.
    4. OL Connect Server Extensions (Enterprise edition only) on remote systems which connect to this machine as the Master Server.

Uninstallation Wizard

The uninstallation is done by running the OL Connect Setup Wizard in uninstall mode. The Wizard contains the following important pages:

1. **OL Connect Setup**: This page allows selection of what is to be done. An modification to the existing installation (**Add or Remove Features**) or full Uninstall.
   NOTE: If the Uninstall option is selected the **Remove user data** option is made available. For information about exactly what data would be saved or deleted, please see "Pre-existing User Data" on page 78.
2. **Component Selection**: If **Add or Remove Features** was selected on previous screen, this page provides options for adding or removing Connect features.
   For detail descriptions of the options, see Installation "Component Selection" on page 43 .

Walkthroughs

Learn to work with OL Connect at your own pace with these step-by-step walkthrough guides.

The walkthroughs are best followed in this order:

- "Creating an XML data mapping configuration" below
- "Designing a letter template" on page 196
- "Designing an invoice template" on page 160
- "Building an automated print process" on page 220 in OL Connect Workflow with print presets created in Designer

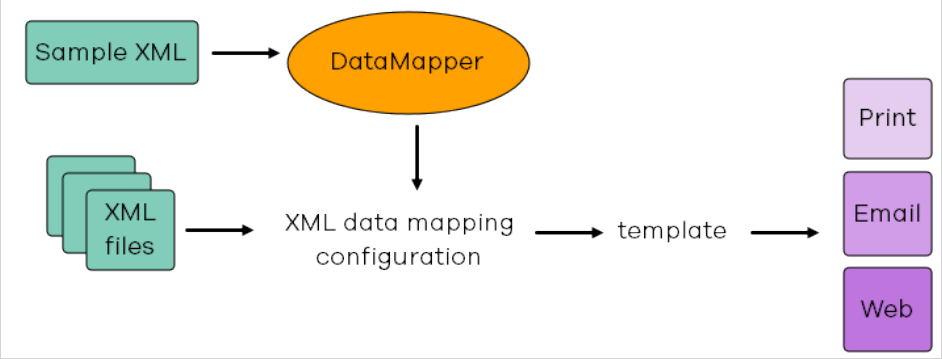
They explain common techniques applicable to other types of data and documents.

If you would like additional training, please contact your reseller or Support.

Creating an XML data mapping configuration

With OL Connect Designer you can create data mapping configurations to extract data from a variety of files. Data mapping configurations can then be used to add variable data to OL Connect Designer templates.

This walkthrough guides you through the process of creating a data mapping configuration for an XML file. This data mapping configuration can extract data from all XML files that have the same structure.



This tutorial introduces you to the DataMapper which is part of the Designer. You will learn to create a data mapping configuration by opening a sample file and defining records in it. Then you will extract the data by adding extraction steps and detail tables to the configuration. You will also learn how to rename fields and detail tables, and how to fine-tune the data for use in the Designer.

After completing this walkthrough, you'll have a basic knowledge of how to create a data mapping configuration for XML files. However, this walkthrough describes only one way to

do things. It shows how to extract data using toolbar buttons, for example, whereas it would also be possible to do that via drag-and-drop, the menu, or the shortcut menu.

So, go on and explore more! To discover new ways and enhance your skills, please visit the OL Resource Center and browse the DataMapper Help.

Note: A PDF version of this walkthrough is available for offline use: DataMapper XML Walkthrough PDF.

Note: The resources needed for this walkthrough can be downloaded: olsg-data.zip

Creating a data mapping configuration

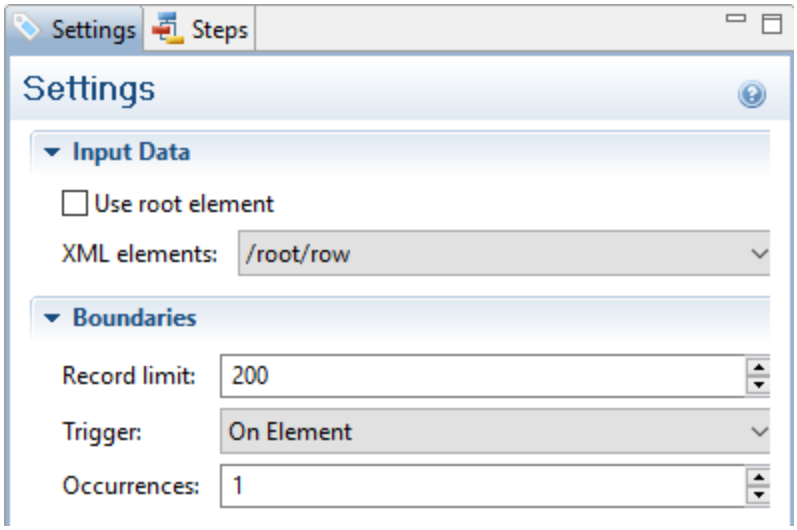
The first step towards a Data Mapping Configuration is to open a source file and help the DataMapper to identify records. In this exercise you will open an XML file and set the record boundaries.

1. Open the XML file

1. Start OL Connect Designer and select **File > New** from the menu.
2. Expand **Data mapping configuration**, then expand **Files**.
3. Click **XML file**.
   There is also a Wizard for XML files. Save trying that for later; it automates a number of the steps that this Walkthrough demonstrates.
4. Click **Next** and select the sample file: **olsg-data.XML**.
5. Click **Finish**.

2. Identify records

Take a look at the **Settings** pane on the left.



The **Input Data** and **Boundaries** settings determine what identifies a record in the source file.

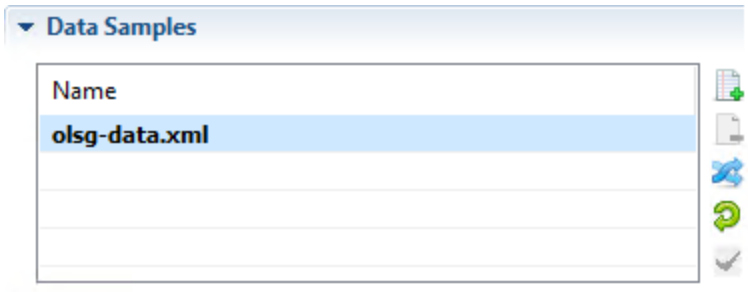
For this file, the correct settings are:

- **XML elements**: /root/row.
- **Trigger**: On element
- **Occurrences**: 1

This means that every time a new <row> element occurs, the DataMapper is triggered to start a new record. Setting **Occurrences** to **2** would make two <row> elements go into one record.

The **Record Limit** limits the number of records that can be browsed in the **Data model** pane.

> **Note:** The Record Limit does not limit the amount of records that can be extracted from an XML file using this data mapping configuration in OL Connect Workflow.

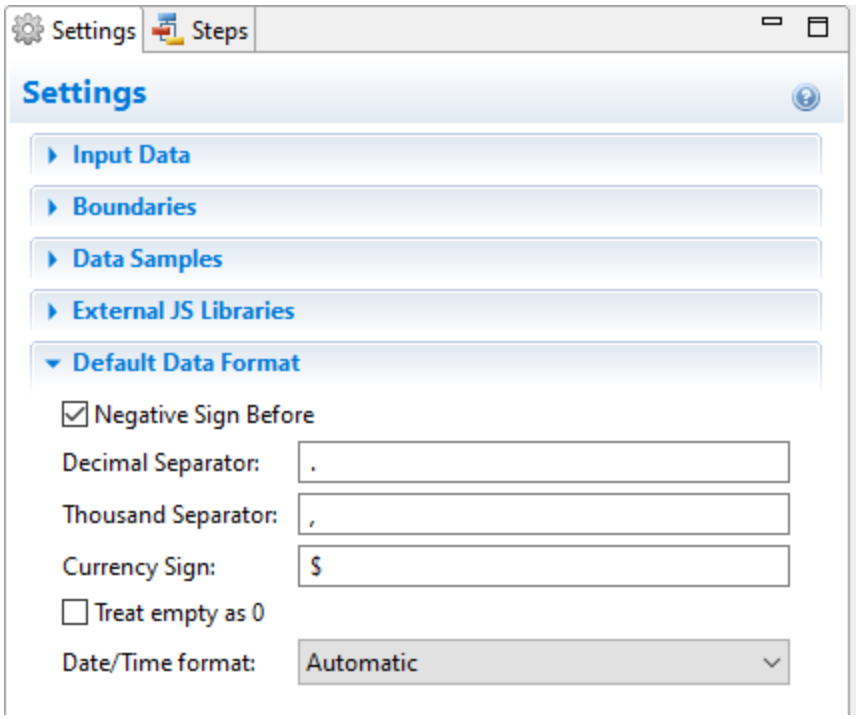


The XML file has been added to the Data Samples. Via the menu **File > Add data** you could add more samples if you'd need to. For this tutorial one sample is sufficient.

3. Set the default data format

By setting the default data format you are telling the DataMapper what format it can expect when it has to read a field as a date or a number from the source file.

1. Below **Data Samples** on the **Settings** pane, there is a heading **Default Data Format**. Click this heading.



   Note: The initial settings for the Default Data Format can be set via the menu: **Window > Preferences > Datamapper > Default format**.

2. Both separators look fine. Leave them as they are.
3. In **olsg-data.XML**, amounts of money do not have a currency sign. Remove the currency sign.
4. Next, we can set the default date/time format. The dates in the source data don't look the same. <Date> has the format day/month/year (or rather, dd/mm/yyyy), while <DueDate> looks like this: yyyy-mm-dd. Leave the setting to **Automatic**, to let the DataMapper try and parse them automatically.
   The other options can come in handy, for example when dates in a single source file belong to different time zones, or have different languages. See the Settings help page for an explanation of those options.

4. Save the file

The DataMapper does not automatically save the configuration, so now that the initial settings are done, it's a good idea to save the file.

1. Select **File > Save as**, and give the data mapping configuration a name.
2. Press **Ctrl** + **S** every now and then to save the file while working on the data mapping configuration.

> **Tip:** In the Save preferences (**Window > Preferences > Save**) you can configure auto-backup and auto-save.

Extracting data

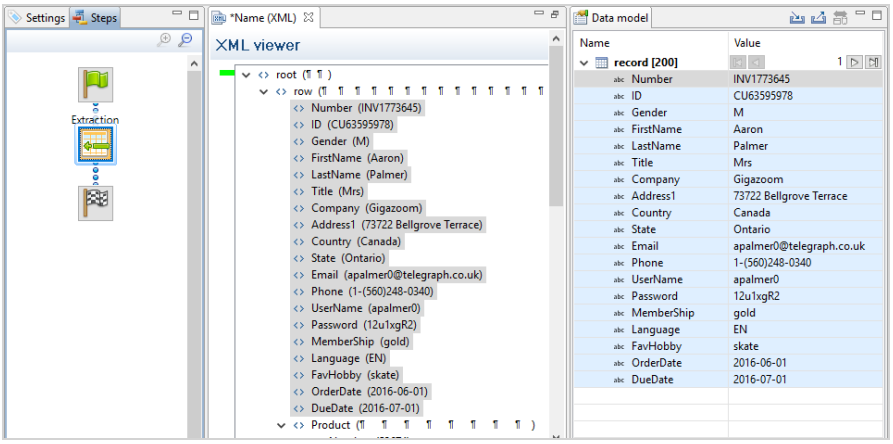
The next step is to extract data. Open the **Steps** pane, next to the **Settings** tab. Here you can see two flags, representing the Preprocessor step and the Postprocessor step. With these you could, among other things, define variables and modify the source file before extracting data (Preprocessor step) or modify the output file after extracting data (Postprocessor step). But in this case you can immediately start extracting data.

1. Create an Extract step

At the heart of this data mapping configuration are an **Extract** step and a **Repeat** step. First you'll add the **Extract** step.

1. In the first row element, select the elements from **Number** up to and including **DueDate**. (E.g. Hold the **Shift** key, click Number and then click DueDate.)
2. Click the **Add Extract Step** button . This adds an **Extraction step** to the **Steps** pane, between the Preprocessor step and the Postprocessor step.
3. Take a look at the **Data model** pane at the right. You can now browse the resulting

records.



2. Add extra fields to an Extraction step

At the bottom of each row element, below the products, there's some more information that needs to be extracted. You could create a new Extraction step for that, but it's better to add it to the existing Extraction step: the fewer Extraction steps, the faster the data mapping will be.

1. Click the **Extraction** step on the **Steps** pane to select it.
2. Click in the **XML viewer** (in the middle) and scroll down past all <product> elements. Select the elements **InvSubTotal**, **InvTaxTotal** and **InvTotal**. (Hold the **Ctrl** key while clicking the fields.)
3. Click the **Add Extract Field** button to add the fields to the existing Extraction step, instead of creating a new Extraction step.

Tip: To create the fastest possible data mapping configuration, add as few Extraction steps as possible.

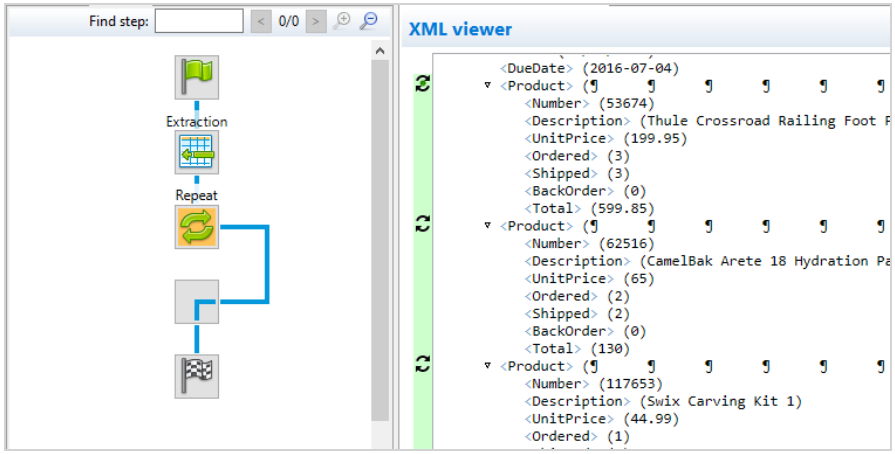
3. Extract line items using a Repeat step

When the number of a certain type of element in a record can vary, like the number of products in a row in this XML file, they have to go in a **detail table**.

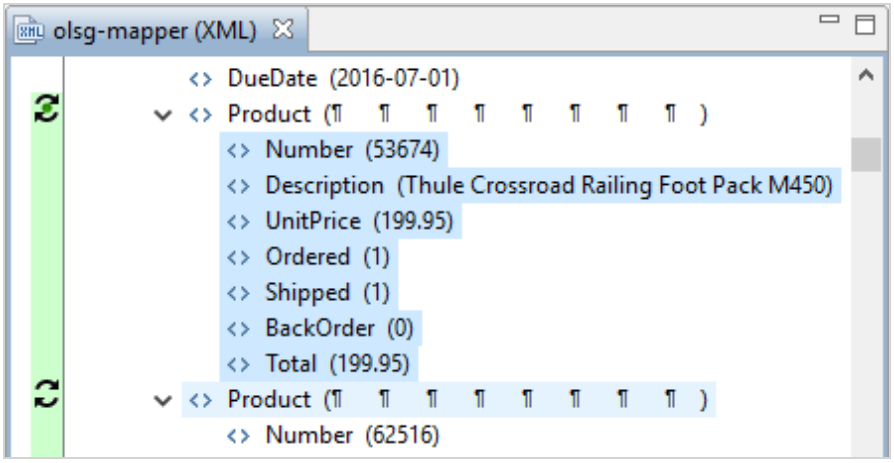
1. The cursor in the data file should be located at the first of the repeated elements. Click the first **Product** element.
2. Click the **Add Repeat Step** button. In the left margin of the XML viewer you will see a green line. If you don't, there is probably only one product in the current record; browse some more records via the Data model pane. Also check that the path in the **Collection** field on the **Step Properties** pane (below the XML) is *./row/Product*, which

means: every Product in every row element. The Collection field determines what the loop will go through .

The **Add Repeat Step** icon appears before each product.

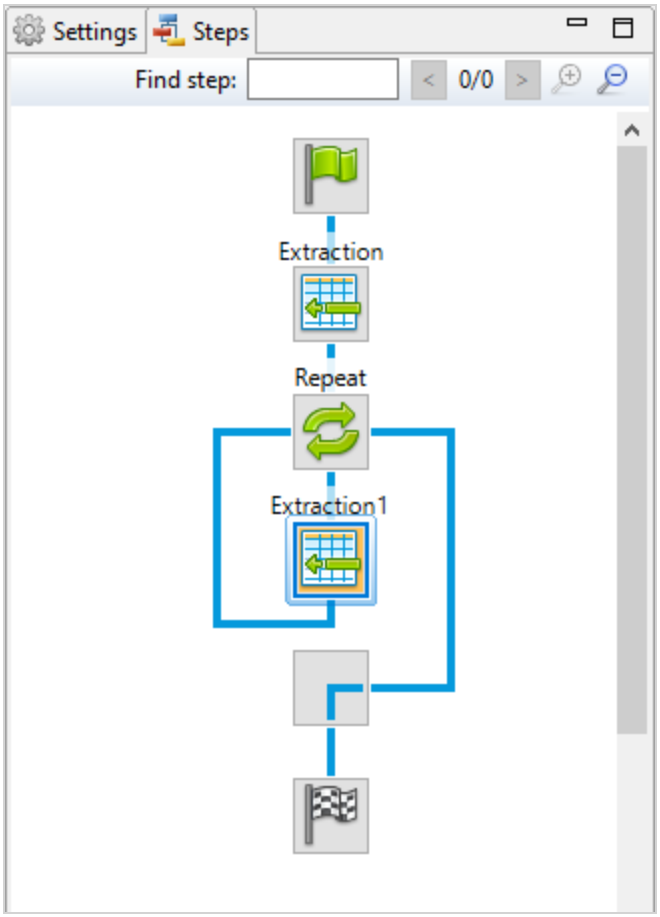


3. Select all elements inside the first Product element: click **Number**, and then, while pressing the **Shift** key, click **Total**.

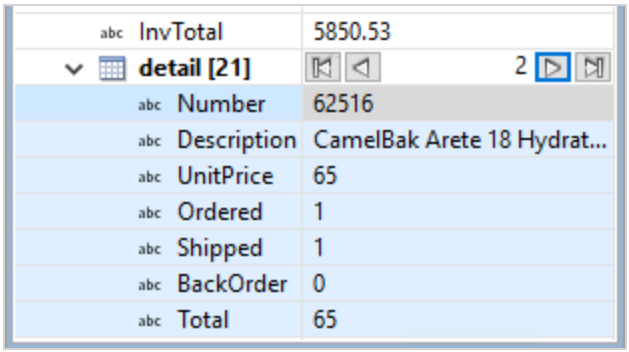


4. Make sure the Repeat step is selected on the Steps pane. Then click the **Add Extract Step** button .

On the **Steps** pane, a new **Extraction** step has appeared within the **Repeat** step.



5. A detail table has been added on the **Data Model** pane. Try browsing the products that belong to one record.



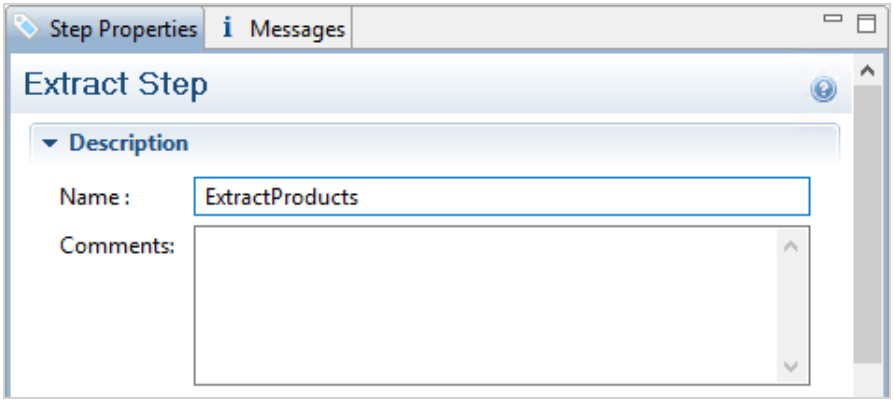
Renaming steps, fields and detail tables

1. Rename extraction steps

With only two extraction steps, this data mapping configuration is fairly simple. Extraction steps in bigger and more complex data mapping configuration need clear names to help

you keep track of what each step does. This is how to rename **Extract** steps.

1. On the **Steps** pane, click the **Extraction1** step inside the **Repeat** step.
2. On the **Step Properties** pane (below the XML), click **Description** to expand it, and rename the Repeat step, for example to **ExtractProducts**.

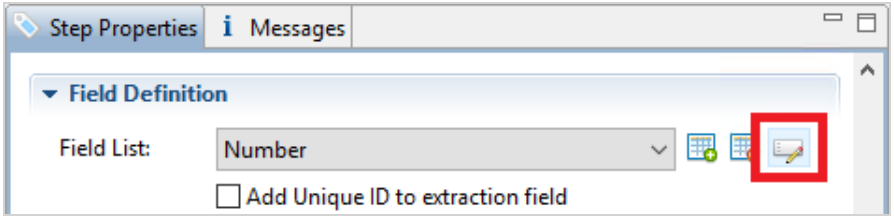


2. Rename fields

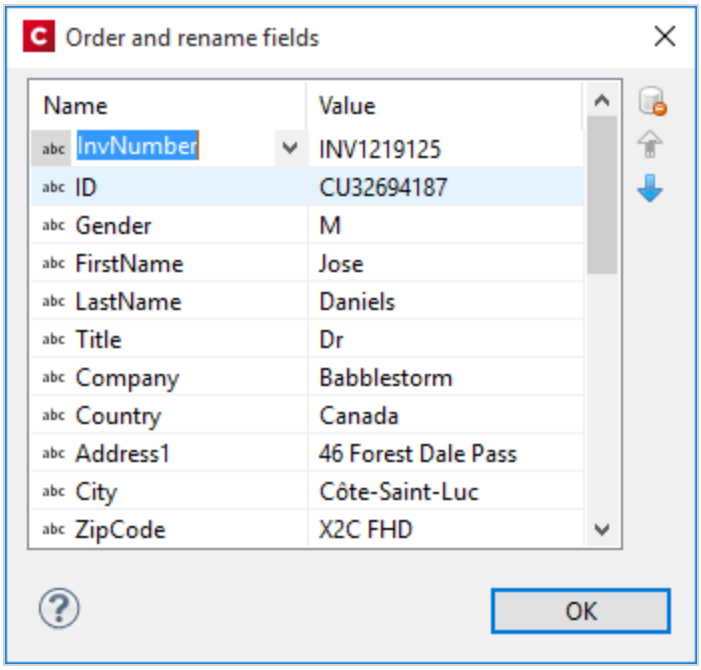
The field names in the **Data model** pane will also be visible in the Designer. Field names like Number and Number2 can be confusing when you are creating a template with variable data. In this exercise you will rename those fields and one of the fields in the detail table.

The first Number field actually contains an invoice number. Rename the field so that the field name makes this clear:

1. On the **Data model** pane, click the **Number** field (or any other field in the same **Extract** step).
2. On the **Step Properties** pane, click the **Order and Rename Fields** button .



3. In the dialog that appears, click the field **Number** and rename it to **InvNumber**.



There is an easier way to do this. We'll use that to rename the Number2 field in the detail table to make clear that that field contains a product number:

1. On the **Data model** pane, **right-click** the **Number** field in the detail table.
2. Select **Rename**.
3. Rename the field **Number** to **ProdNumber**.

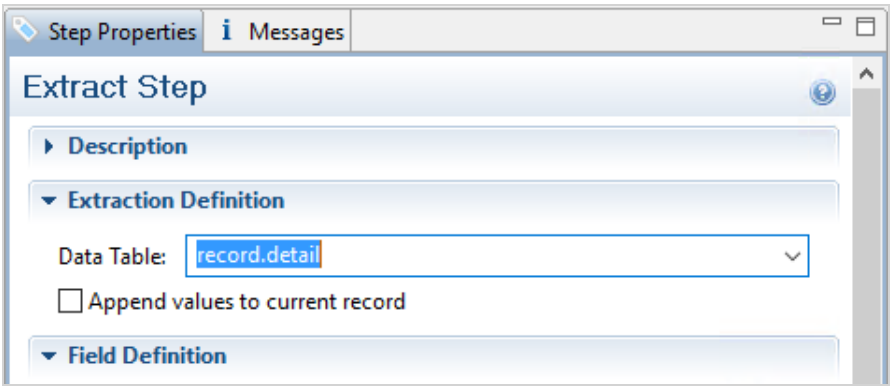
3. Rename a detail table

Renaming detail tables is especially useful when there are more detail tables in one record, or when a detail table contains another detail table. For this detail table, 'products' would be a better name.

This must be done on the Step Properties pane.

1. On the **Data model** pane, click one of the fields in the detail table.

2. On the **Step Properties** pane, click **Extraction Definition**.



3. Rename the detail table from **record.detail** to **record.products**.

> **Note:** Note: A detail table's name should always begin with 'record.'.

4. Click somewhere else on the **Step Properties** pane to update the data model. You will see the new name appear.

Changing the data type of fields

In addition to renaming them, fields need to be prepared for use in OL Connect Designer templates by setting their data type.

You have already set a default data format. This has had no effect on the extracted data, because by default, all fields are extracted as Strings (text). But for dates, numbers and currencies, other data types are available. Select a data type for these fields to make it easier to use them in a template.

1. Set a field's data type to Date

Two fields in the Data Model actually contain a date. Set their data type accordingly, so they can be interpreted and processed as such in the DataMapper and in templates.

1. On the **Data Model** pane, click **Date**.
2. On the **Step Properties** pane, under **Field Definition**, set the **Type** to **Date**.
3. Under **Data Format**, make sure **Date/Time Format** is set to **Automatic**.
4. Carefully compare the date in the data source to the date in the Data Model. Is the date parsed correctly?
   <Date> has the format day/month/year (dd/mm/yyyy).
   According to the ISO 8601 standard the date in the Data Model should be formatted : yyyy-mm-dd . So "25/06/2016" in the data should be represented as "2016-06-25" with

the added time and time zone.

> **Tip:** Dates in the Data Model can be displayed in local time or as UTC/ISO. From the menu, select **Window > Preferences > DataMapper > Default format**, and set the Data Model time stamps option.

5. Repeat this procedure for the **DueDate** field.

2. Set a field's type to Integer

Integers are whole numbers. In **olsg-data.XML**, there are a few elements that actually contain an integer. Set the data type of the respective fields in the data model accordingly.

1. On the **Data model** pane, select **Ordered**.

   > **Tip:** Hover with the mouse over the Extraction step. Extracted fields are displayed as links that you can click to select the field.

2. On the **Step Properties** pane, under **Field Definition**, set the **Type** to **Integer**.
3. Repeat this procedure for **Shipped** and **Backorder**.

3. Select a field and set its data type to Currency

In **olsg-data.XML**, there is also a number of elements that contain an amount of money. Set the data type of the respective fields in the data model to Currency.

1. On the **Data model** pane, click **InvSubTotal**.
2. On the **Step Properties** pane, under **Field Definition**, set the **Type** to **Currency**.
3. Repeat this procedure for **InvTaxTotal** and **InvTotal**.
4. Also repeat it for the currency fields in the detail table.

> **Tip:** If the extraction fails after you set a field to a different type, check whether its data format is different than expected. You can set its format on the Step Properties pane, under Data Format.
> Note that the default data format is set for all fields in an Extraction step the moment that step is created.

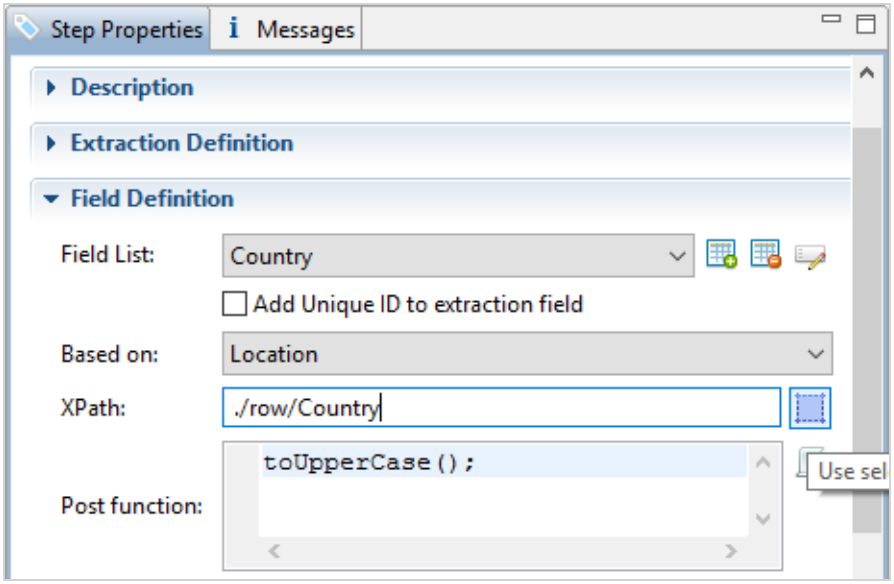
Fine-tuning using JavaScript

This part explains how to fine-tune a record set according to your needs. Understanding JavaScript is an advantage, but for this walkthrough you won't need to write any code yourself.

1. Post function: capitalize country

What if the extracted data is correct, but you'd like the data to be a little different: in capitals for example? Then you could use the "Post function". Code typed in the **Post function** field on the Step Properties pane will be executed after (hence: 'post') the extraction of the selected field, on the extracted data. In order to capitalize the letters of the Country field:

1. On the **Data model** pane, click **Country**.
2. On the **Step Properties** pane, in the **Post function** field, type `toUpperCase();`
   This is a standard JavaScript function to capitalize the letters of a String (a text).



3. Click somewhere else on the **Step Properties** pane and check the result on the **Data Model** pane.

2. Split a field and keep one part of it

Sometimes you will want to keep only one part of the information that has been extracted to a field, and remove the rest. In olsg-data.XML, the Membership level is 'membershiplevel:bronze', 'membershiplevel:silver', or 'membershiplevel:gold'. The word 'membershiplevel' is superfluous. Here's how to remove it from the data field.

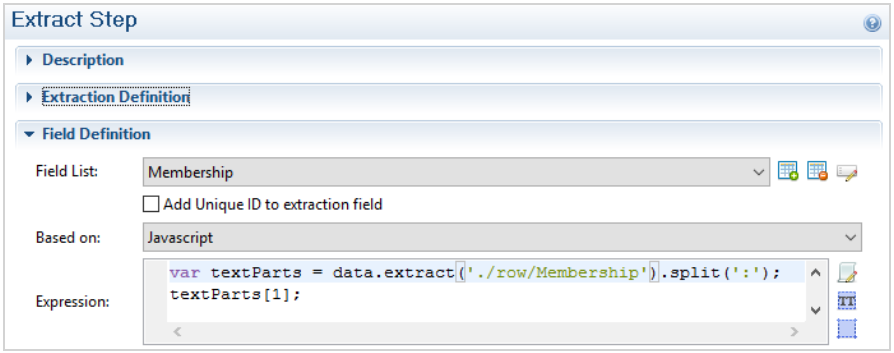
1. On the **Data model** pane, click **Membership**.
2. On the **Step Properties** pane, change **Based on** from **Location** to **Javascript**. The **Expression** field now shows this line of code:

   ```
   data.extract('./row/Membership');
   ```

This is the code that normally extracts data from a specific location, in this case, the data found in the Membership element in a row element.

Replace this line by the following code:

```
var textParts = data.extract('./row/Membership').split(':');
textParts[1];
```



This code extracts the data and then splits the resulting text in two parts, using a colon (':') as the separator.

The parts of the text are stored in a list variable called `textParts`. The first item in this list is membershiplevel, the second item is the level itself.
To get the second item from the list, you need to use `textParts[1];` because the list is a JavaScript array and JavaScript arrays always start counting at 0.

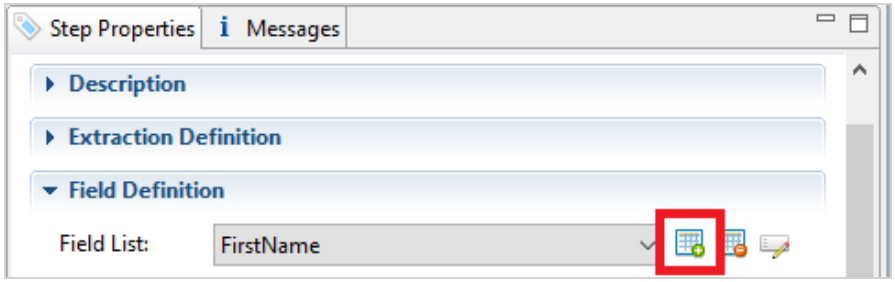
> **Note:** The last line of code is essential: the value of the variable at the end of the code becomes the value of the data field.

3. Take a look at the result on the **Data Model** pane: the Membership field now only contains the membership level itself. The word 'membershiplevel' and the colon have been removed from the data field.

3. Add a concatenated field

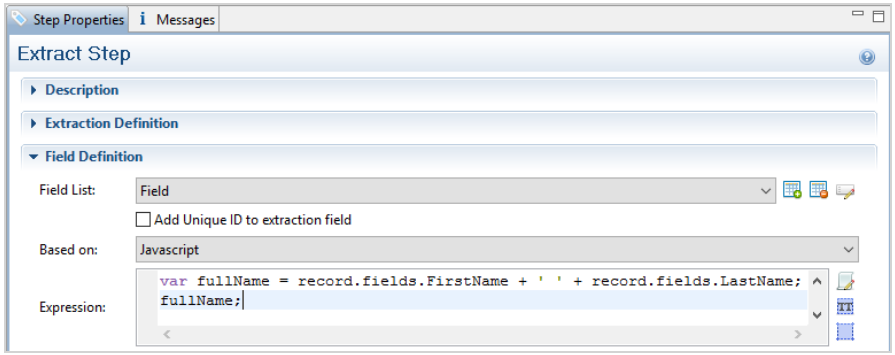
It can be very useful to add a field that isn't filled directly via an extraction. Let's add a field that combines information from two data fields.

1. On the **Steps** pane, click the first **Extraction** step.
2. On the **Step Properties** pane, under **Field Definition**, click the **Add field** button .



3. In **Expression** type this code:

```
var fullName = record.fields.FirstName + ' ' + record.fields.LastName;
fullName;
```



4. Click another field in the **Step Properties** pane. Now you will see the result on the **Data model** pane.
5. Rename the new field to **FullName**.

What's next?

The data mapping configuration is now ready.

It can be used in automated processes to extract data from any XML file that has the same structure as the sample file, olsg-data.xml.

It can be used in OL Connect Designer, to create templates with variable data. To do this, you have to have the data mapping configuration and an OL Connect Designer template open at the same time. The data extracted from the sample data file will be visible in the Data Model pane in OL Connect Designer.

You could also export the Data Model from the DataMapper and import it in the DataMapper again when extracting data from other types of files, or in OL Connect

Designer when creating a template. The exported Data Model doesn't contain the data sample so in these cases the sample data will not be visible.

Designing an invoice template

This step by step walkthrough will teach you how to create a new basic invoice from scratch. It will explain common techniques that are applicable to other document types too.

OL Connect Designer lets you create and design print documents, web pages and emails in one tool. Every aspect of a document can be easily personalized thanks to its feature-rich and user-friendly interface.

The file format of the templates is HTML, the main markup language for web pages and e-mail messages. When exploring the user interface you will see several items that reveal this underlying technology. You will be able to work with the application without knowing HTML, but for users who do know HTML and are familiar with related technologies and standards like JavaScript and Cascading Style Sheets, the user interface does include the appropriate features.

This tutorial will teach you to create a new basic invoice from scratch. It will explain common techniques that are applicable to other document types too. But there is more to learn. To discover new ways and enhance your skills, please visit the OL Resource Center and browse the Online Help.

Note: A PDF version of this walkthrough is available for offline use. Designer_Invoice_WT.pdf.

Note: The resources needed for this walkthrough can be downloaded: Designer-Invoice-resources.zip

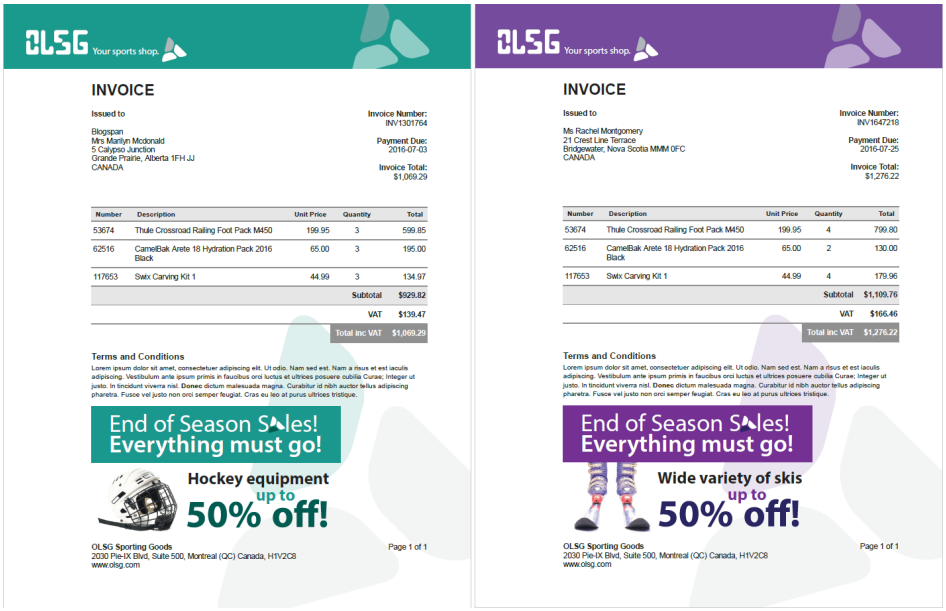
Template requirements

Here are the requirements for the invoice template that needs to be created:

- The template contains a letterhead. Which color it has, depends on the hobby of the receiver: green for skating, purple for skiing.
- The general invoice information, like the invoice number and the due date, should be printed at the top, under the letterhead.
- The invoice data has to be printed in a table. The table should contain a transport line if it runs over two or more pages. The pages should be numbered.

- The company has several offices. The sender's address should be the office that is nearest to the receiver of the invoice.
- To fill any remaining white space a picture should be printed. Which picture is printed depends on the hobby of the receiver. If no white space remains, the picture should not be printed.

The generated invoice will look similar to this picture:



Creating a template

This chapter introduces the concepts of sections, content and formatting. You will also learn how to navigate the content.

1. Creating a template

We'll start by creating a template. A Print template can be printed to a printer, to a file, or it can be attached to an email designed in OL Connect Designer.

1. Launch the Designer by double-clicking the application icon.
2. Choose **File > New... > Template > Print Template** and click **Next**.
   Or, in the **Welcome** screen, under **Create/Open**, click **New Template** and choose **Print**.

   The **New Print Template** wizard is shown. Here you can enter parameters for the new document, including its page size, the page margins and the number of sections.
   A **section** can contain one or more pages. Text will flow automatically from one page to the next.

3. Make the following settings:
    - Page size: Letter
    - Margin Top: 1.4in
    - Margin Left: 1.6in
    - Margin Bottom: 1.2in
    - Margin Right 0.8in
4. Click **Finish**. The new document is shown in the main area of the user interface.
5. On the **Resources** pane at the left, expand the **Context**, and then the **Print** folder. Right-click **Section 1** and rename the section to **Invoice**.
6. To save the document, choose **File > Save As**. The **Save Template** dialog box displays.
7. Navigate to a folder (the folder where you unzipped the Walkthrough files, for example), enter **My OLSG Invoice.OL-template** in the **File name** field and then click **Save**.

2. Importing images

The next step is to import images into the template. Images can be static elements or they can be dynamically replaced by expressions or scripts based on a data field. They can also be used as a dynamic element to fill white space, as you will see in a later exercise.

1. In case the Resources pane is not shown, choose **Window > Show View > Resources**.
2. Locate the images folder provided with this tutorial.
3. Copy **OLSG_letterhead-green.pdf** (right-click, then select Copy) to the **Images** folder in the **Resources** pane (right-click > Paste).

3. Adding a Virtual Stationery background

In this exercise you will learn how to set a PDF image as a background to emulate pre-printed paper. After specifying a so called "virtual stationery image" the pre-printed media will be displayed so that you can place the content in the correct position. The image will be repeated for all pages that use this media.

To set a PDF as media:

1. Locate the file **OLSG_letterhead-green.pdf** in the **Images** folder on the **Resources** pane (the image was added in the previous exercise).
2. Drag **OLSG_letterhead-green.pdf** to the **Media 1** entry in the **Media** folder on the Resources pane. The first page in the PDF will be set as the front side image of the media.

After adding the image to the Media you will see that the image appears in the background of the page(s).

What if you dropped the wrong PDF on the Media or want to change it?

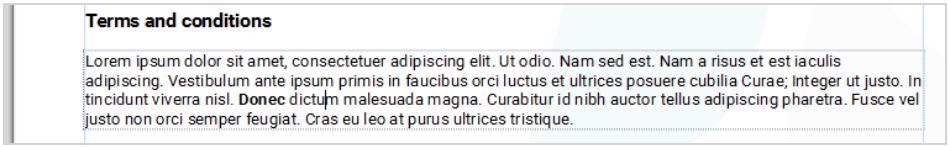
The **Media properties** dialog allows you to delete a PDF file from a Media. It also lets you select an image for both the front and back of the Media and specify the position of the image.
To open the dialog, right-click the **Media 1** entry and select **Virtual Stationery** from the shortcut menu.

> **Note:** By default, Media backgrounds are omitted from the output (because in the end, the pre-printed paper will have this imagery). In case you are printing on blank paper you could print this background image together with the document by selecting the **Print Virtual Stationery** option in the Advanced section of the Print Wizard interface or in an Output Creation Preset.

4. Adding text and applying basic formatting to it

In OL Connect Designer every print section consists of a main text flow. This is the area defined by the page size and margins. The margin guides are shown in light blue.

1. Place the cursor at the start of the document.
2. Type **INVOICE**.
3. Select the word INVOICE and change the style from Paragraph to **Heading 1** using the **Formatting** toolbar.
4. Press **Enter** and type **Terms and Conditions** on the next line. Change the style to **Heading 3**.
5. Press **Enter** and then click the **Insert Lorem Ipsum** toolbar button to enter dummy text.
6. Select one word in the dummy text and use the **Bold** toolbar button to make it bold.

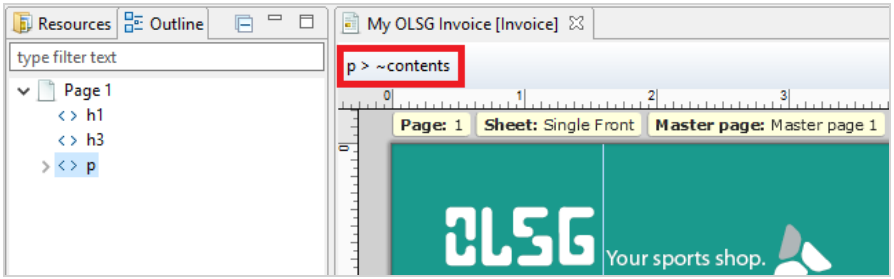


Alternatively you could copy-paste text from a text processor, or from a Microsoft Word file. Basic style characteristics specified in Word travel with the content to OL Connect Designer. Formatting options like bold, italic and formats like Heading 1, Heading 2 are maintained.

5. Exploring the document

There are two easy ways to explore and quickly select elements on the page: using the **Outline** pane and the **breadcrumbs**. The Outline pane at the left shows a hierarchical tree structure of the pages and their elements. Advanced users will refer to this as the HTML Document Object Model or DOM.

1. The **Outline** pane is located in the same area as the Resources pane. Click the Outline tab to open it. If the Outline tab isn't visible at all, choose **Window > Show View > Outline**.
   The Outline pane lists the elements that are currently present in the document: two headings and one paragraph. The subheadings are marked with <h1> and <h3> and the paragraph is represented by the <p> entry in the tree.
   A <p> element defines a paragraph in HTML. Other common HTML elements are:
   - h1 through h6 (headings)
   - img (image)
   - table
   - tr (table rows)
   - td (table data: table cells)
   - span (a piece of styled text inside another HTML element)
   - div (a box).
2. Click the **<p>** element in the Outline pane. The main document window jumps to that paragraph and shows an outline around its contents. The paragraph is now selected and you can see a **p** in the bar below the tab that holds the title of the document. This line is referred to as the **breadcrumbs**.



3. Click on **INVOICE** in the text. Note that both the Outline pane and the breadcrumbs are updated to show that the currently selected element is **h1**.
4. Click on the word under **Terms and Conditions** that you've bolded. The breadcrumbs show that the selected, bold word is located inside a paragraph: **p > b > ~contents**.

5. Now click on the **p** in the breadcrumbs. Note that the paragraph gets selected in both the text and the Outline pane.
6. With the paragraph still selected, select font-size **9pt** in the Formatting toolbar to reduce the size of the Lorem ipsum text.
7. Save the template.

Adding personal data

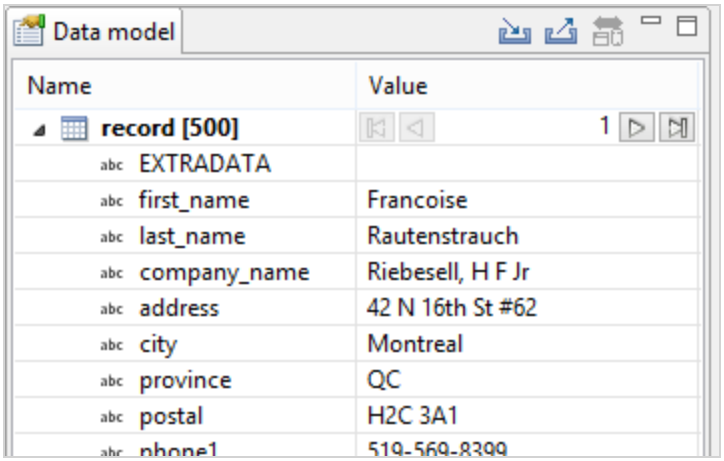
To create any personalized document, you need data about the recipients. Typically such data include fields like the first name, last name, email address, job title etc. This chapter shows how to load data and demonstrates various techniques to add data fields to the page.

1. Loading data

In this exercise you will learn how to load data by opening a **data mapping configuration**. It is also possible to open a data file or database in the Designer, but transactional data – such as line items in an invoice – can't be imported in the Designer directly. That's why we need a data mapping configuration.

1. Go to the **Welcome** screen by clicking the **Home** icon at the top right.
2. On the right, under **Create/Open**, click **Open File**. Change the file type to DataMapper files (*.OL-datamapper).
3. Select the data mapping configuration provided with this tutorial: **OLSG-invoice.OL-datamapper**, or a data mapping configuration made by you with a DataMapper walk-through.
   The DataMapper workspace is now shown.
4. Toggle back to the Designer perspective by clicking the **My OLSG Invoice.OL-template** tab at the top of the workspace.
5. The **Data Model** pane now shows the data model and the data of the first record in OLSG-invoice.OL-datamapper. If not, hit the **Synchronize Fields and Structure** button in the Data Model pane to update the Data Model to the current data mapping con-

figuration.



6. Use the browse buttons at the top of the Value column to browse through the imported data.
7. Scroll down in the Data Model pane to see the **detail table** and browse through its detail records.

2. Adding a table to hold the address and invoice information

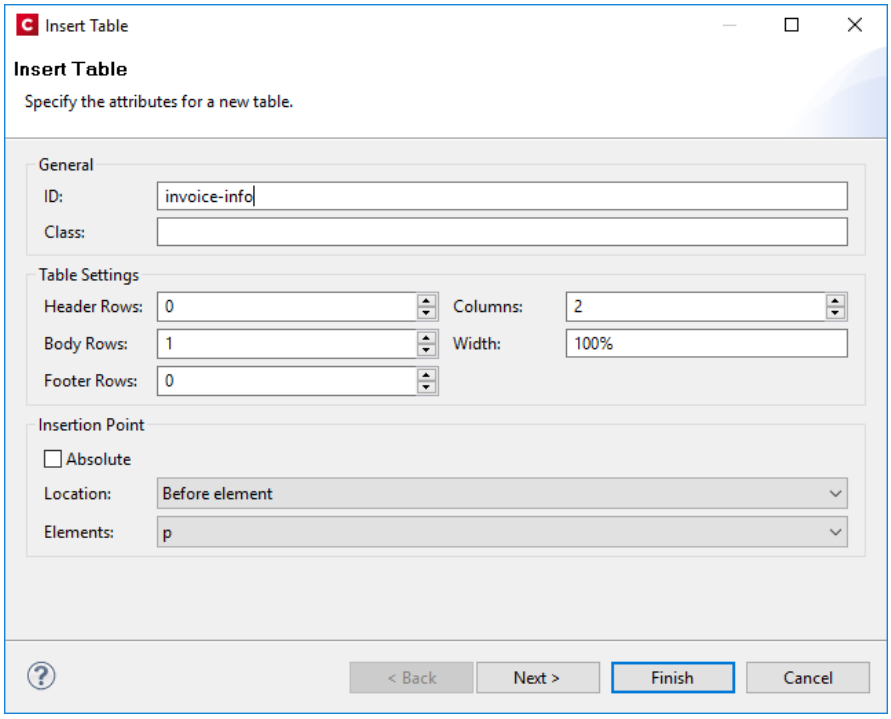
A table can function as a container for the address of the client and the invoice information. The table, which in this case is placed in the main text flow, will have two columns: the left column is meant to hold the address data and the one on the right contains invoice information.

The following image shows the end result.

To add a table to the main text flow:

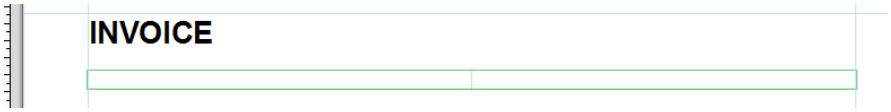
1. In the invoice template, create a new paragraph after the INVOICE heading by pressing the Enter key after the last character of the heading.

2. Select **Insert > Table > Standard...** from the Menu or click the Insert Table icon on the toolbar. The **Insert Table** dialog appears. This dialog lets you configure the new table.



3. Make the following settings on the first page of the **Insert Table** wizard:
    - ID: invoice-info
    - Header Rows: 0
    - Body Rows: 1
    - Footer Rows: 0
    - Columns: 2
4. Use the **Location** drop-down to select **Before element** as the insertion point. This will insert the table before the new paragraph.
5. Click **Next** and select the table style at the top left, which does not have any borders or background colors. The table will take up the maximum available space horizontally and runs between the left and right margin guides.
6. Click **Finish** to insert the table.
7. As the table doesn't have any borders or background it blends into the page. Click the **Show Edges** button on the toolbar to reveal its position. The table shows a non-print-

ing green border.



3. Adding the invoice information

In this exercise you will add the invoice information like the invoice number, due date and invoice total to the table. You will do this by dragging the data fields into the template.

1. Place the cursor in the right table cell of the initial table.
2. Type the following text:
    - Invoice Number: (press Enter)
    - Payment Due: (press Enter)
    - Invoice Total: (press Enter)
3. Now, to add the variable data, drag the data fields **InvNumber**, **DueDate** and **InvTotal** one by one from the Data Model pane into the table, after the respective label. To push them to the next line, place the cursor before the first {{, press the Enter key once.
4. Select the text of each label and click the **Bold** icon on the toolbar to set the font-weight for the selected label.
5. Click the **Preview** tab along the bottom of the main editor to view the merged template. When viewing the document in Preview mode (or when generating output), the scripts are executed, by which the placeholders get replaced with the value of the data field.

4. Adding the address

In this step you will learn how to add variable data, not by dragging data fields to the document but by using the **Handlebars Helper wizard** instead.
One of the big advantages of using a Handlebars Helper wizard is that empty lines and spaces will be removed when data fields are empty.

The Handlebars Helper wizard creates a *Helper* that concatenates multiple data fields and that can add a prefix and/or suffix to each data field.

Note: A *Helper* is a function that can be used in an expression.

Tip: Although the Helper wizard *can* be used to create an address block, like we do in this walkthrough, it is actually much easier to add an address block with the Address Block wizard.

1. Type the following text in the first table cell: **Issued to**.
2. Select 'Issued to' and click the **Bold** icon in the **Formatting** toolbar.
3. Click after 'Issued to', press **Enter** two times
4. Open the **Handlebars Text Helper** wizard: on the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Handlebars Text Helper**. A script entry is added to the Scripts pane and opened in the **Edit Text Helper** dialog.
5. Change the **name** of the script so that it clearly describes the purpose of the script (e.g. setAddress). The name must be unique.
6. The **Handlebars Text Helper** wizard is able to combine data from multiple data fields. Each field is placed on its own line in the wizard. New lines can be added by clicking the Plus sign next to the table or simply by clicking on the next empty line.
   The **Prefix** and **Suffix** fields add text to the data of each field.
   Important! When a data field is empty the respective line is skipped including its prefix and suffix.

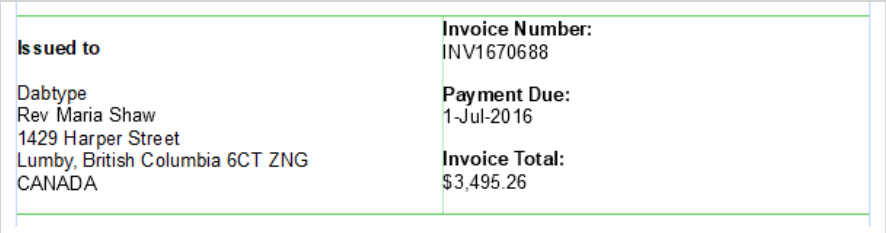
   Add lines for the following fields and set the respective suffix text.

Prefix	Field	Suffix
	Company	 (adds a line break)
	Title	one space
	FirstName	one space
	LastName	
 	Address1	
 	City	, (comma, then space)
	State	
one space	ZipCode	
 	Country	

> **Tip:** For two spaces to be used as a prefix or suffix, enter two non-breaking space characters (represented by the HTML entity: ** **) instead of two regular spaces. In HTML by default multiple regular spaces will be collapsed to a single space character.

7. At the bottom of the dialog, click **Options** and verify that the **Insert Method** is **HTML**
8. Click **OK**. The **Handlebars Text Helper** will be added to the **Scripts** pane under **Control** scripts folder.
9. Drag and drop the helper setAddress from the **Scripts** pane to the template after the text 'Issued to:' in the first cell of the invoice information table. Alternatively, you can write the name of the Helper enclosed in double curly braces {{setAddress}}.
10. Toggle to the Preview mode and use the browse buttons of the Data Model pane to navigate through the records. The address block should look like this:

    Company
    Title First name Last Name
    Address1
    City, State Zip code
    COUNTRY



Issued to	**Invoice Number:** INV1670688
Dabtype Rev Maria Shaw 1429 Harper Street Lumby, British Columbia 6CT ZNG CANADA	**Payment Due:** 1-Jul-2016 **Invoice Total:** $3,495.26

5. Changing the date format

Predefined date format

When you drag and drop data from the Data Model, they will initially have the default format. For a date, this is the *medium* date format, for example "Apr 1, 2022". To change this, you need a **Format Helper**.

In this exercise you'll change the format of the due date to the *long* date format in which the month is written out in full, for example "April 1, 2022".

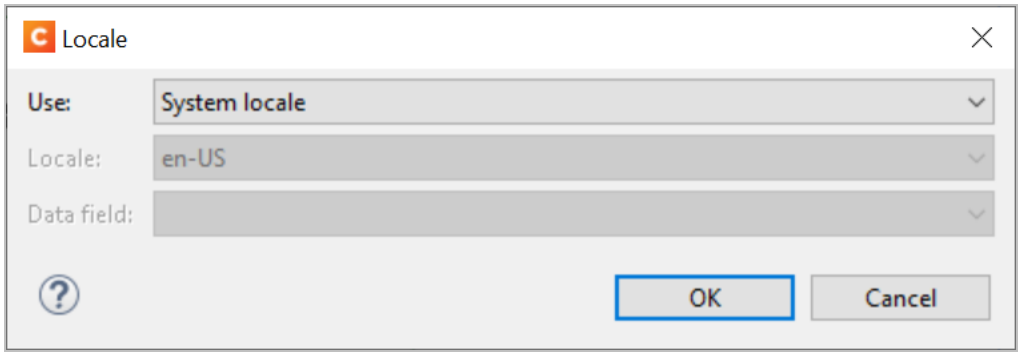
1. In Design mode, click the cell that contains {{date DueDate}}.
2. In the expression, change **date** into **dateLong**. There should be a space between **dateLong** and **DueDate**: **{{dateLong DueDate}}**.
3. Toggle to the **Preview** mode and use the browse buttons of the **Data Model** pane to navigate through the records.

Note: "dateLong" is a *Format Helper*. OL Connect comes with a number of Helpers to format a value in an expression. See Format Helpers.

Locale

The actual date format also depends on the locale, set for the document.

1. Choose: **Edit > Locale...** The **Locale** dialog appears. By default the template makes use of the system locale.



2. Change the **Use** field to **Explicit Locale** and select a different entry from the **Locale** drop-down.
3. Click **OK** to store the new setting and see its effect.

Tip: The **Data field** drop-down in the Locale dialog gives you the possibility to use a variable locale, that is, if your data contains a field that holds a locale code. The locale will be record-specific in this case. Language codes (en, fr etc.) and language-country codes (en-CA, fr-FRA) are both valid locales.
Yet another option is to use a runtime parameter. Runtime parameters can be defined in the template and filled with a value in Workflow.

Custom date format

In addition to the predefined date formats, it is possible to use a custom format like "MMM/yyyy/dd". You can specify a custom pattern after the name of the data field, and enclose it in double quotes.

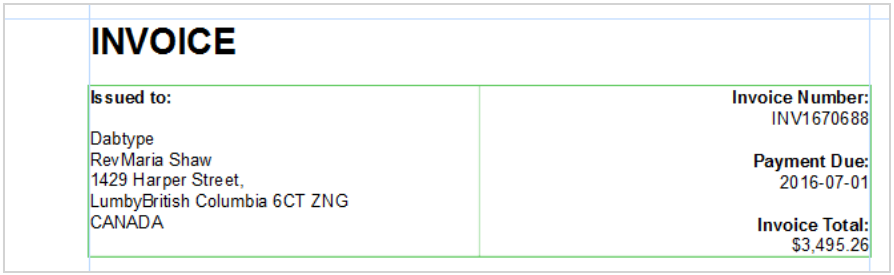
In this exercise you'll change the format of the due date to *MMM/yyyy/dd* custom date format, for example "01/2022/17".

1. In Design mode, click the cell that contains {{dateLong DueDate}}.
2. In the expression, change **dateLong** into **date**. Put the custom pattern *MMM/yyyy/dd* after the data field name DueDate separated with space like this: **{{date DueDate "MMM/yyyy/dd"}}**.
3. Toggle to the **Preview** mode and use the browse buttons of the **Data Model** pane to navigate through the records.

6. Aligning text in table cells

The texts in the two table cells are not aligned at the same height. This is because the text is aligned vertically to the middle of the cell. Changing the alignment of the text in the table cells will make the invoice prettier.

1. Switch to the **Design** tab.
2. Select the paragraphs in the right table cell, one by one, and click the **Align Right** icon in the toolbar.
3. Select the table as a whole. You can do this by clicking **table #invoice-info** in the breadcrumbs.
4. Select **Format > Table Cell...** from the Menu.
5. On the **Cell** tab, set the **Vertical Align** drop-down to **top**.
6. Click **OK**. The text in both table cells will now be aligned to the top of the cell.
7. Switch to the **Preview** tab to see the result.



Adding the dynamic invoice data

The Dynamic Table feature provides an easy way to lay out rows and columns for line item information such as the description of a product, the unit price, total price, quantity etc. Table rows are automatically inserted based on the number of rows in a detail table. Once the table reaches the bottom of the page a new page is inserted automatically. The invoice

will continue on the next page. Headers and footers can be repeated across pages, and subtotals, transport lines and the invoice's total can be added automatically.

You'll learn how to insert a Dynamic Table to lay out the information of a detail table stored in the data set.

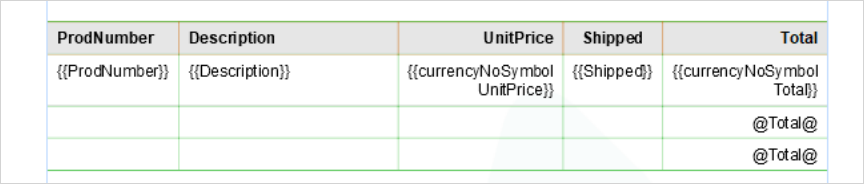
1. Adding the dynamic table

1. Toggle to the **Design** mode and place the cursor at the beginning of the heading 'Terms and Conditions'.
2. Choose: **Insert > Table > Dynamic Table...**
   The **Insert Dynamic Table** dialog appears.
3. Change the value of the **ID** field to: **invoice-data**.
4. Make sure that the **Detail table** field reads **products**. This is the detail table in the data that holds the invoice line items.
5. From the **Location** drop-down, choose **Before element**. This causes the table to be inserted directly before the level 3 heading (h3).
6. Click **Next**. The Dynamic Table Builder is shown.
   It contains one '***row***' which is associated to the selected detail table. The row will be repeated in the output (or in Preview mode) as many times as necessary to display all the records in the associated detail table.
   The '***fields***' - cells in the table - correspond to data fields in the detail table.
7. Delete the "Ordered" and "BackOrder" fields by clicking on the red cross. The following fields should be listed:
   - ProdNumber
   - Description
   - UnitPrice
   - Shipped
   - Total

   The fields will appear in the table in the order in which they are listed here. To rearrange them, you can drag and drop them by clicking to the left of a field.
8. Without currency signs the values in the Unit Price and Total columns will look better. Change the **Format** for these fields to **Currency No Symbol**.
9. Note how the **alignment** is set for text in different kinds of fields: text is left-aligned, amounts in currency fields are right-aligned and numbers are centered.

10. Click **Next**. The Dynamic Table Options page is shown. The options on this page let you add subtotals to the footer and transport lines (in case the table runs across multiple pages).
11. Check the **Calculate Subtotals** item.
12. Verify that **Total** is selected as the data field that will be used to calculate the subtotals and totals.
13. Check **Show subtotal lines before page break** options to have subtotals shown before page break and after end of the table.
14. Click **Next**. The predefined table styles are shown. Select the 6th table style (the second from the top right) and click **Finish** to insert the table.
    How to further style the table is demonstrated later in this walkthrough.

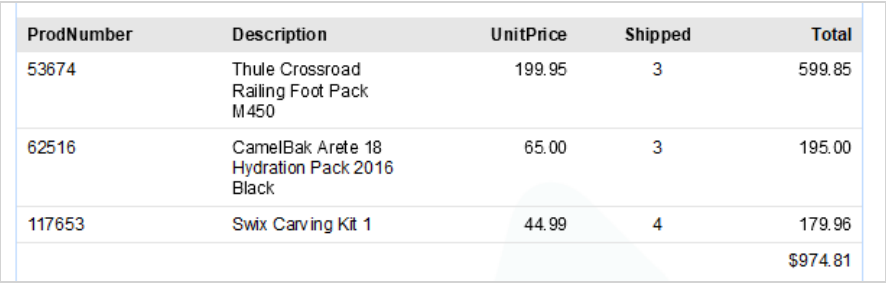
    The table is inserted at the current cursor position. It has a column for each of the selected data fields. The cells in the body table row, between the header and footer rows, each have an expression (the name of a data field, enclosed in double curly brackets). The *currencyNoSymbol* Format Helper formats the amounts as a currency without a currency sign.



ProdNumber	Description	UnitPrice	Shipped	Total
{{ProdNumber}}	{{Description}}	{{currencyNoSymbol UnitPrice}}	{{Shipped}}	{{currencyNoSymbol Total}}
				@Total@
				@Total@

> **Tip:** Click the **Show Edges** button on the toolbar to show the edges of the table with thin green lines. These lines are visible in Preview mode, but will not be printed.

15. Toggle to **Preview** mode. You will see that the body table row is repeated for each detail record and that the expressions are replaced with the corresponding values.



ProdNumber	Description	UnitPrice	Shipped	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
				$974.81

Note that there is only *one* subtotal row on each page in output and in Preview mode, whereas in Design mode, you can see two. The bottom footer row is only shown on the

last page. The row above it is shown on all other pages, if the table runs over multiple pages.

2. Adding labels and rows

At this point the table shows the basic information of the invoice line items. In this exercise you'll learn how to add additional rows and information to this table.

To add a **label** to the subtotals in the footer:

1. In Design mode, place the cursor in the cell next to the @Total@ cell in the first row of the footer of the table. Type **Subtotal**.
2. Do the same in the bottom footer row.

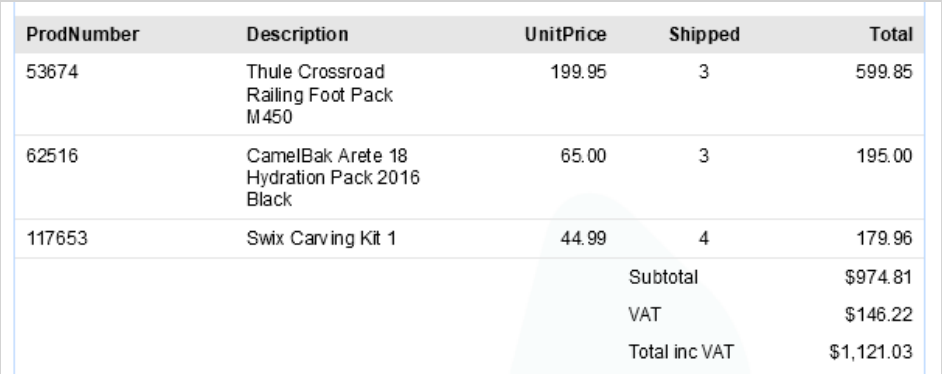
Now add **additional rows** for the tax and invoice total:

1. Right-click one of the cells in the bottom row and choose **Row > Insert Below** from the shortcut menu. Repeat this to add a second row.
2. On the **Outline** pane, click on the second to last footer row (the third **<tr>** in **<tfoot>**).

   Alternatively you click in the row and then on **tr** in the breadcrumbs: … > tfoot > **tr** > td …
3. On the **Attributes** pane, change **Show Row** to: **At end of table**. This prevents that a row is shown on every page if the table runs over multiple pages.

   > **Tip:** The **Show Row** setting on the Attributes pane can also be used to show a row at the top or bottom of every page. For example, for a row that reads: *To be continued on the next page*, you would set it to **Before page break**.
4. Repeat the previous step for the last footer row.
5. Place the cursor below the last Subtotal cell and type: **VAT**.
6. Move the cursor one row down and type: **Total inc VAT**.
7. Locate the **InvTaxTotal** data field in the Data model panel and drag this field into the cell next to the VAT cell.
8. Drop the **InvTotal** data field in the last table cell of the table.

Now, in Preview mode the table will look like this:



ProdNumber	Description	UnitPrice	Shipped	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	$974.81
			VAT	$146.22
			Total inc VAT	$1,121.03

Formatting the Dynamic Table

This chapter explains how to format a Dynamic Table. You will not only change the text and text alignment, but also change the column widths, borders and background colors.

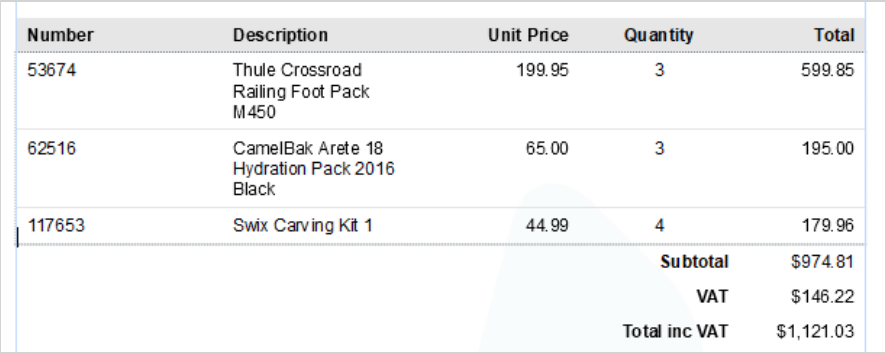
Note: Changes are made in **Design** mode.

1. Formatting text and currencies

In this exercise you will change some labels, text alignments and the formatting of amounts of money. Formatting text in a Dynamic Table is no different from how it is done in a Standard Table.

1. The headers of the columns contain the name of the respective data fields. In Design mode, change them to proper headings: change ProdNumber to **Number**, UnitPrice to **Unit Price** and Shipped to **Quantity**.
2. Select the first **Subtotal** label and click the **Bold** icon on the toolbar. Then set the text alignment of the cell to the right by clicking the **Align Right** icon on the formatting toolbar.
3. Do the same with the other labels in the footer: **Sutbtotal**, **VAT** and **Total inc VAT**.

4. Toggle to **Preview** mode to verify your changes. The table should look like the image below:



Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	$974.81
			VAT	$146.22
			Total inc VAT	$1,121.03

We have already set the format for the Unit Price and Total fields when adding the table. To change the formatting of the **subtotals** after adding the table:

1. In Design mode, click on the cell.
2. On the **Attributes** pane, select the desired format from the **Format** drop-down.

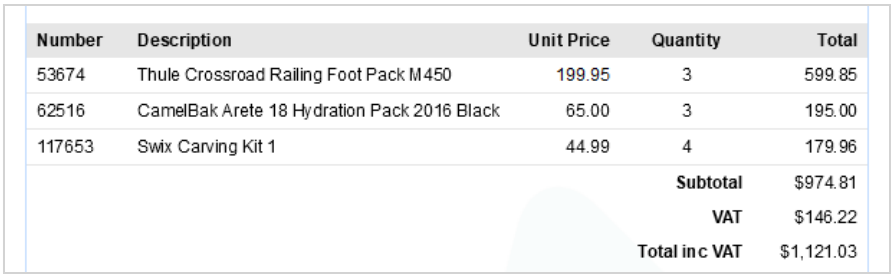
If you would not want to display the currency sign in the VAT and Total inc VAT fields, click in the cell and replace the *currency* Format Helper with *currencyNoSymbol*.

2. Changing column widths

The product descriptions are rather long. It would be best if most of them would fit on a single line. This can be achieved by changing the column widths.

1. In **Design** mode, hover the mouse between the Number and Description columns. The cursor changes to a column resizer icon.
2. Click and drag the column separator to the left. Then resize the other columns, from right to left, dragging the separator to the right.
3. Toggle to **Preview** mode to see the effect. You will see that the widths could be improved in columns where the actual data are less wide than the placeholders.
4. Toggle to **Design** mode, click in the field that holds the **Number** label, and on the **Attributes** pane, in the **Geometry** section, set **Width** to a smaller percentage, e.g. 10%.

5. Toggle to Preview mode and back to Design mode, and change the widths of cells in the header row until all labels and (most) line items fit on a single line.



Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	$974.81
			VAT	$146.22
			Total inc VAT	$1,121.03

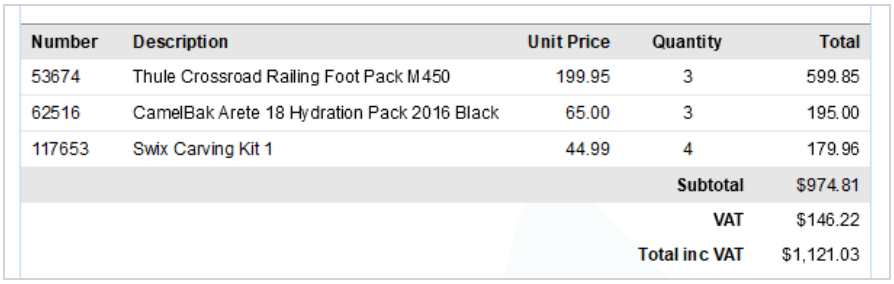
6. To ensure that the widths can't be changed accidentally, we will disable the option to resize the table. Select the table: click somewhere in the table, and then click on **table #invoice-data.table--light** in the breadcrumbs.
7. On the **Attributes** pane, uncheck the option **Allow Resizing**. (You may have to scroll down in the Attributes pane to see this option.)

3. Using the Formatting dialog

In this exercise you will learn how to use the **Table Cell Formatting** dialog. The next exercise shows how to use the Stylesheets dialog.

1. In **Design** mode, place the cursor in the first cell of the header row. Press the **Shift** key and click the last cell in this row. The cells of the header row are now selected.
2. Right-click one of the selected cells and select **Cell...** (note the dots) from the contextual menu. The Table Cell Formatting dialog appears.
3. Click the **Border** tab.
4. Deselect the **Same For All Sides** option and make the following settings for the **Top** border:
    - Width: 1pt
    - Style: solid
5. Click the small square next to the **Color** field. The **Color** dialog appears. Under RGB, enter:
   R: 144, G: 144, B: 144.
6. Click **OK** to close the Color dialog.
7. Click **OK** again to apply the border settings to the selected cells.
8. Select the cells in both Subtotal rows and open the Table Cell Formatting dialog.
9. Click the **Background** tab. Click the small square next to the **Color** field. The Color dialog appears.

10. Under RGB, enter: R: 230, G: 230, B: 230. Click **OK** to close the Color dialog and OK again to close the Table Cell Formatting dialog. The table now looks like this:



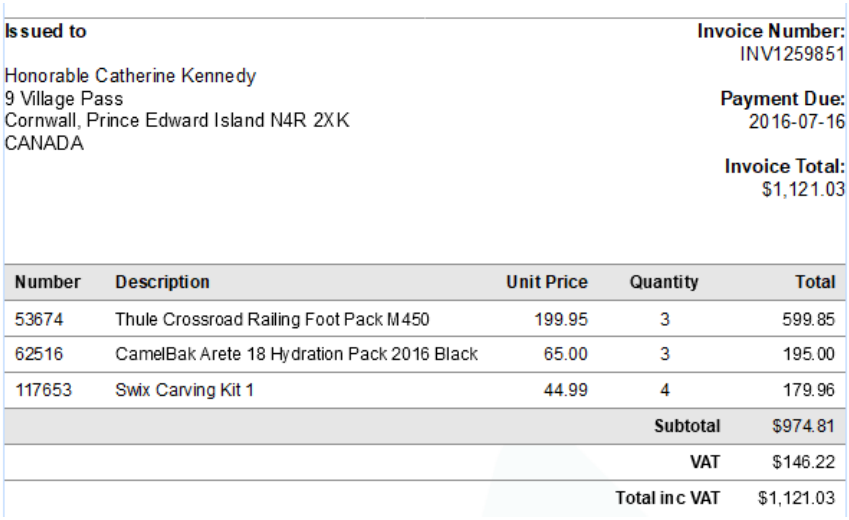
Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	$974.81
			VAT	$146.22
			Total inc VAT	$1,121.03

4. Using the Stylesheets dialog

Horizontal lines make it easier to read the data and they make the table look better. This exercise shows how to change borders and background colors of table cells with CSS (Cascading Style Sheets), via the Stylesheets dialog.

1. Choose **Edit > Stylesheets...**, to make the **Stylesheets** dialog appear.
2. From the **Context** drop-down, choose **Print**, to make the styles apply to Print sections only.
3. Click the **Plus** icon to create a new style rule.
4. In the dialog that appears, type **td** in the **Selector** field. The selector defines to which elements in the document the rule applies. By stating **td** the rule applies to all table cells. In HTML, a table data cell is a `<td>`.
5. Click the **Border** tab.
6. Deselect the **Same For All Sides** option and make the following settings for the **Top** border:
   - Width: 1pt
   - Style: solid
7. Click the small square next to the **Color** field. The **Color** dialog appears. Under **RGB**, enter:
   R: 144, G: 144, B: 144.
8. Click **OK** to close the Color dialog.
9. Click **OK** to close the rule editor.
   The top part of the Stylesheets dialog lists selectors (up to now there's only one: **td**) and the preview in the bottom part lists all style rules for the selected selector. In the style rule that you've just made, "border-top" is the CSS property, and "1pt solid rgb (144, 144, 144)" is the value assigned to that property.

10. Click **Save** to close the Stylesheets dialog.
11. Toggle to **Preview** mode and look carefully at both tables in the template: the standard table with the invoice and address information and the Dynamic Table.



Issued to

Honorable Catherine Kennedy
9 Village Pass
Cornwall, Prince Edward Island N4R 2XK
CANADA

Invoice Number:
INV1259851

Payment Due:
2016-07-16

Invoice Total:
$1,121.03

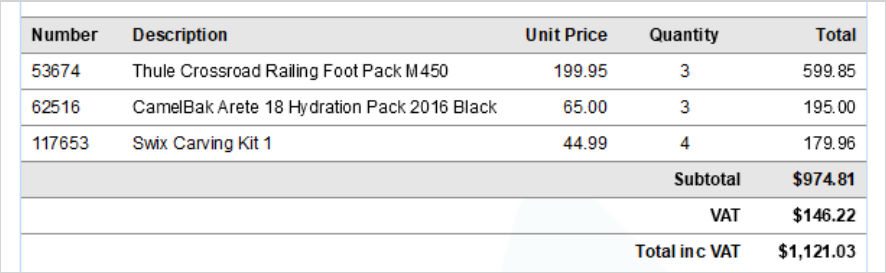
Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	$974.81
			VAT	$146.22
			Total inc VAT	$1,121.03

As you can see, the newly added style rules don't just affect cells in the Dynamic Table. They also apply to the table with the invoice information. This is because the *selector* selects all cells in all tables. So, the selector has to be more specific.

12. Reopen the Stylesheets dialog and select the context: **Print**.
13. Select **td** and click the **Edit** button (with the blue pencil).
14. Change the selector to **#invoice-data td**. Save the settings. Now, the style rule only applies to table cells inside an element that has the ID: invoice-data – which is the Dynamic Table.
15. Create a new rule-set: click the **Add** button, type the selector **#invoice-data tfoot** and on the **Type** tab, check **Bold**. This bolds all text in all footer rows of the Dynamic Table.

> **Tip:** The options in the Rule editor represent the most frequently used CSS properties. If a property is not available here, click the **Advanced** button. In Advanced mode you can add a style rule by typing a CSS property (at the left) and its value (at the right).

16. Save the settings. In Preview mode, the table should now look like this:

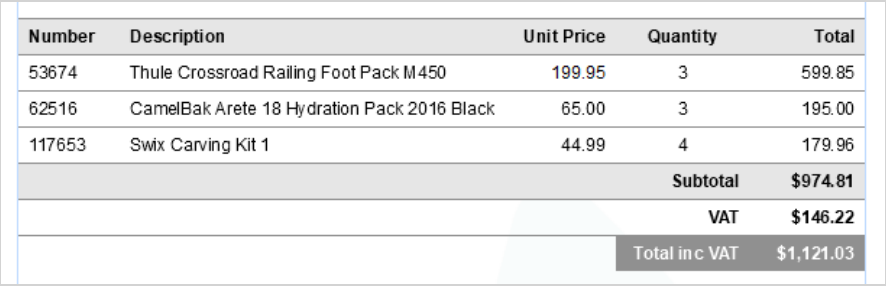


Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	**$974.81**
			VAT	**$146.22**
			Total inc VAT	**$1,121.03**

17. A different background color and text color will make the total amount of the invoice stand out. Coloring two cells with one style rule is easy if they have the same class.
In Design mode, click in the cell that contains the text Total inc VAT and use the bread-crumbs or the Outline pane to select the cell (**td**).
18. On the **Attributes** pane, in the **Class** field, type **invtotal**.
19. Repeat the previous two steps for the cell that contains the {{currency InvTotal}} placeholder.

    Note: Elements can have more than one class. Separate class names by a space.

20. Open the Stylesheets dialog, and select the context: **Print**.
21. Click the **Add** button to create a new style-set.
22. As selector, type **.invtotal** (note the dot). This will make the new rules apply to any element that has the class **invtotal**.
23. On the **Type** tab, in the **Color** field, type **White**. This turns the text color into white.
24. Click the small square next to the **Background Color** field. The Color dialog appears.
Under **RGB**, enter:
R: 144, G: 144, B: 144.
25. Click **OK** to close the Color dialog. Now the table looks like this:



Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	**$974.81**
			VAT	**$146.22**
			Total inc VAT	**$1,121.03**

5. Editing a style sheet

As you've seen in the Stylesheets dialog, a rule consists of a **selector** and a set of **style rules**.

The selector points to the HTML element you want to style. The style-set contains one or more declarations, each ending with a semicolon. A declaration includes a property name and a value, separated by a colon.

In this exercise you will add padding to table cells and change the font size by editing a CSS file directly, instead of via the Stylesheets dialog. As you gain more knowledge of CSS, this option will become increasingly convenient.

1. On the **Resources** pane, expand the **Stylesheets** folder and double-click **context_print_styles.css**. The file opens in the Workspace. You will see that it already contains the selectors and style rules that were added via the Stylesheets dialog.
2. Add the following style rules to the style-set for **#invoice-data td**:

   ```
   font-size: 10pt;
   padding: 6pt 2pt;
   ```

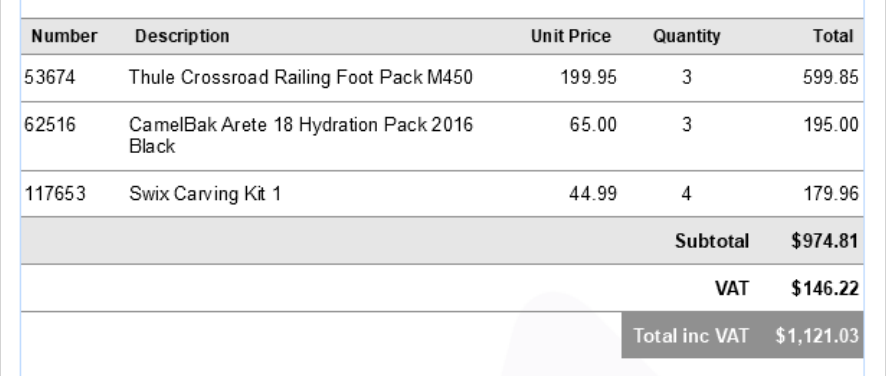
   Padding is the margin inside an element, in this case, a table cell.
3. Add the following style rules to the style-set for **#invoice-data**:

   ```
   margin-top: 10pt;
   ```

   The margin-top sets the top margin outside an element, in this case, a table.
4. Press **Ctrl** + **S** to save the style sheet.

   > **Note:** Never edit a CSS file in the workspace and via the **Stylesheets** dialog at the same time!

5. Click the **My OLSG Invoice** tab and toggle to Preview mode. Because of the padding and the changed font size, you will need to resize the columns to make most product descriptions fit on one line again. (Enable the Allow Resizing option first.)
   The final table layout will look similar to the following image:



Number	Description	Unit Price	Quantity	Total
53674	Thule Crossroad Railing Foot Pack M450	199.95	3	599.85
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	3	195.00
117653	Swix Carving Kit 1	44.99	4	179.96
			Subtotal	$974.81
			VAT	$146.22
			Total inc VAT	$1,121.03

Adding a variable address (using a snippet, master page and script)

The only thing that's still missing in the invoice is the address of the sender. The sender is a company that has offices in different cities. All addresses are available in one *snippet*. In this exercise you will learn how to write a **script** that inserts part of a snippet in the template, depending on the value of a field in the record set.

1. Importing a snippet

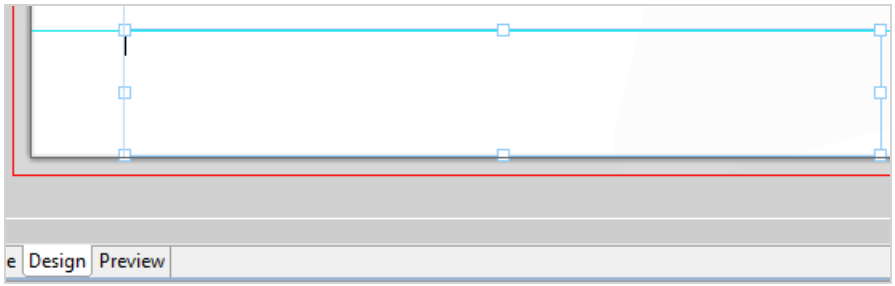
Snippets are small HTML files. They are imported in much the same way as images are imported.

1. Locate the snippet **olsg_address.html**, provided with this tutorial.
2. Copy and paste the snippet into the **Snippets** folder in the **Resources** pane.
3. Click **OK** to confirm that the encoding is UTF-8.
4. Double-click the new snippet to open it in the Workspace.
5. Click the **Source** tab at the bottom of the Workspace to view the HTML of the snippet. Every paragraph in this snippet has an **ID** that is identical to one of the values used in the *StateAbbrev* field, as you can see in the Data Model pane. These IDs will eventually be used to select a paragraph from the snippet and insert that in the template.
6. Close the snippet by closing the tab.

2. Inserting a Box on the Master Page

In this invoice, the letterhead is the same on every page and the address should appear on every page as well. The best place for this type of elements is the **Master Page**. In this exercise you will learn how to create a Box on a Master Page.

1. On the **Resources** pane, expand the **Master pages** folder.
2. Double-click **Master page 1** to open it.
3. Click the **Insert Positioned Box** toolbar button.
4. Drag and resize the Box to fit on the lowest part of the page, as in this image:



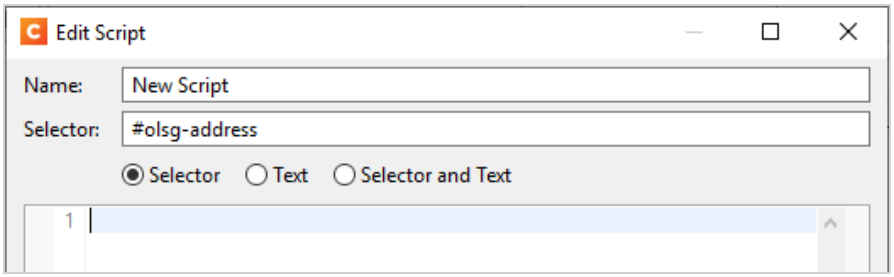
5. Give the Box an ID: on the **Attributes** pane, in the **ID** field, type **olsg-address**. This will be used as the selector of the script. In the box, you could type a placeholder, for example {address}, but that isn't really necessary. In this case the script will simply replace all of the contents of the box.
6. Save the Master Page (**Ctrl S**) and reopen the Invoice template (on the **Resources** pane, expand **Contexts**, **Print** and then double-click the Invoice template).

Note: Although in this walkthrough we will only use one Master Page, it is common to have more than one Master Page and apply different Master Pages to different pages in a template. This is called **Sheet Configuration**. To open the Sheet Configuration dialog, open the Resources pane, expand Contexts, then Print; right-click the Invoice template and select Sheet Configuration.

3. Write a script that uses a snippet

In this exercise you will learn how to write a script that fills the box on the Master Page with a variable address. The addresses are stored in a snippet.

1. Right-click the box and choose **New Script** from the shortcut menu. The **Script editor** appears. The selector of the script - the ID of the box - is added automatically.

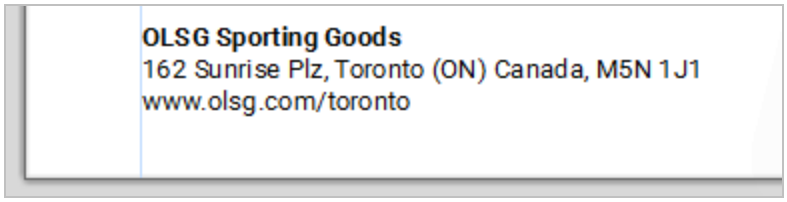


2. Change the script name to **Change Company Address**.
3. Add the following script.

```
var office = loadhtml("snippets/olsg_address.html", "#" +
    record.fields.StateAbbrev);
results.html(office);
```

   In **Preview** mode and when output is generated, this script will read the value of the **StateAbbrev** field in the current record. It will load the paragraph that contains the address of the office in that state value from the snippet and insert that in the template.
4. Click **OK**. The script appears on the **Script** pane in the Standard scripts folder.

5. Switch to Preview mode to see if the correct address appears in the template.



The finishing touch

1. Adding page numbers

Invoices consist of a variable number of pages, so they need to have page numbers. This exercise shows how to add page numbers to a template.

1. Open the **Master Page**. The page number should appear on every page, so the best place to put it in is the Master Page.
2. Add a Positioned Box and put it at the bottom right, next to the box for the company address.
3. In the box, type **Page** (followed by a space).
4. From the menu, select **Insert > Special Character > Markers > Page Number**.
5. Type a space, then **of**, followed by a space.
6. From the menu, select **Insert > Special Character > Markers > Page Count**.

7. Click the **Align Right** icon on the toolbar to align this text to the right of the box.
8. Save the Master Page.
9. Reopen the Invoice template.
10. Toggle to Preview mode to see if the page numbers are inserted correctly.

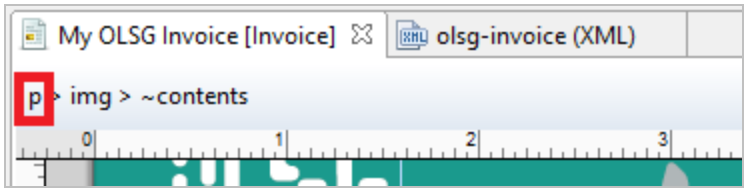
Tip: To change the page number format to, for example, uppercase Roman numerals (I, II, III) or lowercase letters (a, b, c), right-click the Invoice template on the **Resources** pane and select **Numbering**.

2. Using an image as a whitespace element

In this exercise you will add an image to the invoice to be used as a white space filler.

White space on an invoice can be very well used to include an advert or announcement. If there is white space on some invoices, but not on others, you may insert an image in the template and tag it as a **whitespace** element. A whitespace element will only be included in the output if it fits in the available white space on the current page.

1. In Design mode, add one new paragraph below the paragraph Terms and Conditions: place the cursor at the end of that paragraph and press **Enter**.
2. Import the image: locate the images folder provided with this tutorial, and drag **olsg_whitespace_hockey_M.pdf** from that folder to the **Images** folder in the **Resources** pane.
3. Place the image in the template: drag **olsg_whitespace_hockey_M.pdf** from the **Images** folder to the empty paragraph. The image is inserted at the cursor position and anchored to the text. This means that new lines added above the image will push the image down.
4. Select the image in the document. When selected, images can be resized by dragging the resize handles. Resize the image if needed. Keep the **Shift** key pressed while resizing to scale the image proportionally.
   Note that the **Source** field in the **Attributes** pane shows the path to the image e.g. “images/olsg_whitespace_hockey_M.pdf”. In this path, "images" refers to the Images folder in the Resources pane.
5. The breadcrumbs show that the selected image is inside a paragraph. Click the **p** in the breadcrumbs to select the paragraph that contains the image.



   Note: Only top-level elements can be used as whitespace element. In the breadcrumbs, top-level elements come first.

6. On the **Attributes** pane, check the **Whitespace** option.
7. Switch to Preview mode and browse the records to see that the invoice only shows the image if it has space for it.

Note: If you want a larger distance between the text of the invoice and the whitespace filler image, **do not** add a paragraph; that could result in an extra page. Instead, set a top margin for the image:

1. Right-click the image and choose **Image...**
2. On the **Spacing** tab, in **Margins**, uncheck **All sides** and enter a top margin (for example, 10px).

3. Switching images dynamically

In this exercise you will learn how to write a **script** that replaces one image by another, depending on values in the record set.

The Marketing department has provided four images:

- The hockey advert should only be presented to people who like hockey; men should get the green version (**olsg_whitespace_hockey_M.pdf**) and women the purple version (**olsg_whitespace_hockey_F.pdf**).
- Recipients whose favorite hobby is to ski should get the skiing advert; **olsg_whitespace_skis_M.pdf** is for men, **olsg_whitespace_skis_F.pdf** for women.

1. Import the extra images: locate the images folder provided with this tutorial and drag the images that are needed to the Images folder in the Resources pane.
2. Click the image in the template to select it and give it an ID: on the **Attributes** pane, in the **ID** field, type **olsg-banner**.
3. Right-click the image and choose **New Script** from the shortcut menu. The Script editor appears. The selector of the script is added automatically.
4. Change the script name to **Switch Banner**.
5. Add the following script:

```
if (record.fields.FavHobby == "skate") {
    results.attr("src", "images/olsg_whitespace_hockey_" +
        record.fields.Gender+".pdf");
} else {
    results.attr("src", "images/olsg_whitespace_skis_" +
        record.fields.Gender+".pdf");
}
```

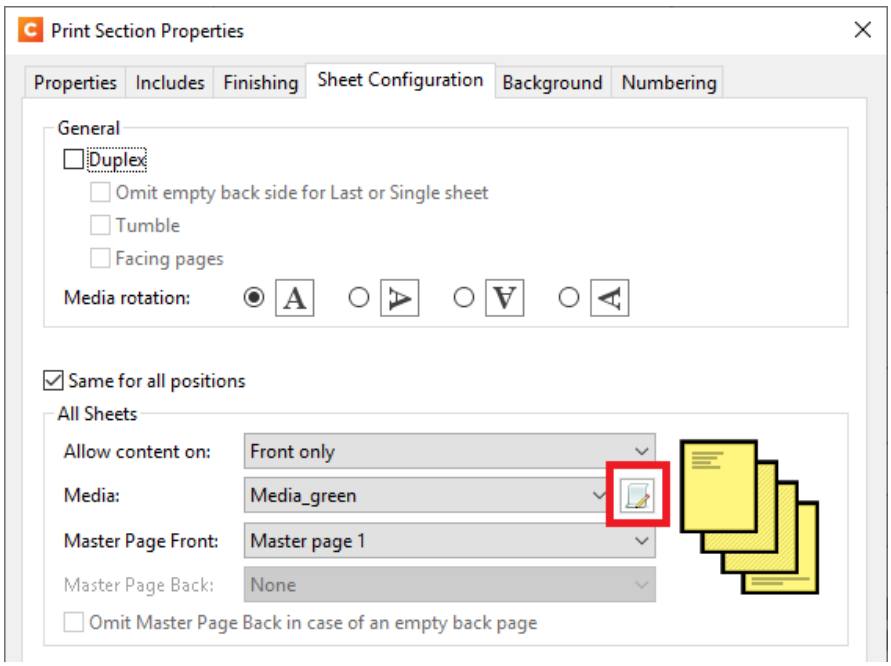
   Scripts start by performing a **query**: they scan the template to find all elements that match the specified **selector**. This script will look for an element that has the ID `olsg-banner`. Then, following the conditions, it attributes a different source to the **results** of that query – which in this case is the image. The "**F**" or "**M**" at the end of the images' name can be returned dynamically by the **Gender** data field

6. Click **OK** to close the dialog and toggle to Preview mode to see if the script works as expected.

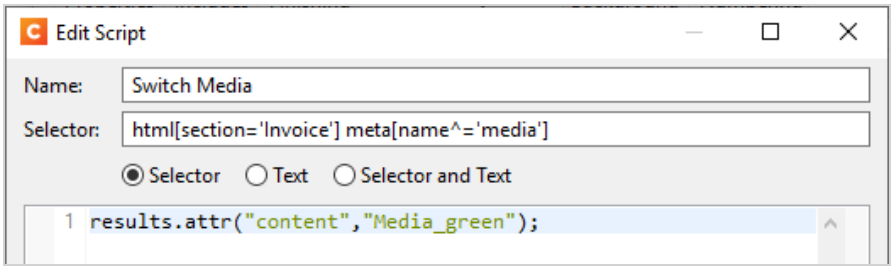
4. Switching Media dynamically

The Marketing department has also provided a new letterhead, to match the images that are colored purple. In this exercise you will use that as a second Media and create a script to switch the Media based on a data value.

1. Import the new letterhead: locate the images folder provided with this tutorial and drag **olsg-letterhead-letter-purple.PDF** to the **Images** folder in the **Resources** pane.
2. Right-click the **Media** folder in the **Resources** pane and select **New Media**. The **Media Properties** dialog appears.
3. On the **Virtual Stationery** tab, check the **Front** option and click the **Select Image** icon.
4. Select **olsg-letterhead-letter-purple.PDF** from the list of resources and click **OK**.
5. Click **OK** to create the new Media entry.
6. Right-click the **Media 1** entry and select **Rename**. Rename the Media to **Media_green**.
7. Rename the second Media entry to **Media_purple**.
8. Open the Sheet Configuration dialog: right-click the invoice and select **Sheet Configuration**.
9. Click the **Edit Script** icon.



10. The Edit Script dialog appears. Change the name of the script to **Switch Media**.



11. Replace the existing script with the following code. It contains a condition to toggle the Media when the hobby of the recipient is skating:

```
if (record.fields.FavHobby == "skate") {
    results.attr("content","Media_green");
} else {
    results.attr("content","Media_purple");
}
```

12. Click **OK** to create the script entry. The entry is added to the Scripts pane.
13. Click **OK** to close the Sheet Configuration dialog.
14. Toggle to the Preview mode and browse through the records to see if the script works as expected.

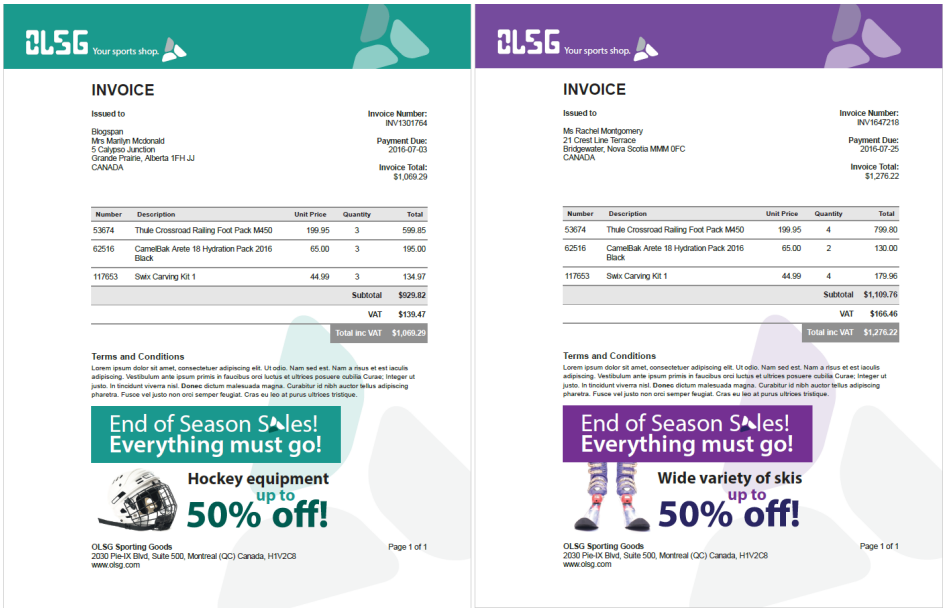
5. Keeping elements together

While browsing the records in Preview mode you may have noticed that sometimes the heading "Terms and conditions" and the paragraph beneath it are on different pages. These two elements should stick together. This can be arranged using the CSS `page-break-after` property.

1. Choose **Edit > Stylesheets...** to make the Stylesheets dialog appear.
2. From the **Context** drop-down, choose **Print**.
3. Click the **Plus** icon to create a new style rule.
4. In the dialog that appears, type **h3** in the **Selector** field. This means that the rule will apply to all level 3 headings.
5. Click the **Formats** tab.
6. In the **Breaks** section, from the **After** drop-down, choose **avoid**.
7. Reduce the space between the heading and the paragraph: on the **Spacing** tab, in **Margins**, uncheck **All sides** and add a margin of **2px** to the bottom of the heading.
8. Click **OK** to close the Rule editor.

9. Click **Save** to close the Stylesheet editor and go to Preview mode to check the effect of the rules.

The invoice is now ready. The result should look similar to this:

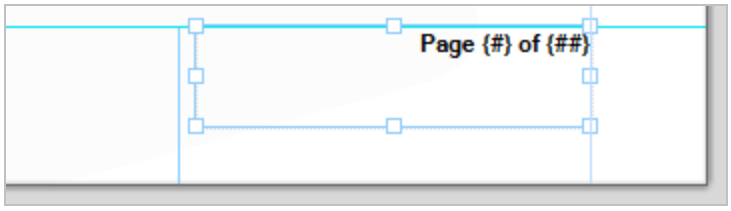


The finishing touch

1. Adding page numbers

Invoices consist of a variable number of pages, so they need to have page numbers. This exercise shows how to add page numbers to a template.

1. Open the **Master Page**. The page number should appear on every page, so the best place to put it in is the Master Page.
2. Add a Positioned Box and put it at the bottom right, next to the box for the company address.
3. In the box, type **Page** (followed by a space).
4. From the menu, select **Insert > Special Character > Markers > Page Number**.
5. Type a space, then **of**, followed by a space.
6. From the menu, select **Insert > Special Character > Markers > Page Count**.



7. Click the **Align Right** icon on the toolbar to align this text to the right of the box.
8. Save the Master Page.
9. Reopen the Invoice template.
10. Toggle to Preview mode to see if the page numbers are inserted correctly.

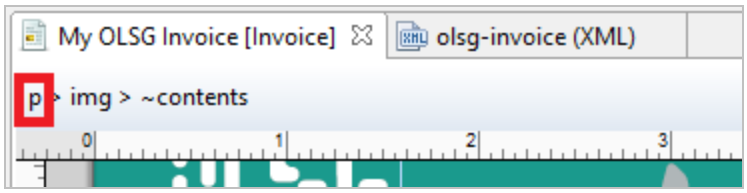
Tip: To change the page number format to, for example, uppercase Roman numerals (I, II, III) or lowercase letters (a, b, c), right-click the Invoice template on the **Resources** pane and select **Numbering**.

2. Using an image as a whitespace element

In this exercise you will add an image to the invoice to be used as a white space filler.

White space on an invoice can be very well used to include an advert or announcement. If there is white space on some invoices, but not on others, you may insert an image in the template and tag it as a **whitespace** element. A whitespace element will only be included in the output if it fits in the available white space on the current page.

1. In Design mode, add one new paragraph below the paragraph Terms and Conditions: place the cursor at the end of that paragraph and press **Enter**.
2. Import the image: locate the images folder provided with this tutorial, and drag **olsg_whitespace_hockey_M.pdf** from that folder to the **Images** folder in the **Resources** pane.
3. Place the image in the template: drag **olsg_whitespace_hockey_M.pdf** from the **Images** folder to the empty paragraph. The image is inserted at the cursor position and anchored to the text. This means that new lines added above the image will push the image down.
4. Select the image in the document. When selected, images can be resized by dragging the resize handles. Resize the image if needed. Keep the **Shift** key pressed while resizing to scale the image proportionally.
   Note that the **Source** field in the **Attributes** pane shows the path to the image e.g. “images/olsg_whitespace_hockey_M.pdf”. In this path, "images" refers to the Images folder in the Resources pane.
5. The breadcrumbs show that the selected image is inside a paragraph. Click the **p** in the breadcrumbs to select the paragraph that contains the image.



> **Note:** Only top-level elements can be used as whitespace element. In the breadcrumbs, top-level elements come first.

6. On the **Attributes** pane, check the **Whitespace** option.
7. Switch to Preview mode and browse the records to see that the invoice only shows the image if it has space for it.

> **Note:** If you want a larger distance between the text of the invoice and the whitespace filler image, **do not** add a paragraph; that could result in an extra page. Instead, set a top margin for the image:
>
> 1. Right-click the image and choose **Image...**
> 2. On the **Spacing** tab, in **Margins**, uncheck **All sides** and enter a top margin (for example, 10px).

3. Switching images dynamically

In this exercise you will learn how to write a **script** that replaces one image by another, depending on values in the record set.

The Marketing department has provided four images:

- The hockey advert should only be presented to people who like hockey; men should get the green version (**olsg_whitespace_hockey_M.pdf**) and women the purple version (**olsg_whitespace_hockey_F.pdf**).
- Recipients whose favorite hobby is to ski should get the skiing advert; **olsg_whitespace_skis_M.pdf** is for men, **olsg_whitespace_skis_F.pdf** for women.

1. Import the extra images: locate the images folder provided with this tutorial and drag the images that are needed to the Images folder in the Resources pane.
2. Click the image in the template to select it and give it an ID: on the **Attributes** pane, in the **ID** field, type **olsg-banner**.
3. Right-click the image and choose **New Script** from the shortcut menu. The Script editor appears. The selector of the script is added automatically.
4. Change the script name to **Switch Banner**.

5. Add the following script:

```
if (record.fields.FavHobby == "skate") {
    results.attr("src", "images/olsg_whitespace_hockey_" +
        record.fields.Gender+".pdf");
} else {
    results.attr("src", "images/olsg_whitespace_skis_" +
        record.fields.Gender+".pdf");
}
```

   Scripts start by performing a **query**: they scan the template to find all elements that match the specified **selector**. This script will look for an element that has the ID `olsg-banner`. Then, following the conditions, it attributes a different source to the **results** of that query – which in this case is the image. The "**F**" or "**M**" at the end of the images' name can be returned dynamically by the **Gender** data field

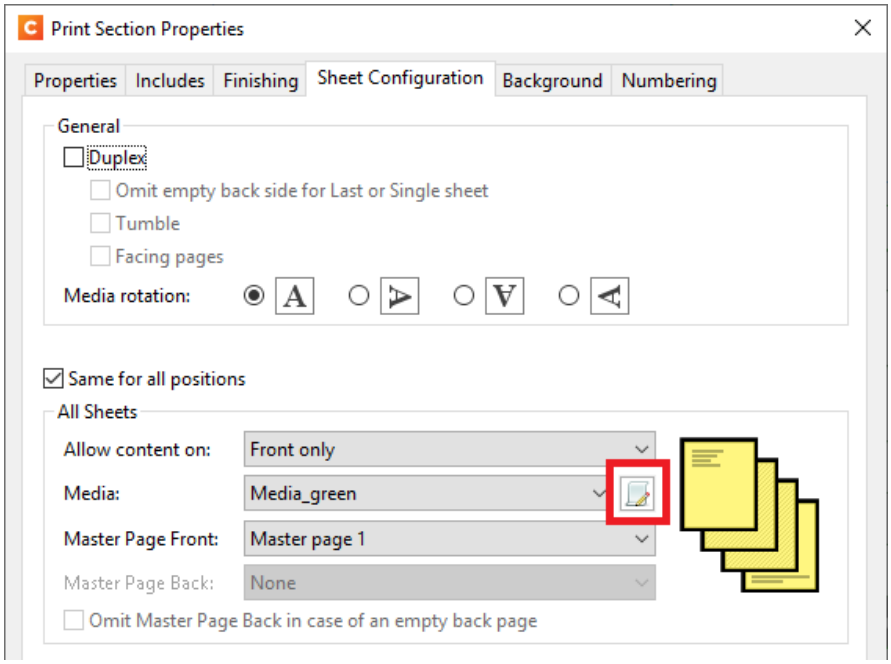
6. Click **OK** to close the dialog and toggle to Preview mode to see if the script works as expected.

4. Switching Media dynamically

The Marketing department has also provided a new letterhead, to match the images that are colored purple. In this exercise you will use that as a second Media and create a script to switch the Media based on a data value.

1. Import the new letterhead: locate the images folder provided with this tutorial and drag **olsg-letterhead-letter-purple.PDF** to the **Images** folder in the **Resources** pane.
2. Right-click the **Media** folder in the **Resources** pane and select **New Media**. The **Media Properties** dialog appears.
3. On the **Virtual Stationery** tab, check the **Front** option and click the **Select Image** icon.
4. Select **olsg-letterhead-letter-purple.PDF** from the list of resources and click **OK**.
5. Click **OK** to create the new Media entry.
6. Right-click the **Media 1** entry and select **Rename**. Rename the Media to **Media_green**.
7. Rename the second Media entry to **Media_purple**.
8. Open the Sheet Configuration dialog: right-click the invoice and select **Sheet Configuration**.

9. Click the **Edit Script** icon.



10. The Edit Script dialog appears. Change the name of the script to **Switch Media**.

11. Replace the existing script with the following code. It contains a condition to toggle the Media when the hobby of the recipient is skating:

```
if (record.fields.FavHobby == "skate") {
    results.attr("content","Media_green");
} else {
    results.attr("content","Media_purple");
}
```

12. Click **OK** to create the script entry. The entry is added to the Scripts pane.
13. Click **OK** to close the Sheet Configuration dialog.
14. Toggle to the Preview mode and browse through the records to see if the script works as expected.

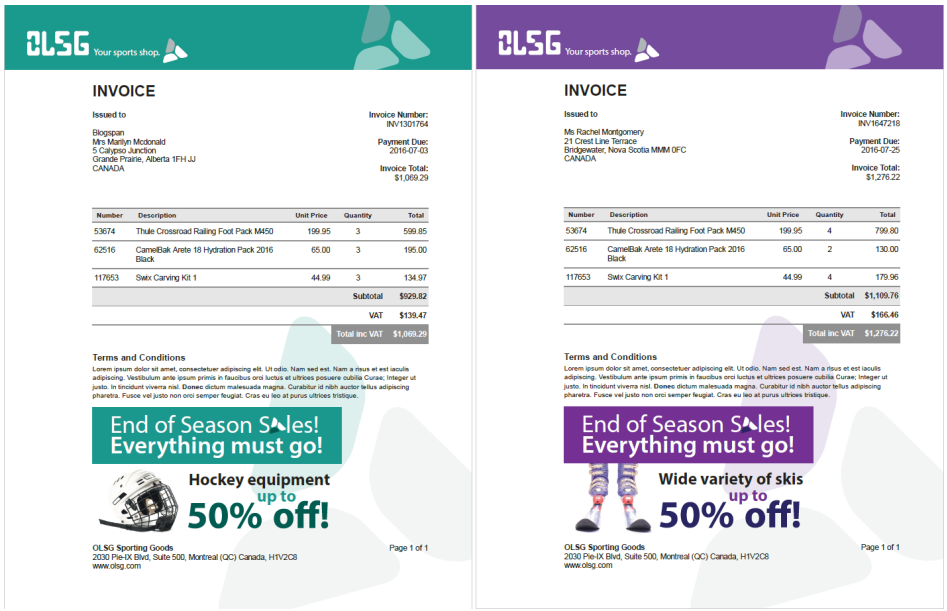
5. Keeping elements together

While browsing the records in Preview mode you may have noticed that sometimes the heading "Terms and conditions" and the paragraph beneath it are on different pages. These

two elements should stick together. This can be arranged using the CSS `page-break-after` property.

1. Choose **Edit > Stylesheets...** to make the Stylesheets dialog appear.
2. From the **Context** drop-down, choose **Print**.
3. Click the **Plus** icon to create a new style rule.
4. In the dialog that appears, type **h3** in the **Selector** field. This means that the rule will apply to all level 3 headings.
5. Click the **Formats** tab.
6. In the **Breaks** section, from the **After** drop-down, choose **avoid**.
7. Reduce the space between the heading and the paragraph: on the **Spacing** tab, in **Margins**, uncheck **All sides** and add a margin of **2px** to the bottom of the heading.
8. Click **OK** to close the Rule editor.
9. Click **Save** to close the Stylesheet editor and go to Preview mode to check the effect of the rules.

The invoice is now ready. The result should look similar to this:



What's next?

The invoice template is now ready.

It can be printed from OL Connect Designer, with the data loaded from a file or database or from an open data mapping configuration.

But it is more likely that the template will be merged with data in an automated process that then distributes the output via email or sends it to a printer.

Designing a letter template

This tutorial introduces you to some of the basic functionality of OL Connect Designer. You will learn how to set up a personalized business letter by creating a new print document, add content from Microsoft Word, import image resources, load a CSV data file and set up variable content elements.

OL Connect Designer lets you create and design print documents, and also emails and web pages if you have the Professional or Enterprise edition. Every aspect of your document can be easily personalized thanks to the feature-rich and user-friendly interface.

The file format of the documents utilizes HTML, the main markup language for displaying web pages and e-mail messages. OL Connect Designer extends this with paginated documents intended for print. When exploring the user interface you will see several items that reveal this underlying technology. Knowing HTML is not a requirement to work with the application; the user interface includes the appropriate features for both novice and advanced users.

After having completed this walkthrough you'll have a basic knowledge of how to use OL Connect Designer, but there are more ways to do things and more things you can do. So, go on and explore more! To discover new ways and enhance your skills, please visit the OL Resource Center and browse the Online Help.

Note: A PDF version of this walkthrough is available for offline use: Designer_Letter_WT.pdf.

Note: The resources needed for this walkthrough can be downloaded from the Welcome screen in OL Connect Designer. Open the application, click New at the left, then select the Bundle Offer letter from the Online resources.

May 21, 2024

Riebesell, H F Jr
Francoise Rautenstrauch
42 N 16th St #62
Montreal, QC H2C 3A1

OL Mobile
2030 Pie-IX Blvd, Suite 500,
Montreal (QC) Canada, H1V2C8
www.olsg.com

Profit from our Bundle Offers

Dear Francoise,

Do you want to combine quality with the benefits OL Mobile can give you? With OL Mobile you can surf the internet, call and even watch Internet TV. OL Mobile now offers you a 3-in-1 bundle.

What's in the Bundle Offer?

Want to know what's in it for you? The bundle package includes **500 free minutes, 500 sms** and **500 MB bandwith** for surfing the internet for only an additional $5, automatically added to your invoice, monthly.

Would you like to take advantage of the Bundle Offer?

Are you interested in our all-in-1 package? Call us free of charge at *1800-112-2333*, or visit us at olmobile.com/bundleoffer. Here you can find all details and register and subscribe to the exclusive OL Mobile Bundle Offer.

If you have friends living close by who could be interested too, let them know! The OL Mobile Bundle Offer for anyone living in Montréal will be valid throughout 30 June 2017.

Looking forward to welcoming you as one of our preferred customers soon.

Best regards,

Pierre Probst
Senior Sales Representative

Creating a letter

This lesson introduces the concepts of sections, content and formatting. You will also learn how to navigate the content.

1. Creating a print template

Start by creating a new document for print.

1. Launch the Designer by double-clicking the application icon.
2. In the **Welcome** screen, click **Create a New Template** and choose **Word-based Print**. Or choose **File > New > Template > Word-based Print Template** and click **Next**. The **New Word-based Print Template** dialog is shown.

   The dialog lets you enter parameters for the new document, which includes its page size, the page margins and the number of sections. A section contains one or more pages. Text will flow automatically from one page to the next.
3. Click **Browse** icon next to the **path:** field to select a Word document. The file explorer opens. Select the sample file **letter-text.docx** from your local drive.

   > **Note:** The Word document **letter-text.docx** is included in the resources that you have already saved to your local drive.
4. Click **Next**. The initial page size, margins, and orientation are taken from the original document settings and displayed in the dialog window that appears. You can adjust them on that dialog window.
5. Make the following settings:
   - **Margin Top**: 1,5in
   - **Margin Right**, **Bottom** and **Left**: 0,75in

   The values may be converted into another measurement unit, depending on your settings (Window > Preferences > Print > Measurements).

   > **Note:** After the template is created, you can change the size of images and text boxes, and adjust the page size and margins.
6. Click **Finish**.
   The new document is shown in the workspace. A Print context is created with the specified number of sections in it. For each page in the document, one page is created in the Print section. A separate word.css file is created in the Stylesheets. The css file is linked to the active section.
7. To save the template, click **File > Save As**. The **Save Template** dialog box displays.
8. Navigate to a folder (preferably, the folder where you unzipped the sample files), enter **My OL Mobile letter.OL-template** in the **File name** field and then click **Save**.

2. Formatting text

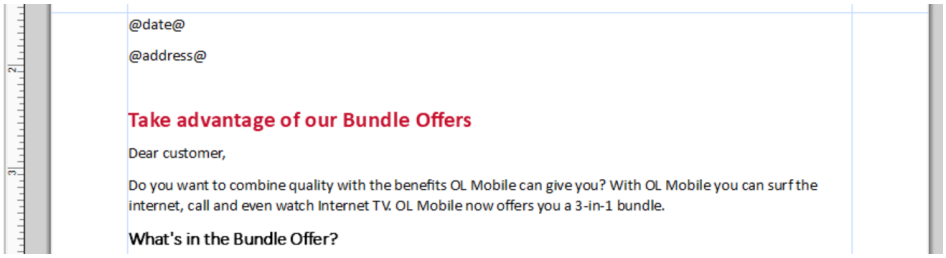
OL Connect Designer has many text editing and formatting capabilities. Applying formatting to selected text is referred to as **local** or **inline formatting**. Formatting could also

be handled via style sheets, as will be demonstrated later.

1. Select the text of the subject line: "Take advantage of our Bundle Offers".
2. Common text properties (e.g. font, size, color) can be set via the options in the **Formatting** toolbar.
   Click the **B** icon on the toolbar to bold the text.
3. Another option is to format selected text using the **Text formatting** dialog. To open it, click **Format > Text...**.

   The dialog can also be invoked by pressing Ctrl + T or via the shortcut menu: right-click the selected text and choose **Text...**
4. Change the font size to: 16pt.
5. In the Text Formatting dialog, click the **Edit Color** icon (the small square next to the color field). The **Color Picker** dialog appears. Colors can be specified in CMYK, RGB and HEX color values.
6. Choose the following color values:
   - Cyan: 0, Magenta: 92, Yellow: 77, Black: 22
   - Color mode: CMYK (this will store the specified color using the CMYK values)
7. Click OK twice, to close the dialogs.

Note: The style changes are applied to the selected text. When you press **Enter** somewhere in this text, a new paragraph is created that inherits these text styles.



3. Exploring the document

This exercise shows how to explore the document and select elements on the page.

1. Take a look at the **Outline** pane. The **Outline** pane can be reached in the upper left area next to the main editor. In case the **Outline** pane isn't available, choose **Window > Show View > Outline**.
   The Outline pane shows a hierarchical tree structure of the pages and their elements and is useful to quickly select elements. It represents the HTML Document Object

Model or **DOM** - a term advanced HTML users will recognize.
In this example the Outline pane shows a series of paragraphs and two subheadings. The paragraphs are presented by the **<> p** entries in the tree and the subheadings are marked with **<> h2**. Other common HTML elements are: h1 through h6 (headings), img (image), table, tr (table rows), td (table data or table cells), b (bold), span (a piece of styled text).

2. Click a paragraph element in the outline. The main document window highlights that paragraph and shows a thin line around its contents. The paragraph is now selected.

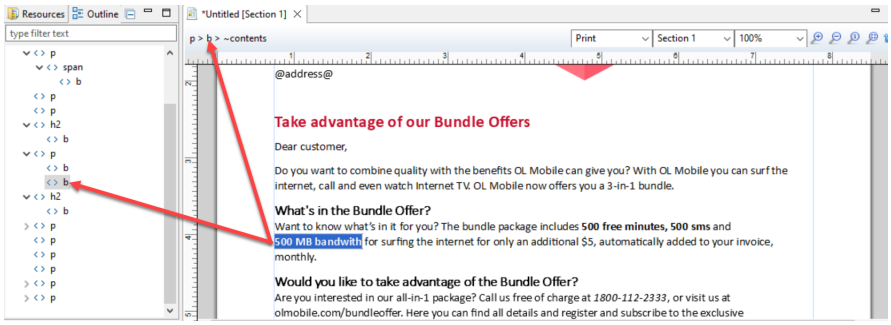
   Note that the same information is shown below the title of the document. This line is referred to as the **breadcrumbs**.

3. Use the arrow up and arrow down buttons to navigate through the elements in the outline.

4. Place the cursor in the text "500 MB bandwidth". Now the breadcrumbs read **p > b**. This signifies that the bold text is a child element of the paragraph.

5. Click the **b** in the breadcrumbs to select the bold text.

6. Click the **p** in the breadcrumbs to select the paragraph. Note that the element gets selected in the **Outline** pane too.



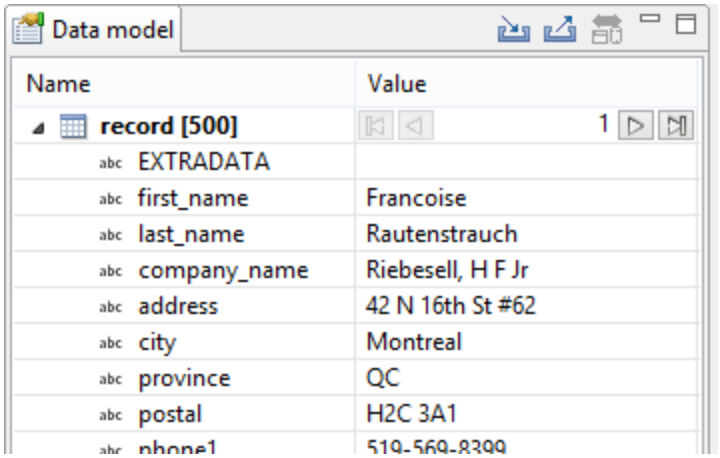
Adding personalization

To create a personalized document, you need information about the recipients: the first name, last name, email address, job title, etc. This chapter shows how to import a sample CSV data file and demonstrates various techniques to add data fields to the page and create dynamic content.

1. Loading a data file

A Comma Separated Values file (.csv) is a simple text format in which data are separated by commas or semicolons. When you load a CSV file, a wizard guides you through the process of importing the data and lets you set several import options.

1. Choose **File > Data > Open Data File...**. The **Open Data Sample File** dialog appears.
2. Select the sample **letter-data.csv** file and click **Open**. The wizard for loading CSV files is now shown.
3. Click **Next** to navigate to the second page of the wizard. Make the following settings:
    - **Separator**: Semicolon
    - Check the **First row contains field names** option.
4. Click **Finish**. The extracted data is shown in the **Data Model** pane at the right. If this pane isn't visible, choose **Window > Show View > Data Model**.



5. Use the browse buttons in the Value column to browse through the imported data.

Note: The Professional and Enterprise editions of OL Connect have more advanced options for importing data. The built-in DataMapper allows to create data mapping configurations to extract, manipulate and use data from a wide range of data sources, including line printer data and PDF documents. It is not available in OL Connect Desktop edition.

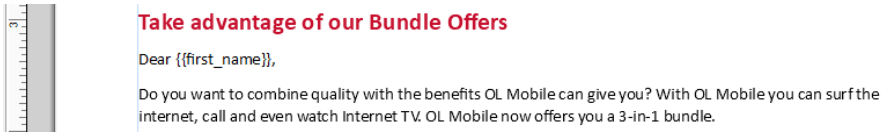
2. Adding a data field to the document

As soon as data have been imported, you are ready to start personalizing the document by adding data fields to the template. When you add a data field to the template, an *expression* is inserted at the cursor position. Expressions look like this:

{{fieldname}}

When the document is viewed in **Preview** mode (or when output is generated), the expression gets replaced with the value of the data field.

1. Remove the text “customer” in the salutation of the letter.
2. Select the **first_name** field in the **Data Model** view and drag the field after the text “Dear” in the document. On releasing the mouse button, the expression **{{first_name}}** is inserted at the cursor position.



3. Switch the view mode of the document from **Design** to **Preview** by clicking the **Preview** tab at the bottom of the workspace. This shows the merge result. Use the browse controls in the **Data Model** pane to scan through the records.

The expression **{{first_name}}** gets replaced with the value of the first_name data field in the current record. This is done by the **Handlebars** library which is integrated in OL Connect. For more information see the online help: Handlebars in OL Connect.

In previous versions, variable data used to be inserted in a template via *scripts* and *placeholders* (see the online help: Variable data in text: scripts and placeholders).
The advantage of expressions is that fewer scripts and sometimes no scripts at all are needed in a template.

3. Adding multiple data fields in a single action

Now you will learn how to add **multiple** data fields to the document in a single action. We will add the name and job title of the sales representative who is the sender of this letter, to the end of the body.

1. Toggle to the **Design** view of the document.
2. Place the cursor after the text “Best Regards” and insert a few new paragraphs.
3. Select the **rep_first**, **rep_last** and **rep_title** field in the **Data Model** pane by clicking each field while pressing the Ctrl key.
4. Drag the selected fields to the end of the document.
   An expression for each of the fields is added at the cursor position. The expressions are placed on a single line. The document shows the following expressions: {{rep_first}}{{rep_last}}{{rep_title}}.
5. Type a space between **{{rep_first}}** and **{{rep_last}}**.
6. Place your cursor between **{{rep_last}}** and **{{rep_title}}** and press Shift + Enter to put the job title on a new line.

A line break is added to the text. (In HTML this is the <br> element, as you can see in the **Source** view of the document.)

7. The expressions can be formatted via the **Format** menu and the icons on the toolbar. Select the expressions **{{rep_first}}** and **{{rep_last}}** and click the **Bold** icon on the formatting toolbar.
8. Toggle to the **Preview** mode and note that the value that replaces the expression is formatted just like the expression.

4. Creating an address block

The Address Block wizard makes it very easy to create address blocks that conform to country standards.

1. Add about 10 lines before the text "Profit from our Bundle Offers".
2. Select **Insert** > **Address Block** from the menu, or click the Address Block icon on the toolbar to open the wizard.
3. Under Address Format, select **Canada**.
4. Now match the following data fields and address fields:

Address field	Data field
First name	first_name
Last name	last_name
Function	rep_title
Company	company_name
Street	address
City	city
Region	province
Postal code	postal

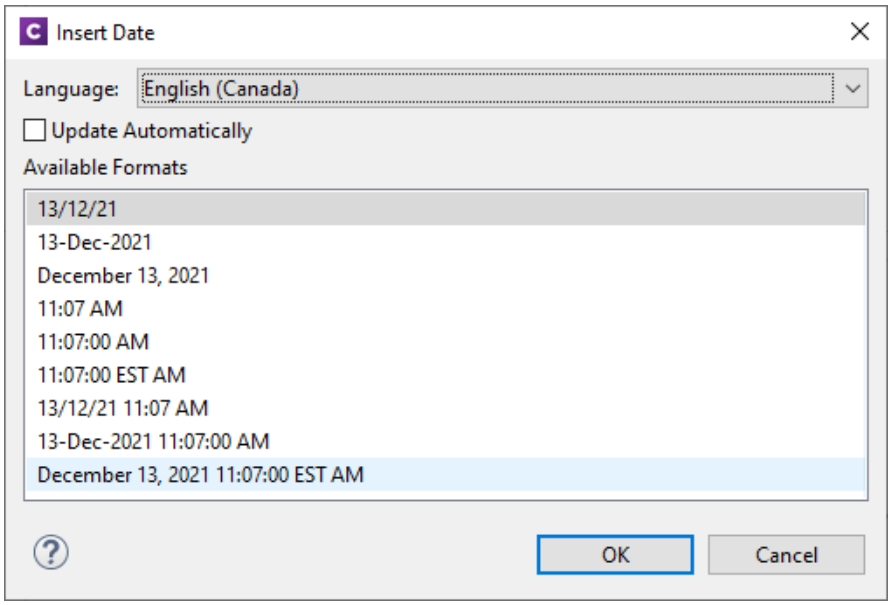
5. Click **OK**.
6. Toggle to the **Preview** mode and use the browse buttons of the **Data Model** pane to navigate through the records. Note that when some data are missing - the company name, for example - the remaining address lines automatically move up.

5. Adding a dynamic date

In Connect Designer you can quickly insert a date. When you are preparing a template today and finish it tomorrow, the date can be updated automatically, so you don't have to manually delete earlier dates.

To replace the "@date@" placeholder with a date:

1. Toggle to the **Design** mode and select the "@date@" placeholder at the beginning of the template.
2. Choose **Insert > Date...**. The **Insert Date** dialog appears.



3. Select a language to define the notation of the date.
4. Select the **Update Automatically** checkbox and choose the format of your preference.
5. Click **OK**. The expression that is added to the template returns today's date.
6. Toggle to the **Preview** mode to see the expression in action.

6. Creating a conditional paragraph

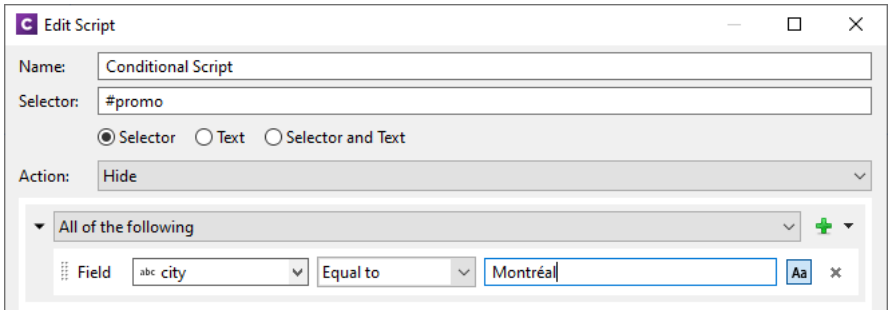
Now you will learn how to create conditional content using the **Make Conditional** script wizard. This wizard provides a user interface to define when a certain element (or elements) should be hidden or shown. In this case we will show a paragraph for recipients living in the city of Montréal and hide it for others.

1. Toggle to the **Design** mode and place the cursor in the paragraph that starts with the text: "If you have friends living close by". Click the **p** in the breadcrumbs. The paragraph is now selected.

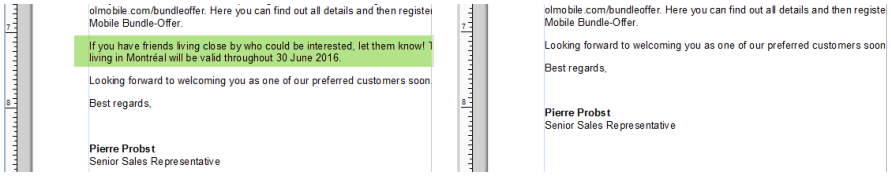
2. On the **Attributes** pane, type **promo** in the **ID** field.
3. Right-click this paragraph and select **Make Conditional** in the shortcut menu. The **Edit Script** script wizard appears.
   Note that the ID of the paragraph has become the **selector** of the script.

   Tip: In selectors, the # sign is used to refer to an ID. A dot (for example: .promo) refers to a class.

4. Make the following settings:
   - **Data field**: city
   - **Operator**: Equal to
   - **Value**: Montreal



5. Enter a meaningful name for the script and click **OK** to save the script.
6. Toggle to **Preview** mode and browse through the records. The paragraph is shown for residents of Montreal and is hidden for others. Note that the text below this paragraph moves up when the content is not shown.



Tip: The same technique can be used to show and hide images.

Working with images

Typically documents do not solely contain text. Often resources like logos and background images are added to complete the document. In this section you will learn how to work with images.

1. Importing and inserting images in the template

In this exercise you will learn how to import images and place them in your document. Images can be static elements or they can be dynamically replaced by scripts based on a data field.

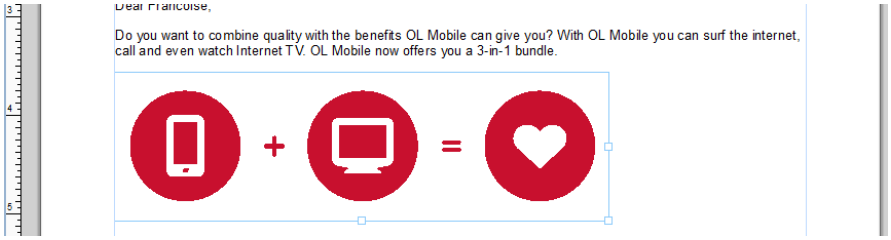
1. Take a look at the **Resources** pane at the top left. In case the **Resources** pane is not shown, choose **Window > Show View > Resources**.

   The **Resources** pane shows files and other resources belonging to the active document. Files added to the **Resources** pane travel with the document when it is copied, shared via email or sent to the OL Connect Workflow component. The **Resources** pane is not limited to image resources; a document may include custom fonts, style sheets (CSS files), snippets, etc.

2. Locate the **Images** folder provided with this tutorial and copy **promo_banner.pdf** to the **Images** folder in the **Resources** pane.

3. Enter an empty paragraph below the paragraph ending with "OL Mobile now offers you a 3-in-1 bundle."

4. Drag **promo_banner.pdf** from the **Images** folder to the new line. The image is inserted at the cursor position and anchored to the text. Adding new lines above the image will push the image down.



5. Select the image in the document and note that the **Source** field in the **Attributes** pane shows the path to the image: "images/promo_banner.pdf". In this path, **images** refers to the **Images** folder in the **Resources** pane.

   When selected, images can be resized by dragging the resize handles.

   > **Tip:** Keep the Shift key pressed while resizing to proportionally scale the image.

2. Adding a dynamic image

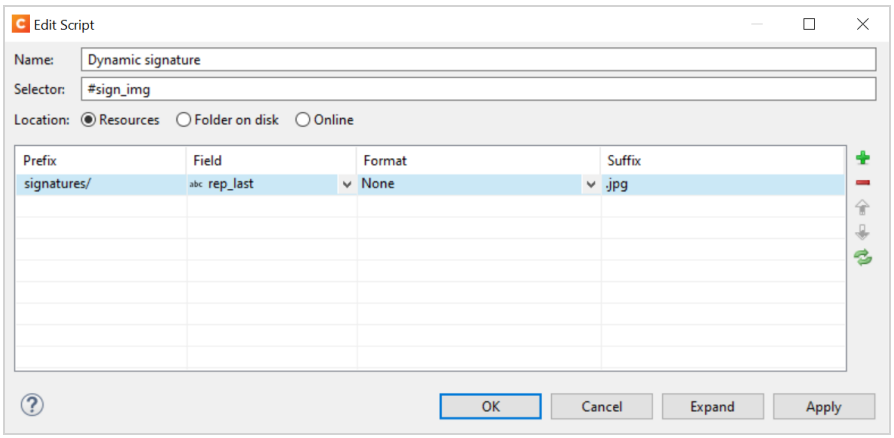
In this exercise you'll learn how to create a variable signature by using the **Dynamic Image script** wizard. The **Dynamic Image** wizard provides a user interface to compose the path to image files using values from a data field. (The same could be achieved using plain script, but that requires knowledge of JavaScript.)

1. Import the images: locate the images folder provided with this tutorial and drag the signatures folder to the **Images** folder in the **Resources** pane.
   The signatures folder contains three signature images: **Probst.jpg**, **van Houtten.jpg**, and **van Buel.jpg**. The value of the data field **rep_last** is equal to the name of one of these images.
2. Make sure there is an empty paragraph between "Best regards," and the name of the sales representative.
3. Drag the **Probst.jpg** file from the **Signatures** folder in the **Images** folder on the **Resources** pane to the empty paragraph. The image is placed inline and will flow with the text when paragraphs are added or removed (like with the conditional paragraph).
4. Click the image to select it. Its resize handles become active. Make the image a bit smaller (e.g. 75% of its original size).

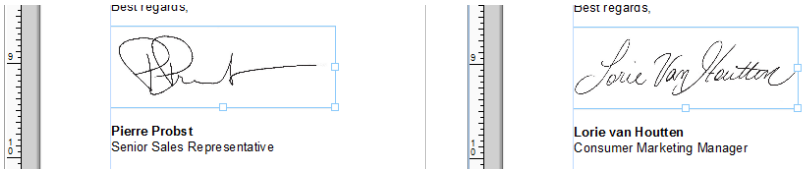
5. Right-click the image to show the shortcut menu. Choose **Dynamic Image...**. The **Dynamic Image** wizard appears.

   This wizard provides a simple way to replace images based on the value of a data field. In this case we are going to use the value of the **rep_last** data field to concatenate the path to the signature images. The three signature files already have the appropriate name.

   On invoking the **Dynamic Image** wizard the path to the current image file is placed in the **Prefix** field and the file extension is added to the **Suffix** field.



6. Select **rep_last** in the drop down of the **Field** column. On previewing and outputting the document the data is concatenated with the prefix and suffix data, which results in the path to the respective signature files.
7. Enter a meaningful name for the script and click **OK** to save the changes.
8. Toggle to **Preview** mode and browse through the records. Note that the signature image is replaced and matches the respective account manager.



Note: In the exercise above, we used the **Dynamic Image** wizard to create a dynamic image based on a data field. A Dynamic Image script writes the path of the image to the **src** attribute of the respective image.

Position the image

To reduce the distance between the image and the name of the sales representative, set the margins Bottom and left to 5px and -5px, respectively.

If you have used the Dynamic Image wizard, an image is already inserted in the template. In this case, you can change the margins as follows:

1. Right-click the image and select **Image...** from the shortcut menu. The **Image properties** dialog appears.
2. Click the **Spacing** tab.
3. Uncheck **All sides** in the **Margin** section.
4. Enter **-5mm** in the **Bottom** field. This reduces the whitespace below the bounding box of the image.
5. Enter **-5mm** in the **Left** margin field to make the image 'hang' outside the text flow.

3. Adding a Virtual Stationery background

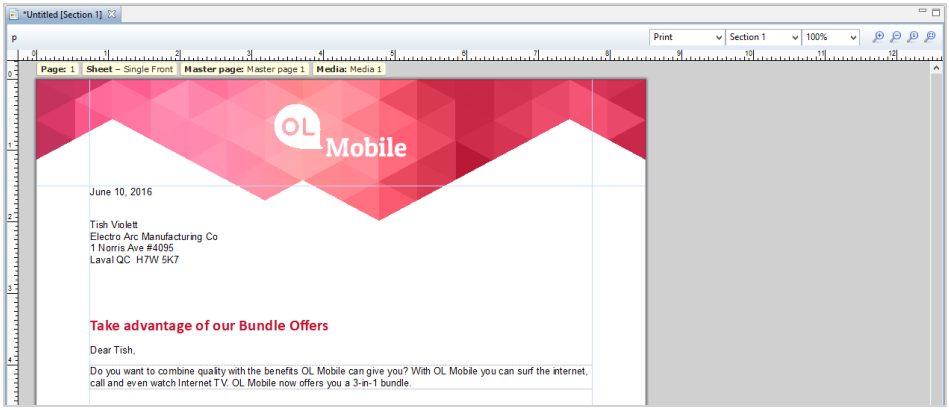
In this exercise you will learn how to set a PDF image as a background to emulate pre-printed paper. By specifying a so called virtual stationery image you will be able to preview

the pre-printed media and position the content in the correct position. The image will be repeated for all pages that use this media.

To set a PDF as media:

1. First import the PDF: locate the file **letterhead.pdf** that is provided with this tutorial and drag it to the **Images** folder on the **Resources** pane.
2. Drag **letterhead.pdf** to the **Media 1** entry in the **Media** folder on the **Resources** pane. The first page in the PDF will be set as the image for the front side of the media.

After adding the image to the media you will see that the image appears in the background of the page(s).



Alternatively you could open the **Media properties** dialog by right-clicking the **Media 1** entry and selecting **Virtual Stationery** from the shortcut menu. This dialog allows you to select the image for the front and back of the media and specify the position of the image.

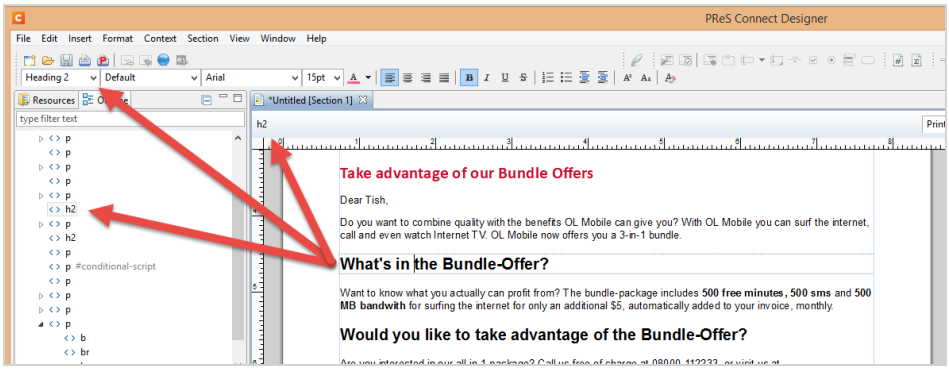
> **Note:** By default media backgrounds are omitted from the output (because in the end, the pre-printed paper will have this imagery). In case you are printing on blank paper you could print this background image together with the document by selecting the **Print Virtual Stationery** option in the **Advanced** section of the **Print Wizard** interface.

Working with style sheets

You've learned how to navigate the content and how to apply formatting to selected text. The **Outline** pane revealed two subheadings that inherited the Heading 2 style from Microsoft Word. Both the **Outline** pane and the **breadcrumbs** show the underlying HTML element. These Heading 2 subheadings are identified by the <h2> element.

To change these headings into something else (a paragraph, or a heading of another level) you can use the **Formatting** toolbar (see the picture) or the menu: **Format > Paragraph**

Format.



In order to change the way a heading or paragraph looks, you could select its text and use the toolbar buttons or open the **Formatting** dialog: choose **Format > Text...** This is called *local formatting*.

However, instead of formatting elements locally, it is recommended to use **Cascading Style Sheets (CSS)**.
You can add a style rule to a style sheet to format several elements at the same time. This may save you a lot of time. Style rules ensure that formatting is applied consistently.
In OL Connect Designer, style rules can be made using the **Stylesheets** dialog.

1. Creating a style rule

In this exercise you will learn how to change the formatting of the level 2 headings with CSS, via the **Stylesheets** dialog.

1. Choose **Edit > Stylesheets...**, to make the **Stylesheets** dialog appear.
2. The selected context is **Global**. Style rules created in this context will apply to all context types. Use the drop-down to select the **Print** context. Style rules created for a specific context (in this case, the **Print** context) will only be applied to sections in that context.
3. Click the green **Plus** icon to create a new style rule.
4. Enter **h2** in the **Selector** field. The selector defines to which elements in the document the rule applies. By stating **h2** the rule applies to all level 2 subheadings.

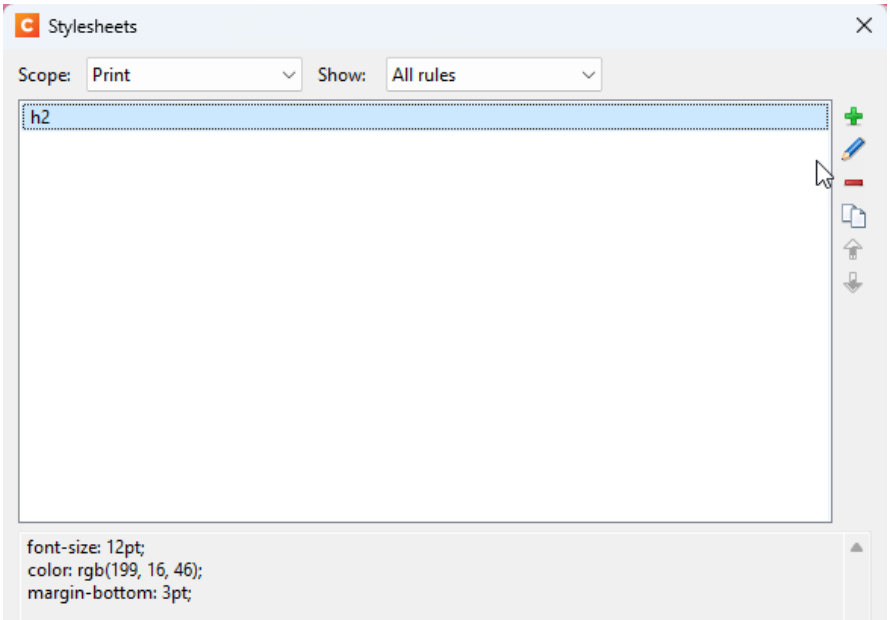
   The **Selector** field in the **New Rule** dialog accepts any standard CSS selector:

   - an HTML tag (for example, type p to select all paragraphs)
   - a class (for example: **.burgundy**)
   - an ID (for example: **#intro**)

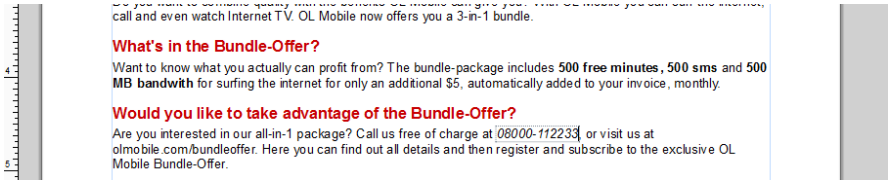
- a combination of CSS selectors (for example: **div#intro p.burgundy** selects all paragraphs with the class burgundy in a that has the ID intro); see [CSS Selectors on MDN](#).

> **Note:** In an HTML file, each ID should be unique. This means that a particular ID should be used only once in each section. A class can be added to multiple elements, even if those elements are of different types.

5. Make the following changes to the style properties on the **Type** tab:
   - **Font size**: 12pt
   - **Color**: Click the **Color Picker** icon next to the **Color** field and enter the following values (or use the eyedropper tool to pick the color of the subject line in the letter):
     CMYK: Cyan 0, Magenta 92, Yellow, 77, Black 22
6. Click **OK** to close the **Color Picker** dialog and go to the **Spacing** tab.
7. Make the following settings in the **Margin** section:
   - **All sides**: uncheck
   - **Margin bottom**: 3pt
8. Click **OK** to create the style rule.
   Note that the lower part of the **Stylesheets** dialog shows the style declarations for the selector in the upper part of the dialog.



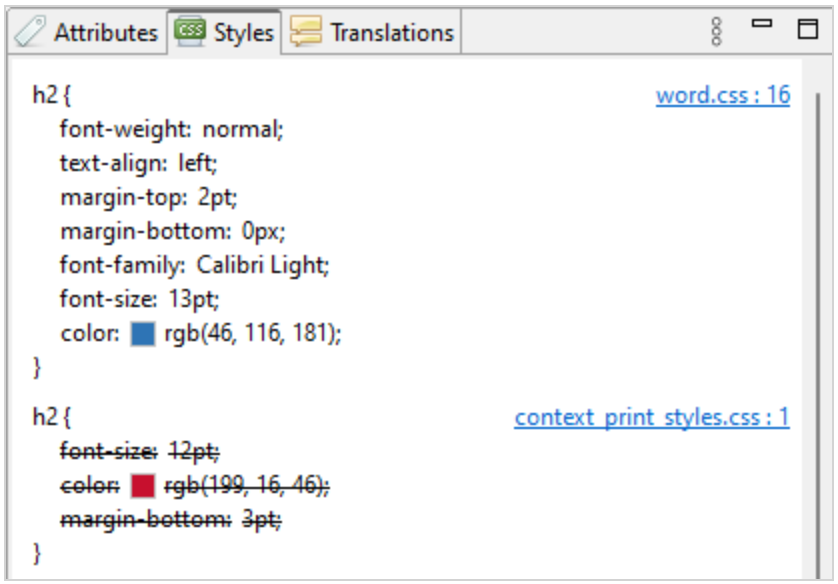
9. Click **Save** to store the rule with the template. The properties should now be applied to the subheadings....



10. ... but they are not.

    Let's take a look at the Styles pane, at the right, to see what happened. Place the cursor in one of the headings and click the **Styles** pane at the top right.

    Note: The **Styles** pane also shows a link to where the style rule is located, e.g. context_print_styles.css. You could click the link to open the CSS file in the workspace and edit it directly.



    As you can see, the style rules we just created got replaced with the style rules that were imported with the Word document.

11. So let's remove those imported style rules.

a. Choose **Edit > Stylesheets...**.

b. In the **Stylesheets** dialog, use the drop-down to select the **Word** context.

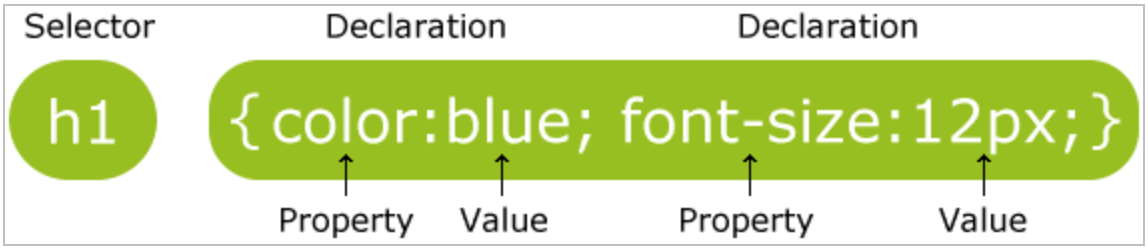
c. Click **h2**, then click the Delete icon.

b. Close the Stylesheets dialog. In Preview mode, the level 2 headings now look the way we want them to look.

2. Editing a style sheet

As you've seen in the **Stylesheets** dialog, a rule consists of a **selector** and a set of **declarations**.

The selector points to the HTML element you want to style.
The style-set contains one or more declarations, each ending with a semicolon. A declaration includes a **property name** and a **value**, separated by a colon.



In this exercise you will change a CSS file directly, instead of via the **Stylesheets** dialog. As you gain more knowledge of CSS, this option will become increasingly convenient.

This letter is designed to fit on one page. But in a longer letter, the signature image and the name of the representative might sometimes end up on different pages. These elements should stick together. This can be arranged using the CSS **page-break-after** property.

1. Select the paragraph that contains the name and title of the representative: place the cursor in the paragraph and then click on **p** in the breadcrumbs.
2. Give the paragraph an ID: on the **Attributes** pane, type **rep** in the **ID** field. This ID will be used as the selector of the style rule.
3. On the **Resources** pane, expand the **Stylesheets** folder and double-click **context_print_styles.css**. The file opens in the workspace. As you can see, it already contains the selectors and style rules that were added via the **Stylesheets** dialog.
4. Add the following style rule:

```
#rep {
        page-break-before: avoid;
}
```

> **Tip:** Type Ctrl + Space to get suggestions. As you start typing, the list of suggestions will get shorter. Use the arrow keys to select a suggestion and press Enter to insert it in the style sheet.

5. Save the file: press Ctrl + S.

Using Master Pages

Each new Print template automatically gets one Master Page. A Master Page is a layer between the Media and the pages in Print sections. Master Pages are meant to contain headers, footers and all other elements which aren't part of the text flow. With Master Pages you can make elements appear on multiple pages, or only on the first or last page of the section. Unlike the Media, Master pages are printed when the template is merged with a record set.

1. Adding an element to a Master Page

This exercise shows how to add an element to a Master Page.

1. Open Master page 1: on the **Resources** pane, open the **Master pages** folder and double-click **Master page 1**. The Master Page is opened in the workspace.
2. Add a Positioned Box by clicking the **Insert Positioned Box** icon on the toolbar.

   > **Tip:** Elements on a Master page aren't part of the text flow, so they need to stay in place. Therefore they have to have an absolute position, or be located in another element that has an absolute position.

3. Position the box to the top right of the page, in a way that it doesn't overlap with the letterhead. Clicking on one of the borders of the box allows you to drag it. Click on one of the small squares to resize it. Alternatively, you could type its top, left, width and height on the **Attributes** pane, for example:
   Y-offset: 1.8in
   X-offset: 5.55in
   Width: 2.2in
   Height: 0.8in

4. Click inside the box and type the following text:

   OL Mobile
   2030 Pie-IX Blvd, Suite 500
   Montreal (QC) Canada, H1V2C8

www.olmobile.com

> **Tip:** Click Shift + Enter to insert a line break.

5. Reopen the Print section by double-clicking it. You will see that the address is now shown on every page of the Print section.

2. Adding and applying Master pages

The address on the Master Page appears on all pages of the section, but it only needs to be visible on the first page of the section. The solution is to add a second Master Page and to apply each Master Page to different pages, or rather 'sheet positions'. This exercise shows how to do that.

1. On the **Resources** pane, right-click the **Master pages** folder and select **New Master Page**.
2. A small dialog will appear with a name for the new master page (Master page 2). Click **OK** to accept the proposed sizes and add the new Master Page.
3. On the **Resources** pane, open the **Context** folder, and in the **Print** folder, right-click the section (**Section 1**) and select **Sheet Configuration...**
4. Uncheck the option **Same for all positions**. In this context, a **position** is the position of a sheet in the output. A sheet can be the first or the last sheet in the output, or one of the sheets in between. If the output consists of one sheet, it has the 'Single sheet' position.
5. For the front of the **First Sheet** and **Single Sheet** position, select the Master Page with the address box: use the **Master Page Front** drop-down to select **Master page 1**. (When **Duplex** – double-sided printing – is not enabled, you can only select a Master Page for the front.)
6. For the **Middle Sheet(s)** and **Last Sheet** position, select **Master page 2** as the Master Page for the front.
7. Click **OK** to close the dialog and apply the changes.
8. Toggle to **Preview** mode to verify that the Master pages are displayed correctly.

Adding a barcode

We want the letter to have a QR barcode that opens a URL with a data field value. This chapter demonstrates various ways to add the data field value to the URL.

1. Adding a QR code

1. Position the cursor at the end of the document.
2. Choose **Insert > Barcode... > QR Code**. The **Insert Barcode** dialog appears.
3. From the Field drop-down, select **first_name**.
4. Change the **ID** into `QRcode`.
5. Click **OK** to insert the barcode.
6. Now click the barcode box, hover over an edge so that the cursor changes, then click again and drag the box to the bottom right corner of the page.

7. Toggle to Preview mode and use the browse buttons in the Value column to browse through the imported data.

2. Adding a URL to the barcode: Helper

The QR code is now based on the value of the first_name data field, but we want it to contain a URL that has the value of the first_name field as parameter. One way to construct such a URL is by using the **Handlebars Helper wizard**.

The Handlebars Helper wizard creates a *Helper* that concatenates multiple data fields and that can add a prefix and/or suffix to each data field.

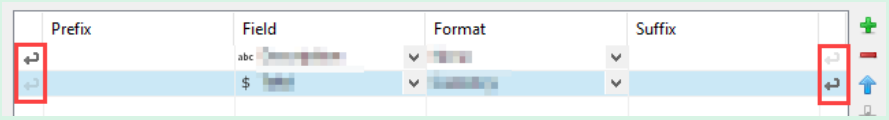
> **Note:** A *Helper* is a function that can be used in an expression.

1. Open the **Handlebars Text Helper** wizard: on the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Handlebars Text Helper**. The Edit Text Helper dialog appears.
2. Change the **name** of the Helper so that it clearly describes the purpose of the script (e.g. *setBarcode*). The name must be unique.
3. The Handlebars Text Helper wizard allows to select one or more data fields. Each field is placed on its own line in the wizard. New lines can be added by clicking the Plus sign next to the table or simply by clicking on the next empty line.
   In the **Prefix** and **Suffix** fields you can add text to be combined with the data.

Enter the following on the first line:

Prefix	https://olresourcecenter.uplandsoftware.com/?salutation=Hello%20
Data field	first_name
Suffix	,

Tip: For a line break, you can click the Prepend line break or Append line break icon.



You can type   for a space that needs to go into the text of the template.

4. On the **Properties** tab, change the **Scale** to **Proportional**, so that the QR code will fit in its box.
5. Click **OK**. The name of the script, *setQRUrl*, is added to the **Scripts** pane under the **Standard** scripts folder
6. Change the text inside the QR Code from {{first_name}} into **{{setQRUrl}}**.
7. Toggle to Preview mode and try the QR code.

2. Adding a URL to the barcode: script

Normally, one of the big advantages of the Handlebars Helper wizard is that empty lines and spaces are automatically removed when data fields have no value.

Note: When a data field is empty the respective line is skipped *including* its prefix and suffix.

However, in this particular case, we want the QR code to have the URL even if the first_name data field is empty. This is better done with a script.

1. On the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Standard Script**. The Edit Script window appears.

   The Script Editor allows to select elements in a template and apply an action to them. The selection is not limited to items inside sections; scripts may also control other elements, such as elements on master pages.

   Actions include changing styles and formatting, hiding, or removing elements, replacing elements and injecting content (from HTML snippets, for example).

2. Change the **name** of the script so that it clearly describes the purpose of the script (e.g. *setQRUrl*). The name must be unique.

3. In the **Selector** field, type `#barcode`. This refers to the ID of the barcode which you set when adding the barcode.

4. Type the following script.

```
let myUrl = "https://olresourcecenter.uplandsoftware.com/?salutation=Hello "
            + record.fields.first_name

results.text(myUrl)
```

   Note: In OL Connect Designer all scripts are written in JavaScript, a scripting language that was originally developed to create interactive web pages. For this walkthrough you don't need to know anything about JavaScript – you may copy the script - but it will pay off to learn the basics.

5. Click **OK**.

6. Toggle to Preview mode and try the QR code.

7. Both the Helper and the script have an effect on the QR code, but the effect of the Helper is no longer visible: it is overwritten by the output of the standard script. You can see this in the code when you scan it: there is no comma at the end of the URL. This is due to the order in which scripts are executed. Control scripts are executed first; then come the Standard Scripts.

   To keep the template clean it is better to remove the Helper and the expression in the QR code.

Using snippets

This chapter explains how to write a script that inserts part of a snippet (a small HTML file) in the template, depending on the value of a field in the record set.

1. Importing a snippet

The company, OLSG-Mobile, has several offices. This exercise shows how to import a snippet that contains the addresses.
Snippets are imported in much the same way as images are imported.

1. Locate the snippet **olsg_address.html**, provided with this tutorial.
2. Copy the snippet into the **Snippets** folder in the **Resources** pane. (If asked, click OK to accept the encoding.)
3. Double-click the new snippet to open it in the workspace.
4. Click the **Source** tab at the bottom of the workspace, to view the HTML of the snippet.

Every paragraph in this snippet has an ID that is identical to one of the values used in the province data field, as you can see in the **Data Model** pane. In the following exercise these ID's will be used to select a paragraph from the snippet and insert that in the template.

2. Inserting a snippet via script

In this exercise you will learn how to write a script that fills the box on Master page 1 with a variable address: the address in Ontario or the address in Québec. A customer living in Ontario should see the Ontario address. All other customers should see the address in Québec. The two addresses are stored in the snippet that you imported in the previous exercise.

1. Open the Master Page that contains the address box.
2. Select the box (**div**) using the breadcrumbs.
3. Give the box an ID: on the **Attributes** pane, in the **ID** field, type `office`.
4. Right-click the box. Make sure not to right-click the text inside it.
5. Choose **New Script** from the shortcut menu. The **Script Editor** appears. The name and selector of the script are added automatically. The ID of the box is used as the script's selector.
6. Change the script name to **Change Company Address**.
7. Add the following script.

```
var office = loadhtml("snippets/olsg_address.html", "#" +
    record.fields.province);
results.html(office);
```

In **Preview** mode and when output is generated, this script will read the value of the **province** field in the current record. It will load the paragraph that contains the

address of the office in that state value from the snippet and insert that in the template.

> **Note:** The functions used in this script are documented in the Designer Scripts API in the online documentation.

8. Click **OK** and switch to **Preview** mode to see if the correct address appears in the template.

What's next?

The letter template is now ready.

It can be merged with data loaded from a file or database and printed from OL Connect Designer.

With the Professional or Enterprise edition of OL Connect Designer you can also open a data mapping configuration and merge and print the template with the extracted data, or you can merge the template with data in an automated process and the output distributed via email or sent to a printer.

Building an automated print process

OL Connect Workflow lets you automate the processing, distribution and printing of business documents. Once installed on a server, it can be set up to automate all tasks related to document processes.

Workflow caters for inputs from a huge variety of sources, such as email, web pages, databases, individual files (PDF, CSV, XML, etc), print streams, FTP, and ERP systems. Data from these inputs can be extracted, analyzed, modified, stored, verified, routed and used as triggers for other processes from entirely within Workflow. And of course data can be merged with Designer templates and outputted in multiple ways (printed, emailed, posted, archived, sent to third party solutions, etc.).

This walkthrough will teach you how to create a **print process** for OL Connect documents – invoices in this case - in the Workflow configuration tool.

It consists of three parts. The first part shows how to build the simplest possible print process in Workflow. Each following part gives you more control over the process and its output and introduces a few more key aspects of Workflow.
The final process merges data with a template to create invoices, prints them to different files – grouped by state - and archives each individual invoice in a folder.

You will discover the OL Connect tasks you need and the appropriate order of tasks in this process and you will learn how to create a branch in a process.

You will also be introduced to basic Workflow concepts, such as the *job file* and *metadata*. This will help you understand what is going on in the process and will make it easier for you to build other Workflow processes in the future.

Note: A PDF version of this walkthrough is available for offline use: Workflow_Print_WT.pdf.

Note: The resources needed for this walkthrough can be downloaded here: Workflow-Walkthrough.zip

Part 1: A simple print process

This part shows you the quickest and easiest way to build a print process in OL Connect Workflow. You will learn how to create a Workflow Print process that uses the All In One task (note that tasks are also called "plugin" in Workflow).

A print process in Workflow normally requires just two or three tasks. (The input file required by each of the tasks is underlined.)

- An Input task, usually a **Folder Capture**. This task retrieves files corresponding to a specified file mask, from a specified folder. The input file typically contains data to be merged with a template.
- The **All In One** task. This task combines four different OL Connect tasks:
  - The Execute Data Mapping task. This task generates a record set by executing a data mapping configuration on a given data file.
  - The Create Print Content task. This task creates print content items by merging a template's Print context with the records generated by the Execute Data Mapping task.
  - The Create Job task. This task collects print content items generated by the Create Print Content task, and can filter, group and sort them using a Job Preset. This (optional) Job Preset needs to be created in OL Connect Designer.
  - The Create Output task. This task generates print output (a spool file, or files on disk) in a format specified by an Output Preset, and sends this output to the appropriate target location. The Output Preset needs to be created in OL Connect Designer.
- A **Send to Folder** task (optional).

Why use the All in One task?

The All in One task is optimized: it exchanges less data with the server than the tasks do when used separately, and content creation can already start whilst the data mapping is in progress.

The All in One can be configured to carry out just the two first tasks. Even then, the All in One delivers better performance than the individual tasks.

> **Note:** Only replace the All In One with the separate tasks that make up a print process when it is absolutely necessary. For example, when the extracted data have to be edited or enhanced before the print content is created.

Preparation: collecting the necessary files

To be able to follow this walkthrough you'll need the following OL Connect files:

- A data file. The XML file provided with this tutorial: **olsg-data.XML**, contains the data to be merged with the template.
- A data mapping configuration. Use the file **olsg-Invoice_XML.OL-Datamapper** provided with this tutorial or the configuration that is the result of the DataMapper tutorial 'Creating an XML Data Mapping Configuration'. This data mapping configuration was designed to match the supplied XML file.
- A template with a Print context. The template **olsg-Invoice.OL-template** is provided with this tutorial. Alternatively, you could use the template that is the result of the Designer tutorial 'Creating an invoice'.

This walkthrough starts with a simple print process without grouping, sorting or filtering, so **no Job Preset** is needed at this time.

In addition to the data file, data mapping configuration and template, the All In One task does need an **Output Preset.** An Output Preset determines in what format and to which destination a print job is outputted. You will create one in the first exercise.

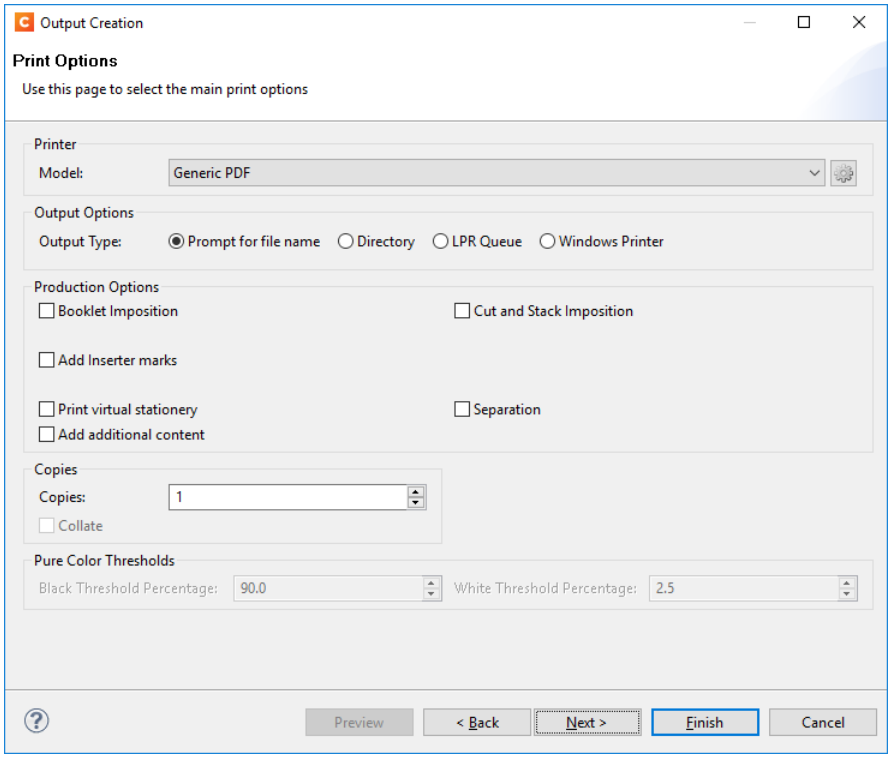
Creating an Output Preset

In addition to the data file, data mapping configuration and template, the All In One task needs an Output Preset. An Output Preset determines in what format and to which destination a print job is outputted.

This exercise shows how to create an Output Preset to store print output in one PDF file.

If it isn't running already, launch OL Connect Designer by double-clicking the application icon.

1. Choose **File > New > Presets > Output Preset**. The *Output Creation dialog* appears.
2. Click **Next** to go to the Print options page.



3. Select **Generic PDF** in the *Printer model* drop-down. In this walkthrough it is assumed that the output has to be saved to a file.

> **Note:** In a new installation, Generic PDF is the only listed option. If you have a printer at hand, you may click the *Import Definition* button at the right, select *Edit available printers* and add your printer model to the list.
>
> In this walkthrough it is assumed that the output has to be saved to a file.

4. Select **Prompt for file name**. Of course, in an automated process you don't want to be prompted for a file name, but don't worry: you won't. The print process in Workflow will be configured to determine the actual output location. Therefore you could choose any option for the output type here; however *Prompt for file name* is easiest because you don't have to make any additional settings.

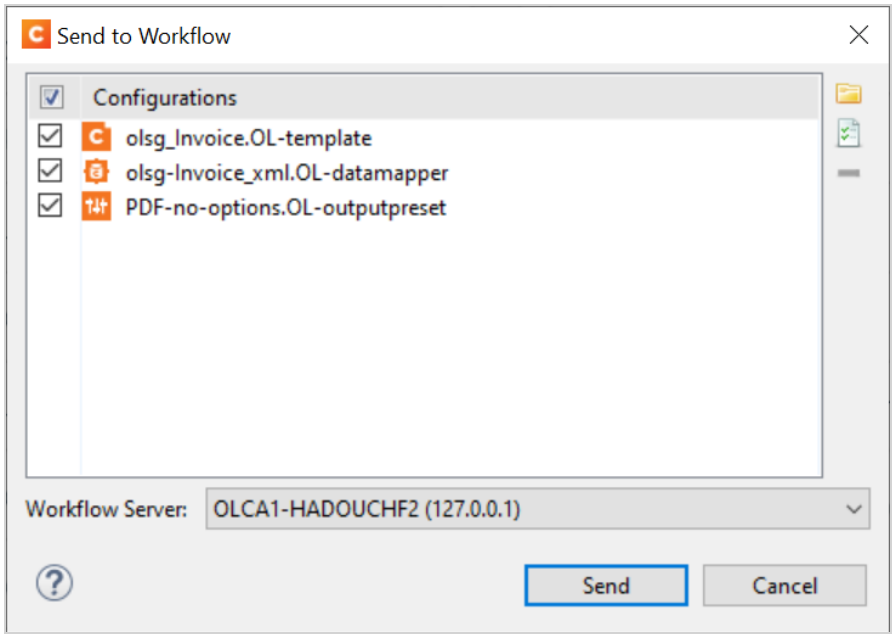
> **Tip:** Check **Print virtual stationery** if you want the related Media to be printed when generating the invoice in a PDF format. The Media (also called virtual stationery) represents the preprinted paper.

5. Click **Next**, then **Finish**. The *Save output preset* window is opened.
6. Save the Output Preset as **PDF-no-options.OL-outputpreset**.

Sending OL Connect Designer documents to Workflow

Before the required files can be used in Workflow, they have to be sent to Workflow from within the Designer.

1. Choose **File > Send to Workflow**... The *Send to Workflow* dialog appears.
2. Click on the Browse button and select the template, data mapping configuration and print presets:
   - **olsg-invoice.OL-template** (or the template that you have made following the Designer walkthrough 'Creating an Invoice')
   - **olsg-Invoice_xml.OL-datamapper** (or the data mapping configuration that you have made following the DataMapper walkthrough 'Creating an XML Data Mapping Configuration')
   - **PDF-no-options.OL-outputpreset**



3. In **Workflow Server**, select your Workflow IP address. Click **Send**. All selected files are now sent to Workflow.

> **Note:** If no IP address is displayed, you can create a package file. To do that, go to the **File** menu and click **Package...** To import a package file inOL Connect

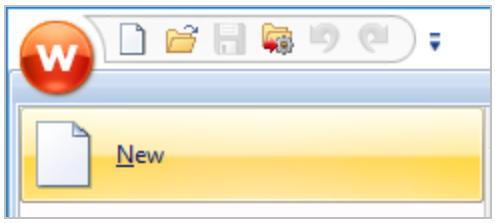
Workflow, right-click *OL Connect Resources* in the *Configuration Components* pane and select **Import Connect Content**.

4. Launch Workflow and verify that all the necessary files are available by browsing the subfolders under *Connect Resources,* in the *Configuration Components* pane.

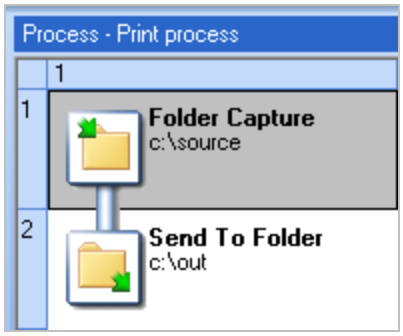
Creating a print process

Creating a Workflow configuration

To start creating a new Workflow configuration, click the Workflow button (W) at the upper left hand corner and choose **New**.



A Workflow configuration can consist of many processes. One process is always added at the start of a new configuration. By default it has two tasks: a Folder Capture input task and a Send To Folder output task.



The Folder Capture task is configured to pick up any file from the C:\source folder. The Send To Folder task outputs the input file, renamed but otherwise unchanged, to the C:\out folder.

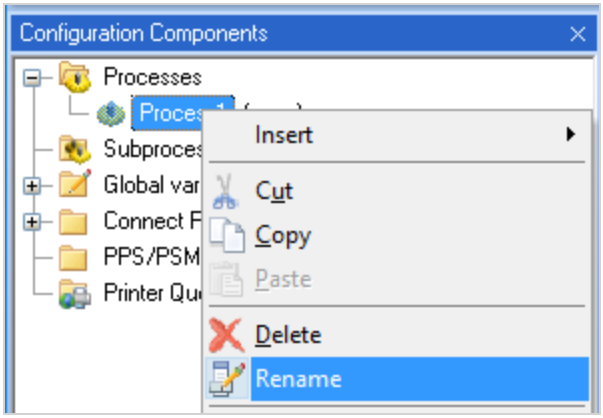
1. Save the configuration: click the Workflow button (W) and choose **Save As**. Type a name and click **OK**.

Creating a process and renaming it

The automatically added process is a good starting point for the print process so you don't have to add a process. But if you'd want to, you can add a process by right-clicking on **Processes** in the Configuration Components pane and then selecting **Insert > Insert**

Process, or by clicking **Process** on the Home ribbon. To delete a process, right-click on the process and select **Delete**.

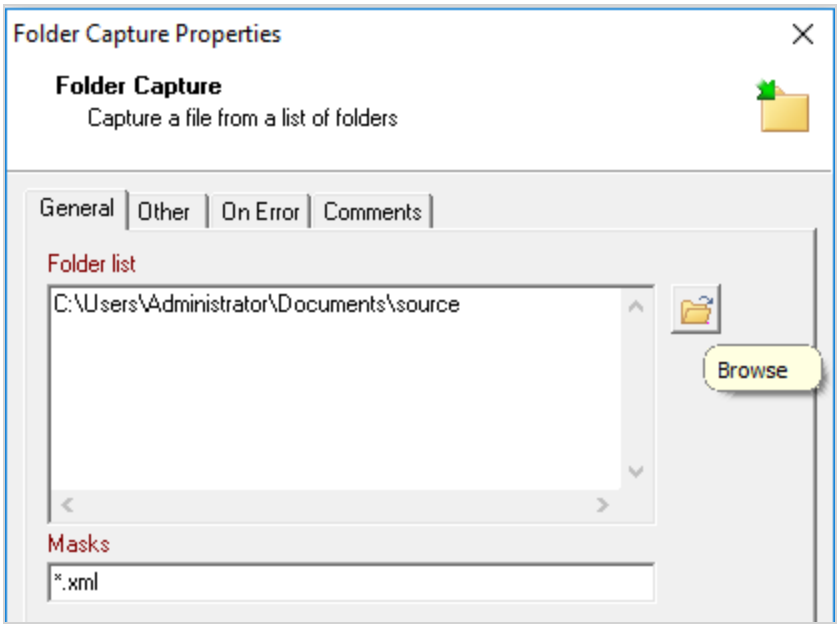
1. Rename the process by right-clicking on it and then selecting **Rename**. Enter **Print process**, for example.



Configuring the Folder Capture task

There are different kinds of Input tasks, as you can see on the *Inputs* tab in the *Plug-in Bar*. (The *Plug-in Bar* is located to the right of the Process area). The **Folder Capture** task retrieves files corresponding to a specified file mask, from a specified folder. In this exercise you will learn how to configure the *Folder Capture* task.

1. Double-click the *Folder Capture* task in the process area, or right-click on it and select **Properties**.The *Folder Capture* Properties dialog appears.



2. Click the Browse button next to the Folder List, and select the folder that you want to be monitored. For this walkthrough it doesn't matter which folder you choose, as long as you have access to it.
3. Change the Masks to *.**xml**

JOB FILE

Every time a file is picked up, the process starts and the input file becomes the job file. This file will be handed down through the process, from one task to another. It may be changed, copied, and split by tasks in the process. Often, as in this case, the job file is a data file, but it could also be an image or another type of file.

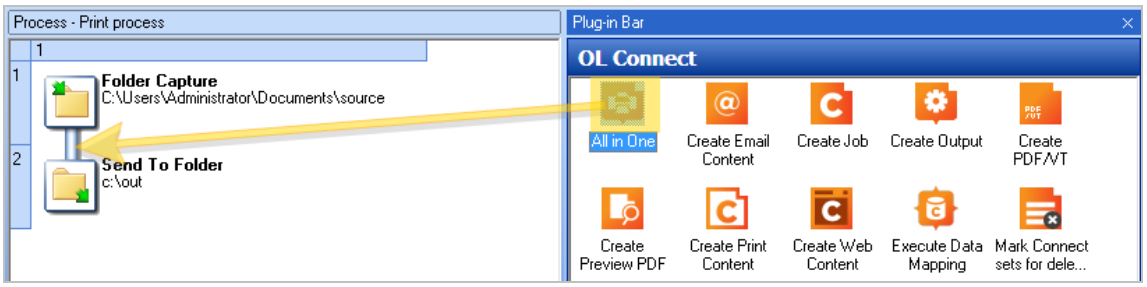
Each branch in a Workflow process can hold no more than one job file. If more than one file is retrieved via an Input task, then the entire process is run for each file sequentially. If the job file gets split during the process, then the rest of the process is run as a loop for each split portion.

(In this walkthrough the job file is not split. It does get copied, in part 3).

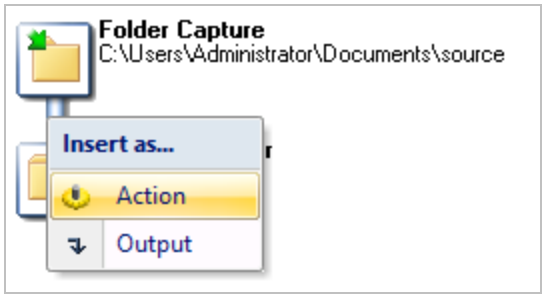
Adding the All In One task

Now you can add and configure the All In One task.

1. Click on the OL Connect tab in the Plug-in Bar. This is where all OL Connect tasks are displayed.
2. Locate the All in One task and drag and drop it below the input task.

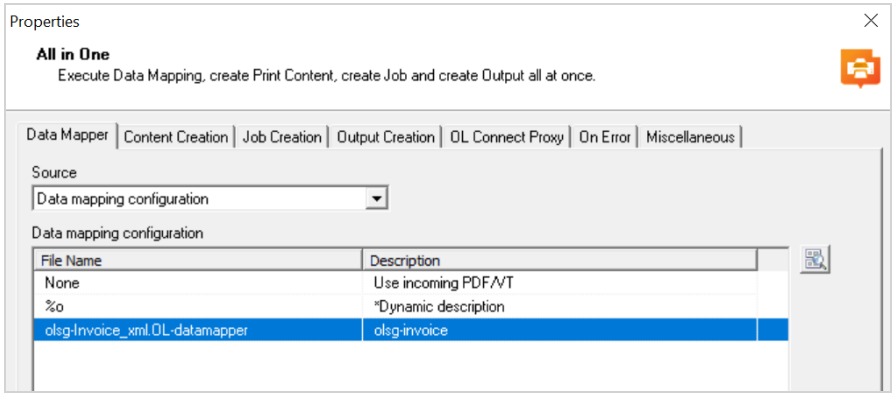


3. Choose **Action** to insert this task as an Action task.



> **ACTION/OUTPUT TASK**
>
> An Action task can be followed by other tasks, while an Output task is always located at the end of a process or branch.

The *All In One Properties* window appears.

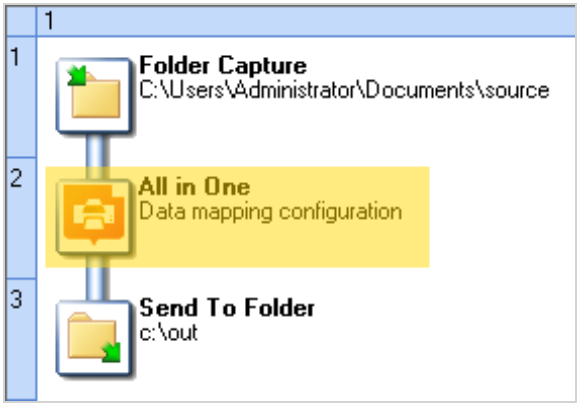


4. On the Data Mapper tab, select the data mapping configuration **OLSG-invoice_XML.OL-datamapper** (or your own data mapping configuration, if you've made one with the DataMapper walkthrough).
5. On the Content Creation tab, select the template OLSG-invoice.OL-template (or the template that you have made following the Designer walkthrough 'Creating an Invoice').
6. On the Job Creation tab, select the option **Default** ("use the IDs in the Metadata"). The process will create a job without sorting, filtering or grouping the print content items.
7. On the Output Creation tab select the Output Preset you have made in the preparation phase: **PDF-no-options.OL-outputpreset.**
8. On the same tab, in the Output Management section, check the **Through Workflow** option. This means that the output file will be handed down to the next task in the

process (the Send To Folder task in this case). Every option set in the Output Preset is still used, except for the output location.

9. Click **OK**.

   The process looks like this:



> **Note:** In this part of the walkthrough the Send To Folder task is maintained as a separate output task to be able to introduce you to a few key concepts in Workflow. But the simplest possible print process would consist of the Folder Capture and All In One task only. You could simply drag the All In One on the Send To Folder task to add it as an Output task. For this the Output Preset would have to define an actual directory or printer to send the output to.

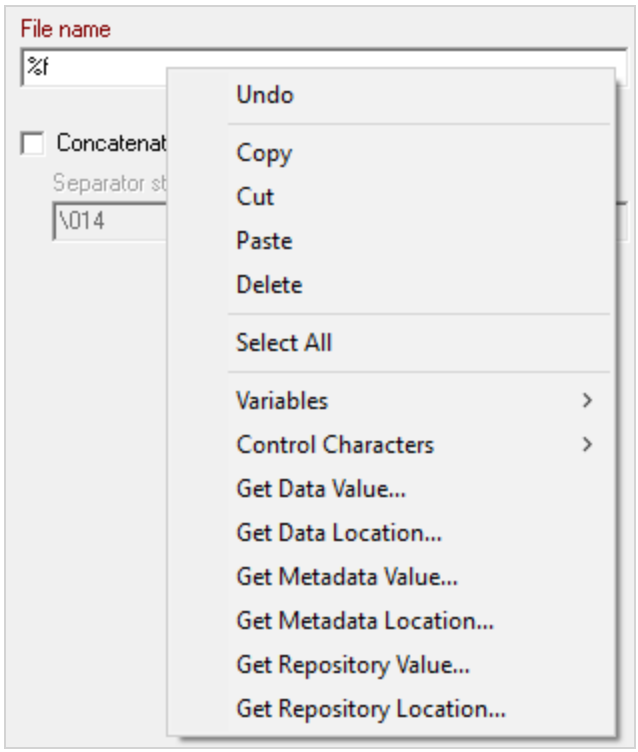
Configuring the Send To Folder task

Now we will configure the Send To Folder task. The Send To Folder task saves the **job file** to a file.

The last task in the All In One (the output creation task) replaces the original job file – the input data file - with the output PDF file. So, at this point in this particular process the job file is the output file.

1. Double-click the Send To Folder task, or right-click it and select **Properties**. The Send To Folder *Properties* window appears.

2. On the General tab, click the *Browse folder* button next to the Folder field and select an output folder. Again, for this walkthrough it doesn't matter which folder you choose, as long as you have access to it.
3. The *File name* field reads **%f**. This is a system variable that represents the job file name. Right-click the File name field. A contextual menu appears, giving access to variables and data selection functions.



> **Tip:** Right-clicking on any a field with a maroon field name opens the contextual menu that gives access to variables and data selection functions.

4. Select **Variables > System > %O** (Original file name without extension) and complete the file name by adding .**PDF**
5. Click **OK**.

Running the print process

When the process is complete, you can test it in debug mode within the Workflow configuration tool, before sending it to the Connect Workflow service. First, though, you have to enter the appropriate server connection settings.

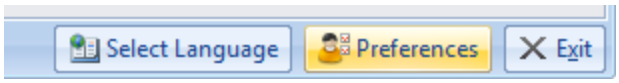
Entering OL Connect Proxy information

All OL Connect tasks require the OL Connect Proxy information. The tab OL Connect Proxy, present in the Properties window of each task, defines where to process the jobs sent through OL Connect tasks. This is common to all OL Connect tasks. The required information is:

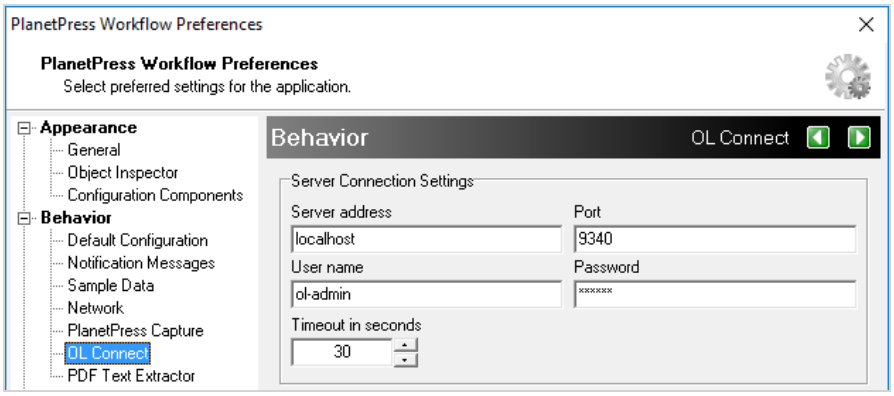
- **OL Connect Proxy Address**: The machine name or IP Address where the OL Connect Server resides. If OL Connect Server is installed on the same machine as the Workflow configuration tool, this is **localhost**.
- **Port**: The port used to communicate with the OL Connect Server. Default: **9340**.
- **Username**: The user name expected by the OL Connect Server. By default it is **olc-user**. Take a look in the OL Connect Server configuration or in the OL Connect Designer preferences to see which user name is used. (Open OL Connect Designer and choose Window > Preferences > Print > General.)
- **Password**: The password expected by the OL Connect Server for the above user name. By default the password is set to **secret**.

Usually these values are not set per task. You can set the default values in the OL Connect User options page of the Workflow Preferences:

1. Click on the Workflow button (W) and then on the Preferences button on the bottom right.

   The *Preferences* dialog appears

2. Click **OL Connect** in the *Behavior* section on the left.
3. Enter the appropriate information under **Server Connection Settings**.

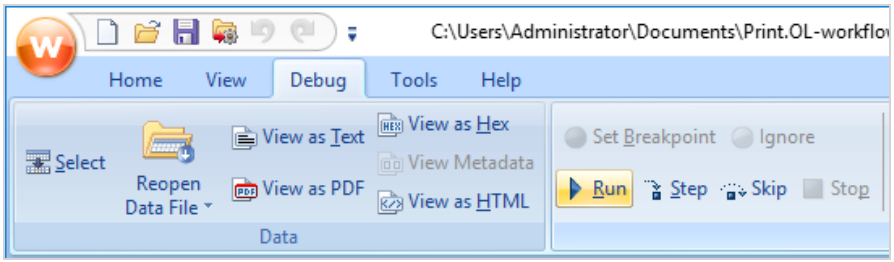
Debugging the process

The process is now complete. You can test it within the Workflow configuration tool before sending it to the OL Connect Workflow service.

As sample data file, use the XML file (**olsg-data.xml**) provided with this walkthrough to debug the process.

1. Click **Debug** in the top menu to open the *Debug* ribbon.
2. Click **Select**. Browse to the XML file (**olsg-data.xml**), select it and click **OK**.

   Tip: You can also use a sample data file included in a data mapper configuration. Select **Connect Resources > Data Mapping Configurations >** *your data mapping configuration*, right-click a data file and choose **Set as sample data file**

   The *Run*, *Step*, *Skip* and other buttons become enabled after a data file has been selected. You can now debug the process using these buttons.
3. Click **Run**. This executes the entire process in one go.

   In the Process area, a blue arrow pointing downwards signals which task is currently being executed.

   The *Messages* pane gives information about each task and displays any error messages

   The *Debug Information* pane gives information about the job and displays the contents of Workflow variables
4. Open the Windows File Explorer, browse to your output directory and check that the output file is there. When you double-click the output file it will be opened with the default PDF reader (often a browser).

Stepping through the process

By executing the process one step at a time, you can observe changes in the data to understand exactly what is happening.

1. Test the process again, now using **Step**. After each step:
    - Open the job file as Text (click **View as Text,** or double-click the name of the job file in the Debug Information pane). You will see that the job file is a copy of the XML input file – until the All In One task has completed its job.
    - Open the metadata file (click **View Metadata,** once the link is enabled, or double-click the name of the Metadata file in the Debug Information pane).

      METADATA FILE

      When an OL Connect task adds data to the **OL Connect database** it also adds data about that data to the **metadata** file. Metadata aren't created just to make debugging easier. These metadata tell the next step in the process what it needs to work on. They could also be used by other tasks to refer to the entities in the OL Connect database (not demonstrated in this walkthrough). A set of special Workflow tasks allows to edit the metadata during a Workflow process.

      OL Connect tasks store their metadata under *User defined information*.

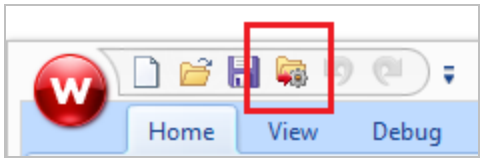
    - Open the job file as PDF (click **View as PDF**, once the link is enabled). The last task in the All In One (the output creation task) replaces the original job file – the XML file - with the output PDF file.
2. Check the output in your output folder again. The previous file has been overwritten.

Sending the configuration to the OL Connect Workflow service

No errors, or all problems solved? Then the Workflow configuration can be 'sent' to the OL Connect Workflow service.

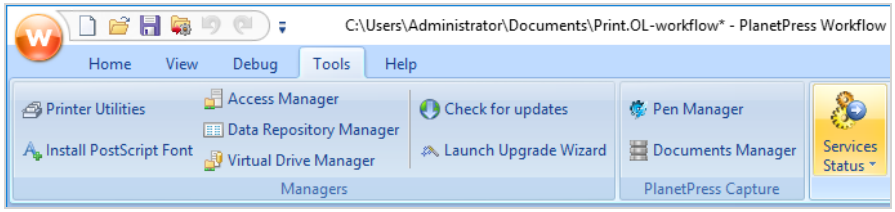
Note: The OL Connect Workflow service can only run one configuration at the same time. The service reads its configuration file when it starts up. If the service is running when you send a new configuration, it automatically stops and restarts with the new configuration.

1. Click on the *Send configuration* icon in the toolbar at the top left of the screen.



   Or use the menu: click the Workflow button (W) > **Send Configuration > Send Local**.

2. Start (or restart) the services: on the *Tools* tab in the menu, click **Services status > Start**



Testing the configuration on the server

Finally, verify that the OL Connect Workflow service functions as expected with the new configuration file.

1. Copy the XML data source file (**olsg-data.xml**) to the input folder defined in the Folder Capture task. The file will almost immediately disappear from the folder when it is picked up by the Folder Capture input task.
2. The print process is executed automatically. When it has finished, you can open the PDF output file in the folder that you have selected in the Send To Folder task.

Part 2: Grouping and sorting

This part of the walkthrough shows how to create a print process that outputs groups of documents to different files (or jobs).

Grouping and sorting documents and outputting them to different print jobs can be important, not just for large mailers who earn a discount by pre-sorting their mail, but also for smaller companies, for example when some documents have to be put in bigger envelopes than other documents.

Key elements in such a print process are the **Job Preset** and **Output Preset**. A Job Preset can group, filter and sort the print content items that have been generated when merging a template with a record set. An Output Preset can then split the output into different files or jobs.

In order to get the invoices sorted by their number of pages and to then output each group of invoices to a different file, the print process needs a Job Preset and a matching Output Preset. We will first create the necessary presets and then adjust the print process.

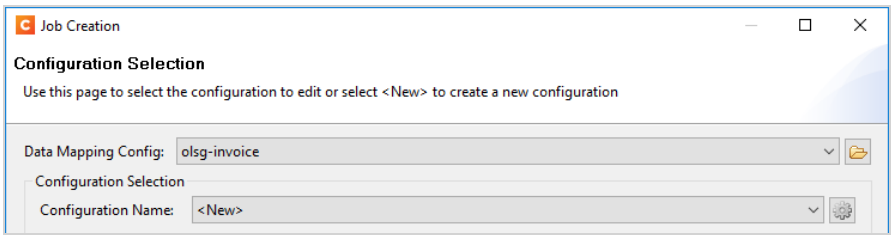
Creating the Print Presets

In this exercise you will create both the Job Preset and Output Preset for a print process that groups invoices by their number of pages and outputs each group of invoices to a different file.

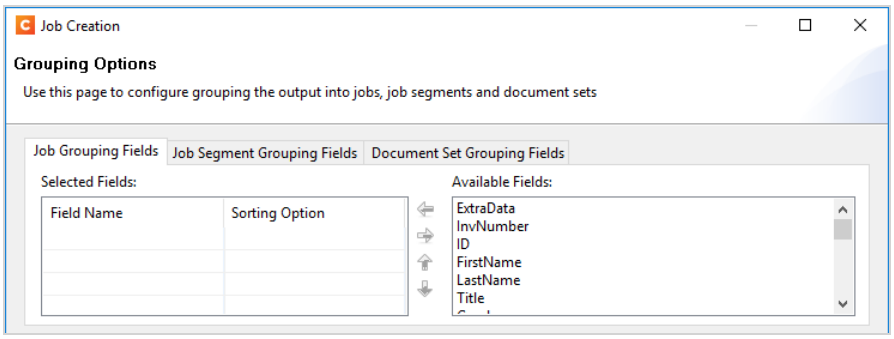
Job Preset: grouping

This Job Preset groups items by their number of pages (1 to 2, or above), so that the actual output can be split into one file (or one job if the output task would actually output to a printer or printer queue) containing documents that have one or two pages, and one with larger documents.

1. Launch OL Connect Designer by double-clicking the application icon.
2. Choose **File > New > Presets > Job Preset**. The Job Creation dialog appears.



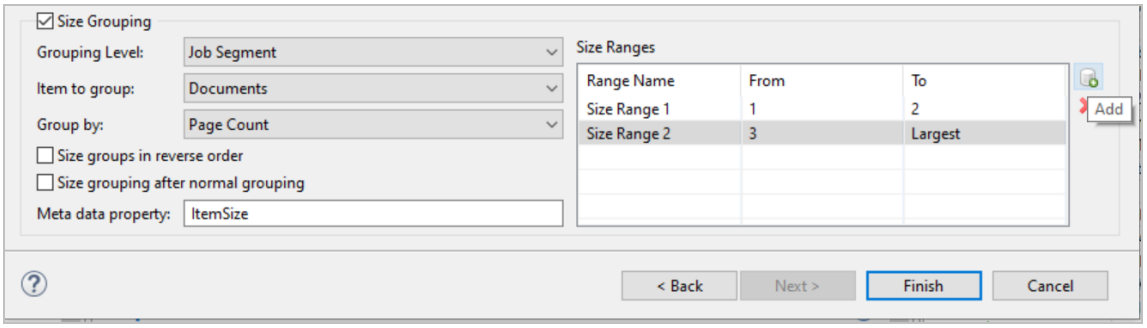
3. Click on the Browse button to the right of the Data Mapping Config drop-down list and select the data mapping configuration for XML provided with this tutorial: **OLSG-invoice.OL-Datamapper**, or the data mapping configuration that you've created with the DataMapper tutorial 'Creating an XML Data Mapping Configuration'. Without a data mapping configuration it isn't possible to select a data field in the criteria for grouping.
4. Check **Use grouping** in the Options section and click **Next**. The Grouping Options dialog opens with the Job Grouping Fields tab selected by default. The tabs are supplementary to one another, so you could group documents in document sets, job segments and jobs. The lower part of this dialog is reserved for page break grouping options.



5. In this exercise we're only grouping the documents by their number of pages. **Size Grouping.**

   Note: Page break grouping can also be used in addition to 'normal' grouping. It can take place before or after the grouping on the chosen level.

6. Size grouping can only be done on one level. Select **Job Segment** from the *Grouping Level* drop-down list.
7. Select **Page count** from the **Group by:** drop-down
8. Click the *Add* button to add a size range.

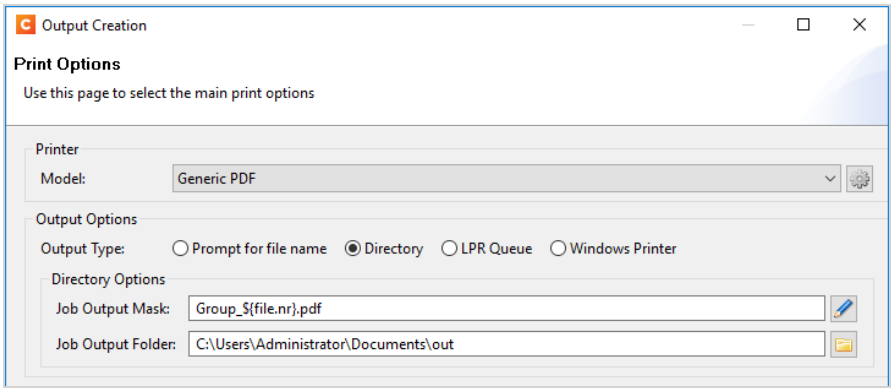


9. *Change Size Range* 1 so that it contains all documents from 1 to 2 pages: click in the *To* field and type **2**. *Size range 2* is automatically adjusted to contain all documents larger than two pages.
10. Click **Finish**. The **Save file** window opens.
11. Save the Job Preset as **Group-by-pages.OL-jobpreset**.

Output Preset: separation

This Output Preset **splits** the groups of content items and outputs them to different jobs/files.

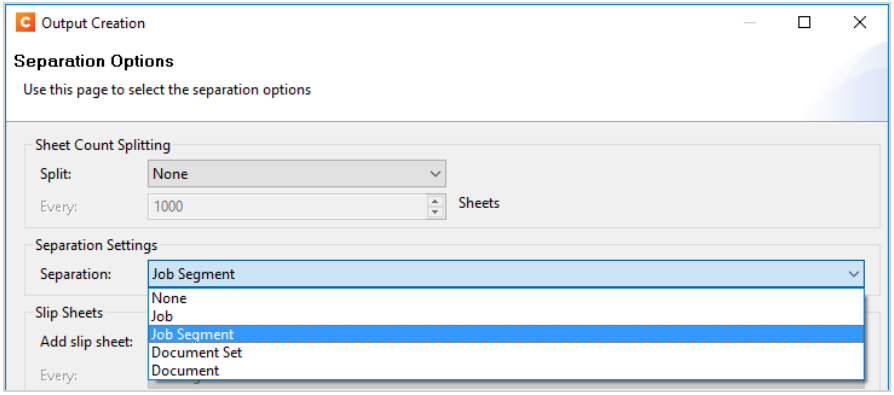
1. In OL Connect Designer, choose **File > New > Presets > Output Preset**... The *Output Creation* dialog appears.
2. Select **Generic PDF** in the *Printer model* drop-down.
3. Select **Directory** and enter a directory.
4. In the Output Mask field, type **Group_${file.nr}.pdf**.



Tip: For a list of variables that start with $ see: Variables available in the Output

You will learn how to use variable **data** in file names in the next part of this walkthrough.

5. Check **Separation**, then click **Next**.
6. The invoices have been grouped on the Job Segment level in the Job Creation task. Choose **Job Segment** from the Separation drop-down list under Separation Options to create one file for each of these groups.



7. Click **Next**, then **Finish**. The Save file window is opened.
8. Save the Output Preset as **PDF-separate-jobsegments.OL-outputpreset**.
9. Send the Job Preset and Output Preset to Workflow (**File > Send to Workflow**...).

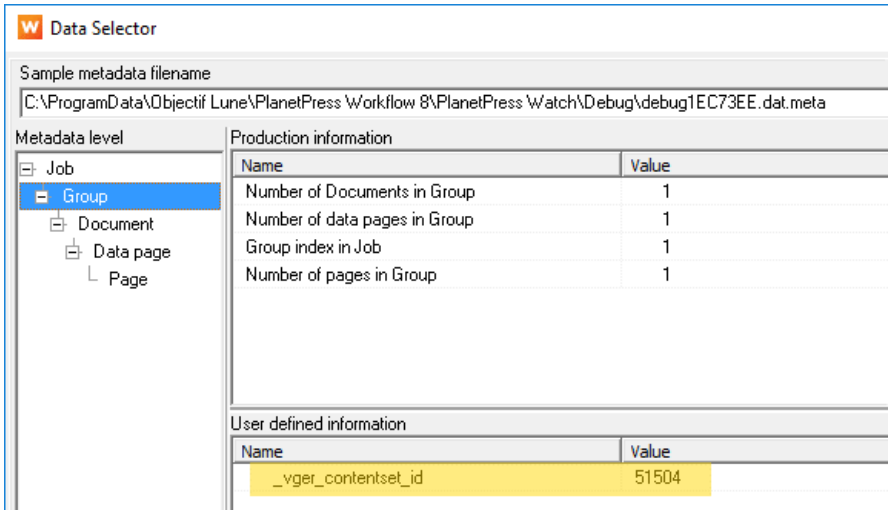
Adjusting the print process

Launch the Workflow Configuration tool if it isn't running yet. The files sent from the Designer will automatically be unpacked and can be found in the OL Connect Resources folder. Now the print process has to be adjusted to use these files.

1. Double-click the All In One task.
2. On the *Job Creation* tab, select Job Preset: **Group-by-pages.OL-jobpreset**.
3. On the *Output Creation* tab, select the Output Preset: **PDF-separate-jobsegments.OL-outputpreset**.
4. Save the process.

Debugging the process

1. Debug the process using the Step button on the Debug tab. Open the job file and metadata file after each step. The **metadata** file now displays data at different levels. Click on a **Metadata** level to the left, to see the data that has been attached on that level (under User defined information). The Job node has information about the job as a whole, a Group has information about a record set and a Document about one record. The Group node in this picture contains the ID of the content set in the OL Connect database:



> **Note:** When working with OL Connect tasks, only Job, Group and Document nodes can contain information about the actual job. Although visible in the metadata file, the Datapage and Page nodes don't contain any actual job related information.

In this process, viewing the output PDF isn't possible, for two reasons:

- The Create Output task is the last task in the process, so the output isn't handed back to the process.
- The Create Output task only replaces the job file with the PDF output if there is only **one** output file. When the output is **separated** it replaces the job file with a **CSV file** that lists the paths to the outputted files.

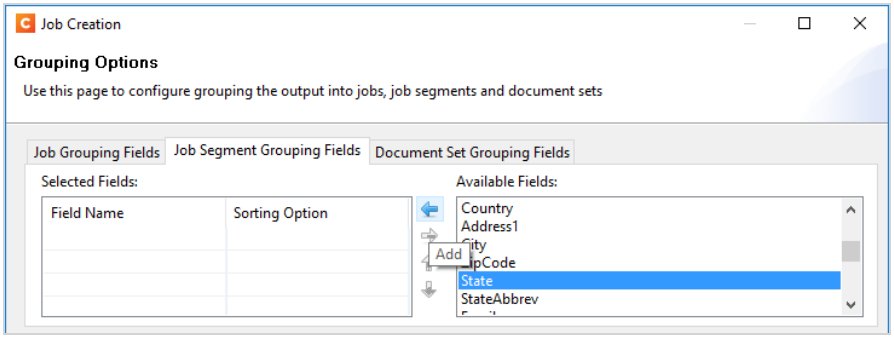
2. Open the Explorer, browse to your output directory and check that the output files are there. You should get two output files: one for each page range. You can open them with a PDF reader or browser.
3. Finally, upload the configuration to the OL Connect Workflow service and test it.

Exercises (optional)

A. Create a Job Preset that groups the documents by **state** instead of their number of pages. You should generate 10 output files, one for each state.

Solution:

1. Create a new Job Preset and open the same data mapping configuration as before.
2. On the **Job Segment** tab, select the data field **State** in the Available Fields list.
3. Click the blue arrow to add the selected data field to the *Selected Fields* area.

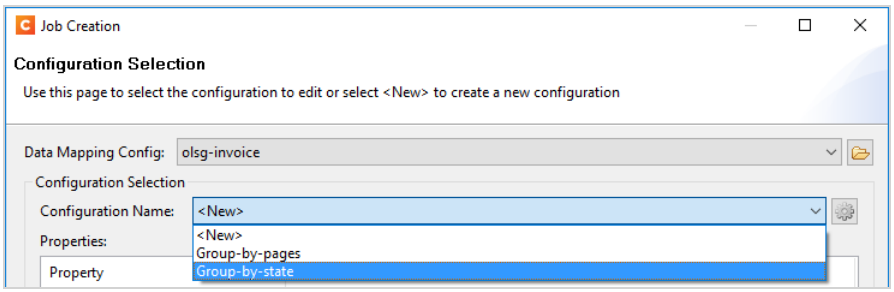


4. Save the Job Preset with the name **Group-by-state.OL-jobpreset**.
5. Send the Job Preset to Workflow, select it in the *Job Creation* tab of the All In One properties window and try debugging the process.

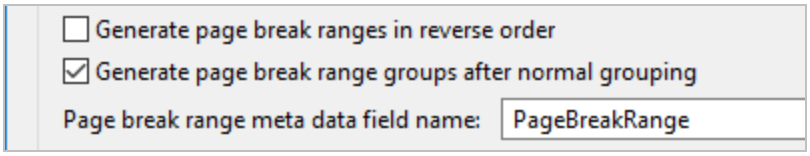
B. Try combining page break grouping with normal grouping.

For example:

1. Open the Group-by-state Job Preset in the Designer: select **File > New > Presets > Job Preset** and choose **Group-by-state** from the Configuration Name drop-down list.



2. Add page break grouping on the **Job Segment** level and create two page ranges, one for documents with 1 – 2 pages and one for larger documents, just as in the Group-by-pages Job Preset. Leave the other options unchecked.
3. Save the Job Preset and send it to Workflow. Select this Job Preset in the Job Creation tab of the All In One task properties window. Now try debugging the process. You should get one or two output files per state (one if there aren't any documents larger than two pages for that state). Note that all documents with smaller invoices are generated first, and then all documents with larger invoices.
4. Open the same Job Preset in the Designer and check the option Generate page break range groups *after normal grouping*.



   The option Generate page break range groups after normal grouping only takes effect when normal grouping and page break grouping are applied at the same level.
5. Save the Preset and send it to Workflow. Again, you should get one or two output files per state, but with this Job Preset they are generated in a different order: per state, instead of per page range.
6. Open the same Job Preset in the Designer and remove the normal grouping on the Job Segment level. Instead, group documents by **State** on the *Document Set* level.
7. Save the Preset and send it to Workflow. When you run this configuration, you should get two output files, one with the smaller documents, grouped by state, and one with the larger documents, also grouped by state.

Part 3 Adding a branch for archiving

A process that archives invoices would up to a certain point do exactly the same as the process that prints the invoices. So instead of creating a whole new process, you can create a branch in the print process.

Like said, the All in One can be configured to carry out just the two first tasks: extract data and create print content items. If we do that, the main process needs a Create Job task that groups the invoices, and a Create Output task that separates the output. The branch needs to have the same tasks, but with a different Job Preset and Output Preset.

We will adjust the main process before we add the branch.

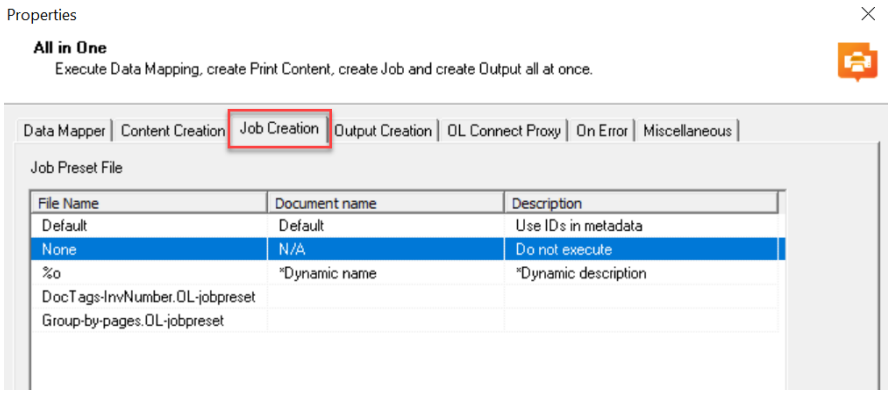
Adjusting the main process

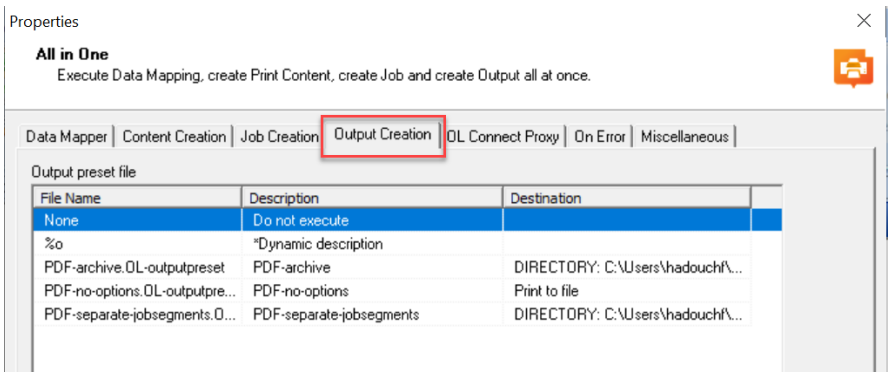
Launch the Workflow Configuration tool if it isn't running yet.

Turning off Job and Output Creation in the All in One task

Since we now want Job Creation and Output Creation to be handled by the separate tasks, these must be turned off in the All In One task.

1. Double-click the All In One task.
2. On the *Job* and *Output* Creation tabs, select **None** as shown in the images below. The All In One task will now stop after having created the print content items.



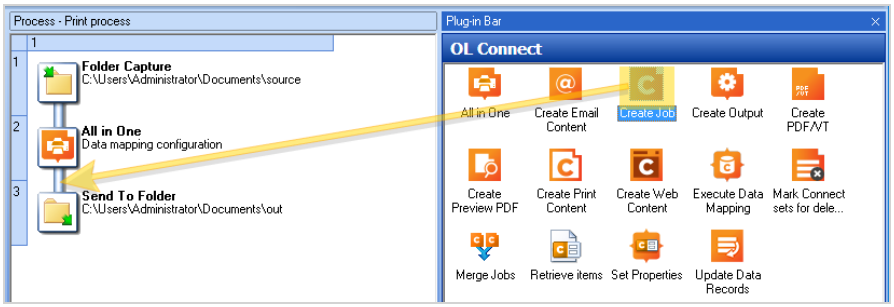


Adding the Create Job task

The next step is to add the Create Job task to the process. This task creates a **job**: a collection of content items. The items have to be grouped according to the Job Preset you

have just made.

1. In the **Plug-in** Bar, on the OL Connect tab, locate the **Create Job** task and drag and drop it underneath the All In One task.



   The **Create Job Properties** window opens.

2. On the General tab, select the Job Preset **Group-by-pages.OL-jobpreset**.

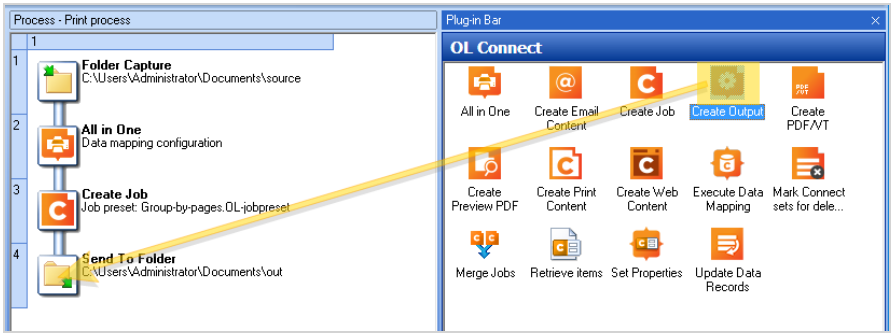
   > **Note:** Even with this task, using a Job Preset is not obligatory. The Default setting uses the IDs defined in the metadata to collect the print content items and put them in one job.

3. Click **OK**.

Adding the Create Output task

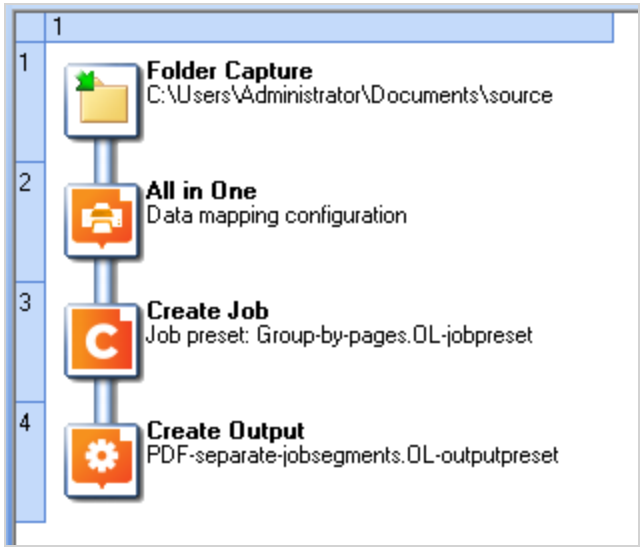
The Create Output task generates and distributes print output in a format and to a destination (a printer queue, Windows printer, folder...) specified by an Output Preset.

1. Locate the **Create Output** task on the *OL Connect* tab in the *Plug-In Bar* and drag and drop it **on** the *Send To Folder* task to replace it.



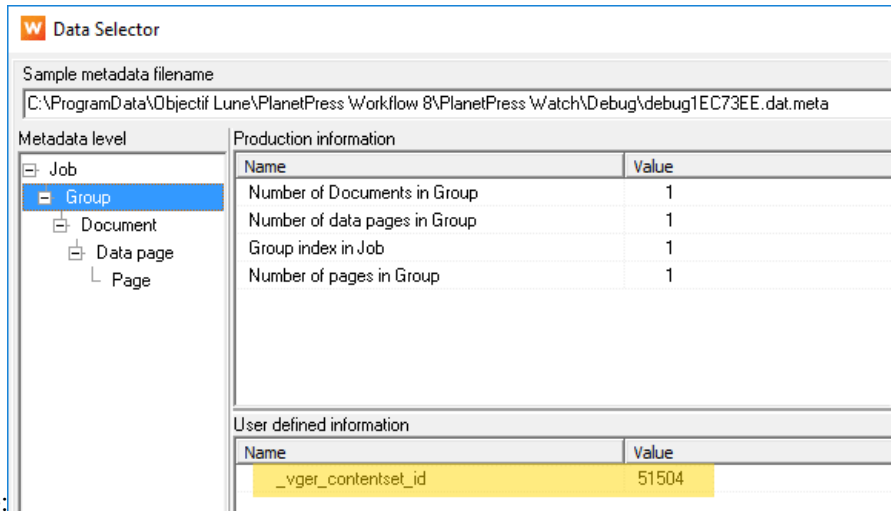
2. The **Create Output Properties** window appears. On the General tab, select the Output Preset: **PDF-separate-jobsegments.OL-outputpreset**.

3. Click **OK**. The Print process now looks like this:



Debugging the process

1. Debug the process using the Step button on the Debug tab. Open the job file and metadata file after each step. The **metadata** file now displays data at different levels. Click on a **Metadata** level to the left, to see the data that has been attached on that level (under User defined information). The Job node has information about the job as a whole, a Group has information about a record set and a Document about one record. The Group node in this picture contains the ID of the content set in the OL Connect database:



Note: When working with OL Connect tasks, only Job, Group and Document nodes can contain information about the actual job. Although visible in the

metadata file, the Datapage and Page nodes don't contain any actual job related information.

In this process, viewing the output PDF isn't possible, for two reasons:

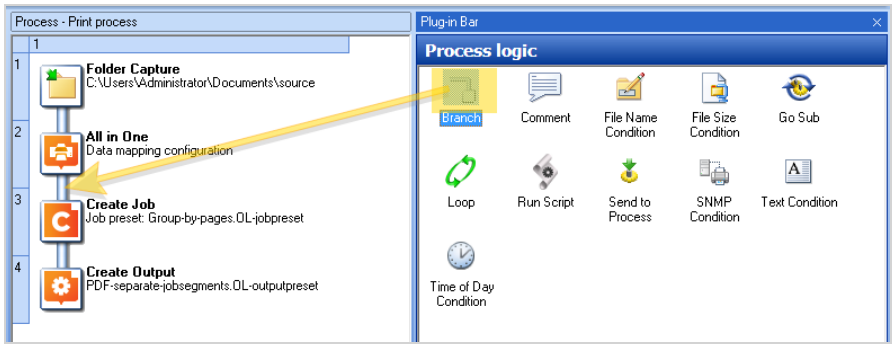
- The Create Output task is the last task in the process, so the output isn't handed back to the process.
- The Create Output task only replaces the job file with the PDF output if there is only **one** output file. When the output is **separated** it replaces the job file with a **CSV file** that lists the paths to the outputted files.

2. Open the Explorer, browse to your output directory and check that the output files are there. You should get two output files: one for each page range. You can open them with a PDF reader or browser.
3. Finally, upload the configuration to the OL Connect Workflow service and test it.

Creating the branch

Now we can create the branch that archives the invoices.

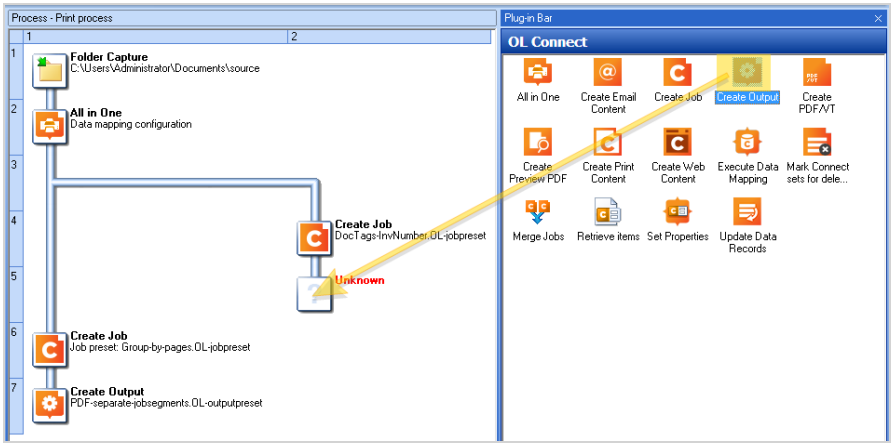
1. Launch Workflow and open the advanced print process.
2. In the Plug-in Bar, the Process logic tab, locate the **Branch** task and drag and drop it underneath the All In One task.



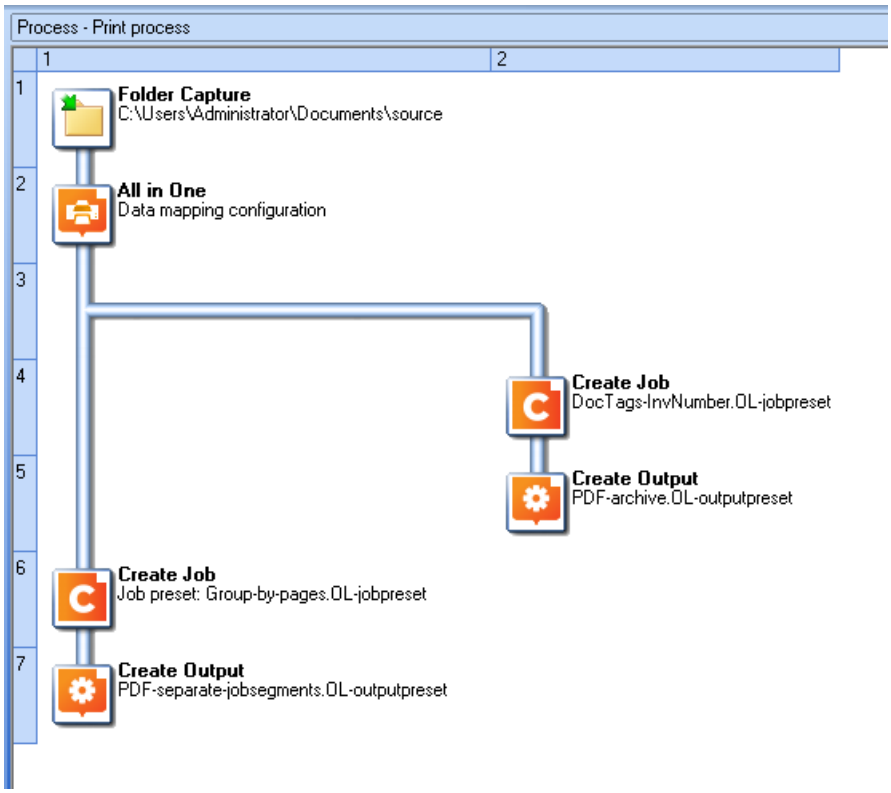
Tip: Dragging an output task - such as the Create Output task - between two tasks in a process and adding it as an Output task creates a new branch with the respective task at its end.

3. In the **Plug-in Bar**, in the OL Connect tab, locate the **Create Job** task and drag and drop it on the new branch. The **Create Job Properties** window opens.
4. Set the Create Job task to use the **DocTags-InvNumber** Job Preset.

5. In the *Plug-in Bar*, in the OL Connect tab, locate the **Create Output** task and drag and drop it on the 'Unknown' output task.



6. Set the Create Output task to use the **PDF-archive** Output Preset. The final process looks like this:



Debugging the process

Before sending the configuration to the OL Connect Workflow service, debug the process using the Step button.

- Pay attention to the **name** of the job file and metadata file: they change once the process has entered the branch. The job file and metadata file get copied when the process enters a branch. When the main process resumes after the branch has completed, it does so with the original files.
- Note that branches are not executed simultaneously. When a process branches, steps in the branch are executed first.
- After the branch has been executed, the main process continues.

The process should now output one PDF per invoice in addition to the PDF files that contain the grouped invoices.

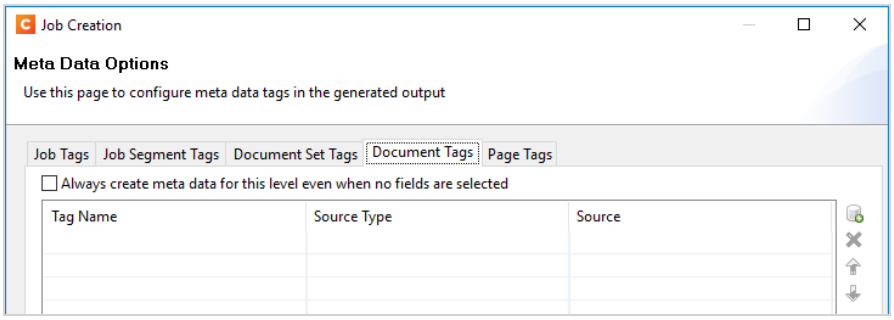
Creating the Print Presets

The branch in the print process that archives the invoices needs a different Job Preset and Output Preset than the main process.

Creating the Job Preset

This Job Preset doesn't group the documents, but provides the invoice number for each invoice, so that the invoices can be saved using their invoice number in the file name.

1. Launch the Designer.
2. Choose **File > New > Presets > Job Preset**.
3. Select the same data mapping configuration as before.
4. Check **Include meta data.**
5. Click **Next**. The Meta data options appear.

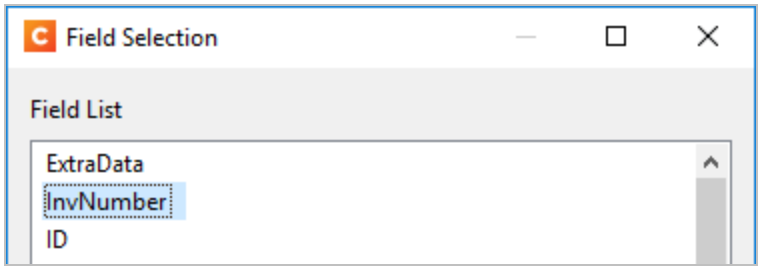


Each tab in this dialog represents one of the five levels in a (PDF or AFP) output file: job, job segment, document set, document and page. A Job Preset can provide the Output Preset with tags on each of these levels.

Note: These meta data tags can only be used in an Output Preset. They do not appear in the metadata file in Workflow.

6. On the *Document Tags* tab, click the Add meta data button and select **Add field meta data**.

7. Choose **InvNumber** from the list of data fields and click **OK**.



8. Click **Finish**.
9. Save the Job Preset as **DocTags-InvNumber.OL-jobpreset**.

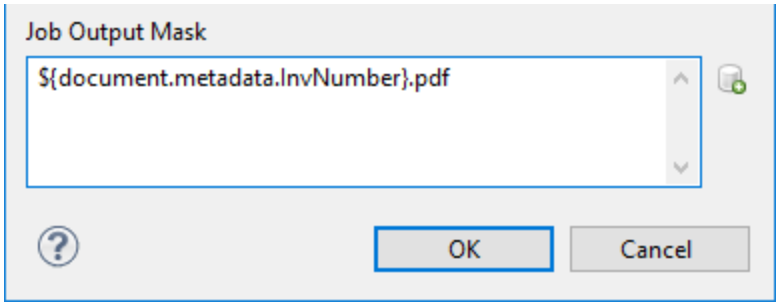
Creating the Output Preset

The Output Preset that archives the invoices must be configured to store each invoice in a folder, using the invoice number in the file name, and to create one file for each document using the meta data tag provided by the Job Preset in the file name.

This Output Preset splits the groups of content items and outputs them to different jobs/files.

1. Choose **File > New > Presets > Output Preset**.
2. Select **Generic PDF** in the *Printer model* drop-down.
3. Select **Directory** and enter a directory.
4. Click the *Options* button next to the *Output Mask* field.
5. Set **File Separation** to **Document** to make the process create one file per document.

6. In the Job Output Mask field, enter **${document.metadata.InvNumber}.pdf.**



   The part between brackets is a reference to the tag name defined on the Document tags tab in the Job Preset. (See also: Variables available in the Output)

   The specified tag points to a data field. Each time the Output Preset creates a file, it will look in the OL Connect database and use the value of that data field as the file name.

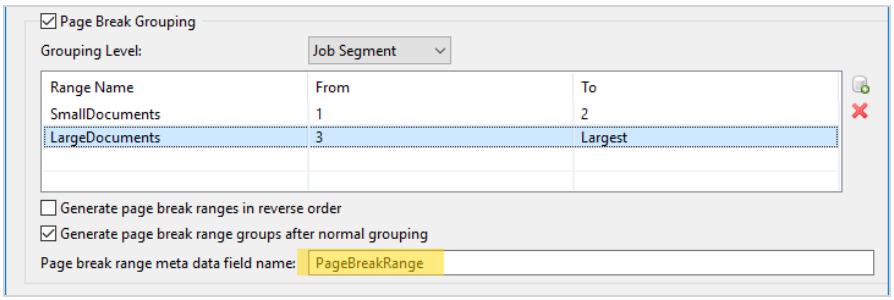
7. Click **OK**. Note that the **Separation** option is now checked.
8. Check **Print Virtual Stationery** to make the invoices look exactly like invoices printed on paper.
9. Click **Next** and verify that the separation level is set to **Document**.
10. Click **Finish**.
11. Save the Output Preset as **PDF-archive.OL-outputpreset**.
12. Send the Job Preset and Output Preset to Workflow (**File > Send to Workflow**...).

Exercises (optional)

A. Change the main process so that it outputs files named after their page range.

Hints:

- In the *Grouping options*, give each page range a name.



- Use the Page break range meta data field name in the output file mask: **${segment.metadata.PageBreakRange}.pdf**

B. Change the main process so that it groups invoices by state and creates output files named after the respective state (Alberta.pdf, etc).

Hints:

- Create a **Group-by-state** Job Preset; see Part 2, exercise 7A.
- Add the **State** field as a Job Segment Meta data tag to the Group-by-state Job Preset.
- For the Output Preset, open the **PDF-separate-jobsegments.OL-outputpreset** and change the output file mask to **${segment.metadata.State}.**

OL Connect projects

An OL Connect project is an automated process, or combination of processes, in which the OL Connect Server and Database are used. (For an overview of the architecture of the OL Connect software, see "OL Connect: a peek under the hood" on page 16).
Typically, an OL Connect project aims at automating (part of) a company's communication with its customers, suppliers, or other parties. Data is received - in whatever form -, processed, stored, and used in communications which are sent out through one or more output channels (print, email, web, etc), immediately or at a scheduled time.

Automation with Workflow

The process automation server included in OL Connect is called **Workflow**.
The Workflow server (also sometimes referred to as the '>*Watch service*') executes processes independently, after a configuration has been uploaded and the services have been started.

Usually, all of OL Connect's design tools are involved in the development process:

- Input data is extracted from a data source using a **data mapping configuration**, created with "The DataMapper" on page 336.
- Communications are created by merging the data with one or more **templates**, designed with "The Designer" on page 587.
- If the output channel is print, **Print Presets**, also created with the Designer, determine how the documents are printed.
- The pieces come together in an automated process, e.g. a **Workflow configuration**, created with the Workflow Configuration Tool.

Each of the tools has its own Online Help, but if you're new to either Workflow or the OL Connect software in its entirety, this chapter will help you get started on building OL Connect solutions.

Tip: Sample Project

The OL Connect software comes with a number of Sample Projects that generate a Workflow configuration, and any Connect templates, data mapping configurations, and Print Presets required to make the project work. For more information, see "Sample Projects" on page 1185 in the online help.

Typically, in an OL Connect project there is a number of **Workflow processes** that communicate with the Connect Server, Database and/or File Store through one or more of the specially developed **OL Connect** tasks.

For help on building these Workflow processes see "Workflow processes in OL Connect projects" on page 301 and "OL Connect tasks" on page 303.

Using the REST API

The OL Connect Server, Database and File Store can also be accessed through the **REST API**, i.e. in a script. The script could be part of a Workflow process (i.e. a Run Script task), but it could also be in a web page or portal, which means you could potentially access the OL Connect Server, database and File Store directly, without having to go through a Workflow process, or with another automation tool such as Node-RED.

The OL Connect REST API gives access to a number of areas including its processes, data entity management and File Store operations.
These services can be used to perform various interactions with the OL Connect server such as:

- Upload and manage data files, data mapping configurations and templates in the File Store.
- Create, manage and find data entities internal to the OL Connect Server.
- Create and monitor processing operations within the OL Connect Server.

For examples and the REST API reference see The Connect REST API CookBook.

Tip: The Designer can send templates, data mapping configurations and print presets to a OL Connect Server; see "Sending files to Connect Server or to another server" on page 594.

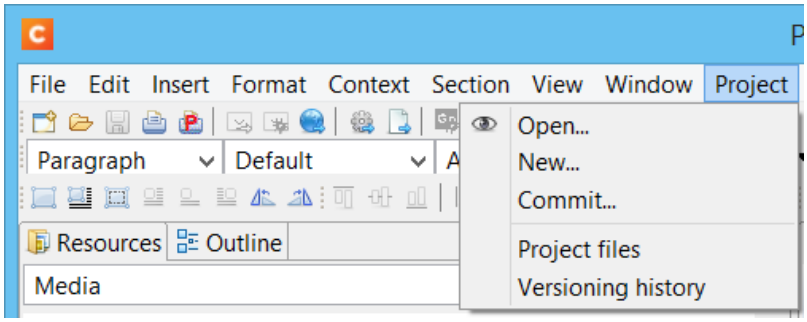
Versioning

You can create versioned projects. A versioned project retains a record of the changes made to any file in the project. You can review changes, and also revert to a previous version of a project, if necessary.

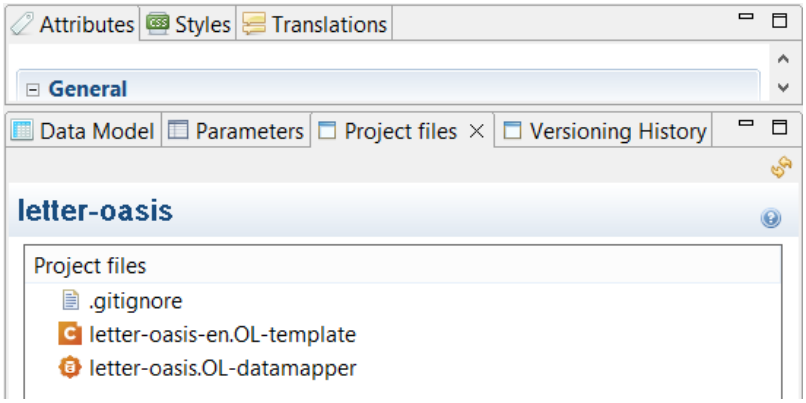
Note: Versioned projects are created from the Project menu at the top of the Welcome (or Home) screen. Projects created with the Sample Projects wizard are not versioned.

As you work on your project files, you can *commit* the files in the project at any time. When you commit the project, a snapshot of the project is saved as a *version*. Any additional information associated with a resource in the project is recorded alongside the file.

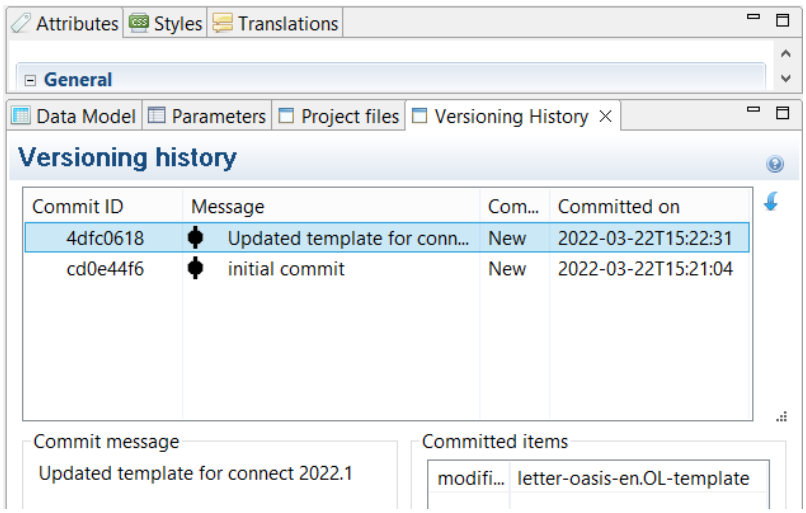
Use the **Project Files** option in the **Project** menu to display the list of resources in the project.



The files comprising a Project display in the **Project Files** panel.



Use the **Versioning History** option in the **Project** menu to display the history of all changes. The **Versioning History** panel contains a complete record of who did what, on which date, and for which reason.



You can elect to revert to a version from within the history if something turns out to be flawed in your current project.

Both the **Designer** and the **DataMapper** use Git integration to maintain the history of a **Project**.

> **Note:** The first time you open or create a new project you may be prompted to enter your name and email. This is used to identify changes in the project history.

Versioned projects

A *versioned project* is essentially a folder that contains a collection of related files whose changes are tracked. When you create a versioned project, all changes to the project are recorded and stored in the *project history*. You can use the project history to review changes made to the project, and even revert to a previous version if something goes wrong.

As you work on your project files, you can *commit* files in the project at any time. When you commit files in the project, a snapshot of the project is saved as a *version*, along with your name, your commit message, and the date. The status of each file - whether it was added, modified, etc. - is recorded as well.

OL Connect uses **Git** integration to maintain the history of a versioned project in a local repository. Projects can be created and committed from within OL Connect **Designer** and **DataMapper**.

Creating versioned projects

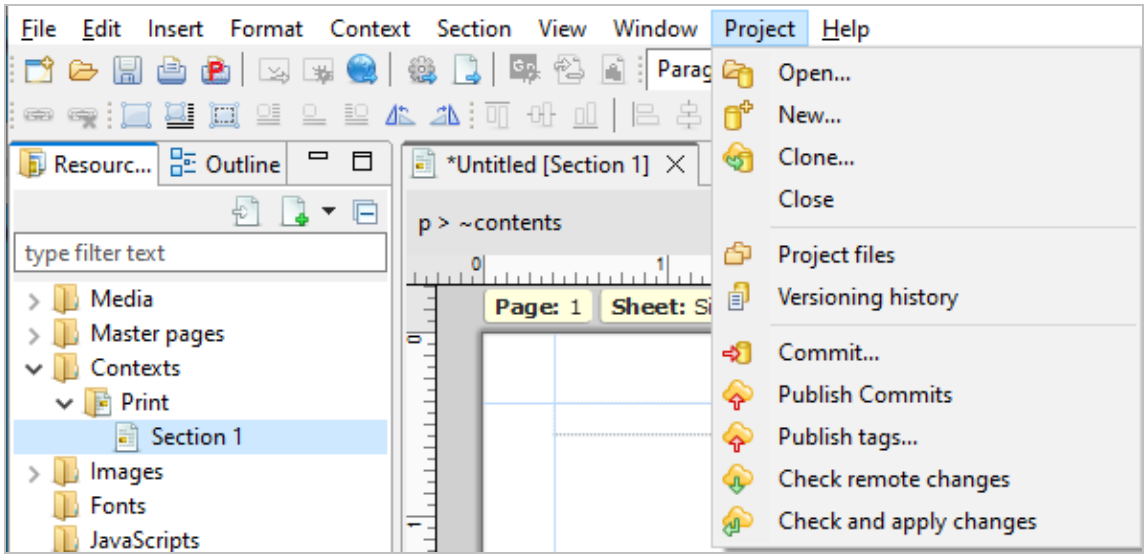
Versioned projects are created from the **Project** menu at the top of the screen in OL Connect Designer and Datamapper.
You have two options:

- **Create** a new versioned project that is only stored *locally*.
- **Clone** a project that exists on a cloud-based platform into a local repository.

How to clone a project is explained in "Versioned projects in the cloud" on page 258.

This is how you create a new *local* project:

1. Select **Project > New** from the menu at the top of the screen.



   The first time you create a project you may be prompted to enter in your name and email address. This information is required by Git, and your name is used to identify changes in the project history.
   Via the menu: **Window > Preferences > Versioning**, you can change this information at any time.

2. Each versioned project needs to be stored in its own **folder**. Select the folder in which to store the project. The folder name will actually become the name of the project.

   The software suggests to store all versioned projects in the **OL Connect** folder that is located in the **Documents** folder, with each project having its own folder under **Documents/OL Connect**.

   Tip: If the folder doesn't exist yet, right-click in the right part of the dialog, where the existing folders are listed, and select **New**. Type the name of the folder that you wish to create and click **OK**.

   Note: If you select an existing folder, be aware that any file that is present in that folder will automatically become part of the project.

3. Now you can create a new template, data mapping configuration or print preset (**File > New**) or open an existing one (**File > Open**) .

   As soon as you **save** a file in the project folder, it becomes part of the project.

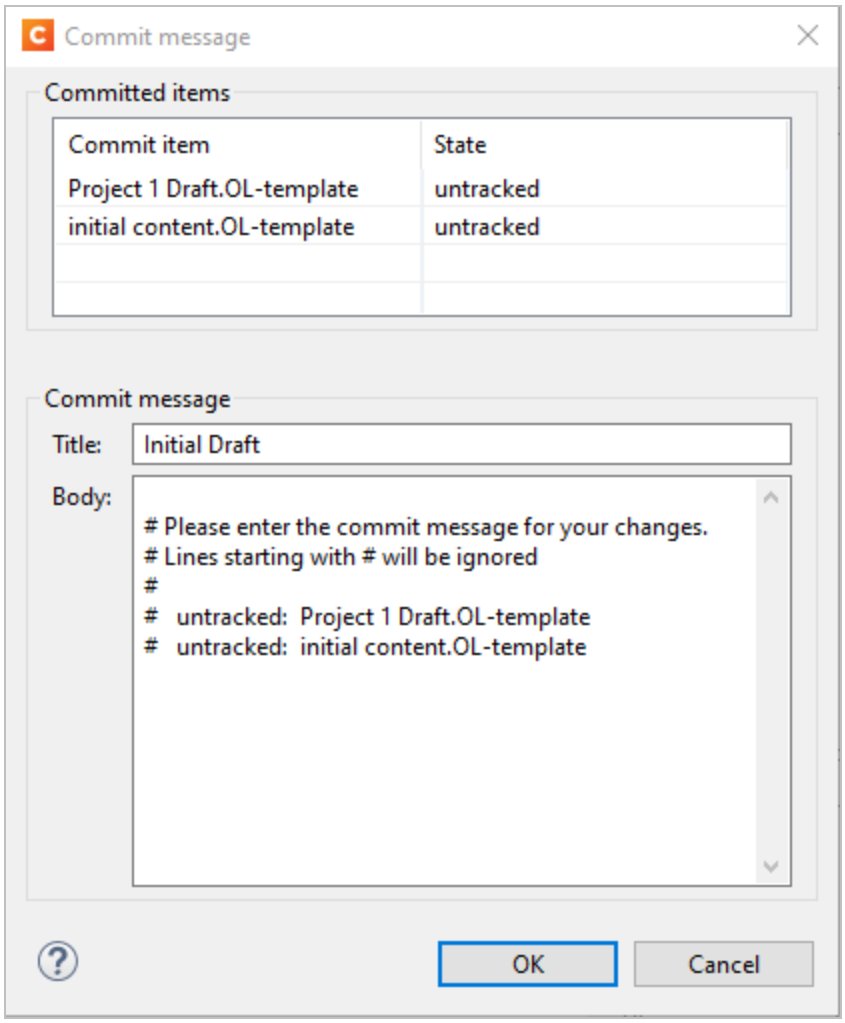
Tip: You can add a Workflow configuration to a versioned project, simply by saving it in the project folder and then committing the project.

4. To record this version in the repository, select **Project > Commit**. This records the project files as a new version in the repository. Remember to save any changes before you commit.

   To commit a single changed file, right-click the file in the **Project files** panel and select **Commit**.

   Note: The *.gitignore* file in the project folder tells Git which subfolders and files should not be committed.

5. Enter information to describe the changes, then select **OK**. Describing the changes is important as it will allow you, later, to readily identify each version as you browse through the project history.



Once created or opened, a versioned project remains open until it is closed via the menu: **Project > Close**, or until (a file from) another project is opened. As long as a project is open, the software will open the project folder every time you want to open or save a file.

> **Note:** Versioned projects have a hidden *.git* folder. Do not remove this folder. It contains the version history and, in the case of an online versioned project, information about the remote repository.

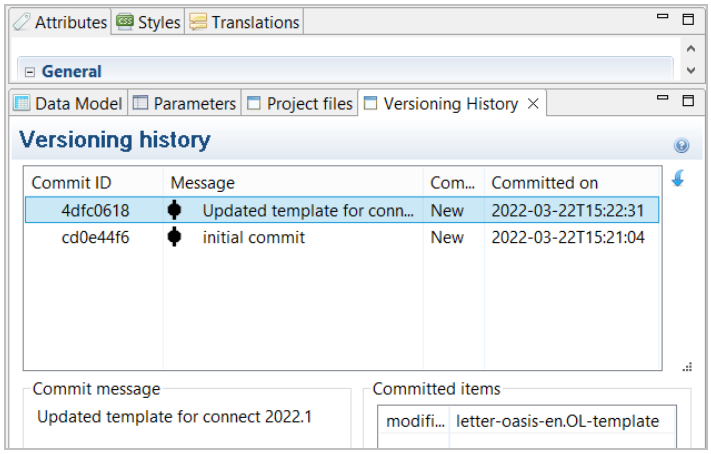
Viewing project history

The project history contains the records for all of the changes made to each resource in the project. When you commit a project, a snapshot of the project resources is stored as a version. You can review the versions to see what changes were made, who made them, and when they were made.

To see the history of the project:

1. With the project open in Designer, select **Project > Versioning history** from the menu bar.
2. The history displays in the Versioning History tab.

You can select a version from the list to display more information about the changes made in that version.



Restoring a version

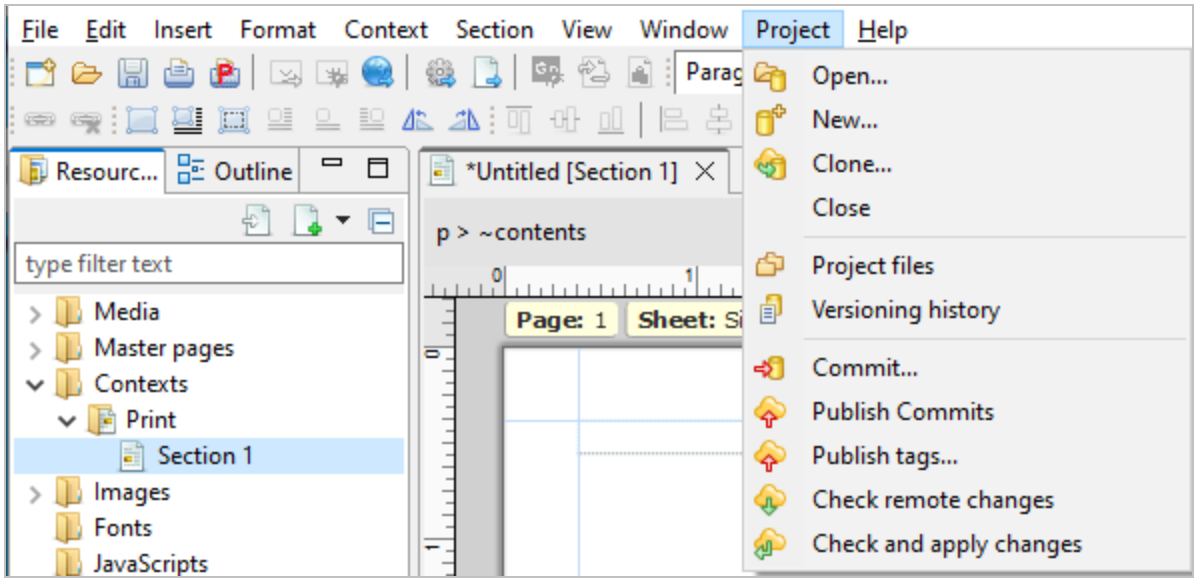
To **restore** a previous version, select the version, then select the Restore icon. The project is reset to the state it was in when that version was committed.

Reverting individual files

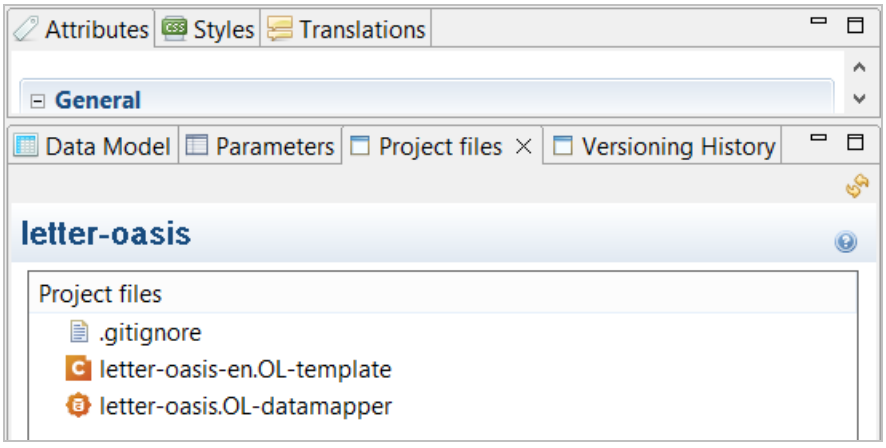
Instead of restoring an entire version you may revert individual files to a previously committed version. This is done in the **Project files** panel. Right-click the file and select **Revert**. Remember to commit the project so that the reverted files become part of the current version.

Viewing project content

Select the **Project files** option from the **Project** menu to display the list of resources in the project.



The files comprising a project display in the **Project files** panel.



Double-click a file to open it.

Right-click a file to open the contextual menu. From there you can not only open, commit or revert it, but also select to **view its properties** or to **open the folder** in which the file is located in Windows' File Explorer.

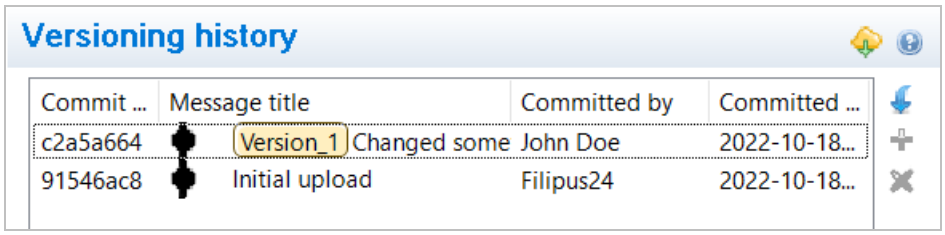
Using tags

The **Tag** functionality allows any commit to be identified with a user-defined label. Think of it as setting a flag on your versioning history whenever you want to highlight that, at that specific stage, something is more remarkable than on other commits. For instance, you can mark the latest commit with a tag named *Version_1*, thereby indicating that this is the first version of the entire project that goes into production.

To do that, simply right-click on one of the commits in the **Version History** and select **Set Tag**, then enter a tag name.

Note: Tag names must be unique. A different capitalization does not make a name unique. Spaces are not allowed.

The Versioning history panel displays any tags immediately before the message title:



You can add as many tags as you like, and you can delete them (right-click the tag, then click **Delete**) and re-use them on a different commit (which is the equivalent of moving a tag).

Versioned projects in the cloud

If a *versioned project* is stored on the cloud, you, and other members of your team if you so decide, can work on the project from anywhere and download the very latest version of the files at all times.

OL Connect Designer and Datamapper let you *clone* a Git repository that exists in the cloud to a local repository, and keep the online and local projects in sync. This topic explains how to do that.

For instructions on how to create and maintain a versioned project that is only stored *locally*, see "Versioned projects" on page 253.

> **Note:** Options to upload a local project to a new or existing online repository, and to branch (fork) and merge cloud-based versioned projects in OL Connect will be added in a later release.

Before you start

Before you can start working with versioned projects in the cloud, there are a couple of things you need to do. These steps are only needed once.

Create an account

OL Connect integrates with **Git** to maintain the history of a versioned project. Git is a Version Control System (VCS) that enables developers to save snapshots of projects over time. Web-based platforms like GitHub, BitBucket and Azure DevOps offer a Git repository hosting service that allows projects to be worked on collaboratively.

The first thing you need is an **account** with a Git repository hosting service. GitHub, BitBucket and Azure DevOps have been tested with OL Connect. Others haven't been tested, but if they are Git-based, they should work the same way.

Obtain a token

Secondly, you have to log on to your Git repository hosting service and go into your profile settings to obtain a **token**. This is called an *app password* in BitBucket, or a *personal access token* in GitHub and Azure DevOps. OL Connect will use it as the password to your account when it needs to read or write information in the online repository.
The token can control which operations are allowed. Allow all operations, so that OL Connect will be able to perform all tasks as needed.
When the token has been generated, it is displayed only once.

> **Example:** `go3_bGyW5GlIQx4RWsttlkmxPJZBcFY2UH36B8VP`

Make sure to copy the token somewhere, because you won't be able to see it again!

Set up OL Connect

The user name of your account with a Git repository platform, and the personal access token or app password obtained from that same platform, must be entered in OL Connect, so that the software can access your versioned projects in the cloud.

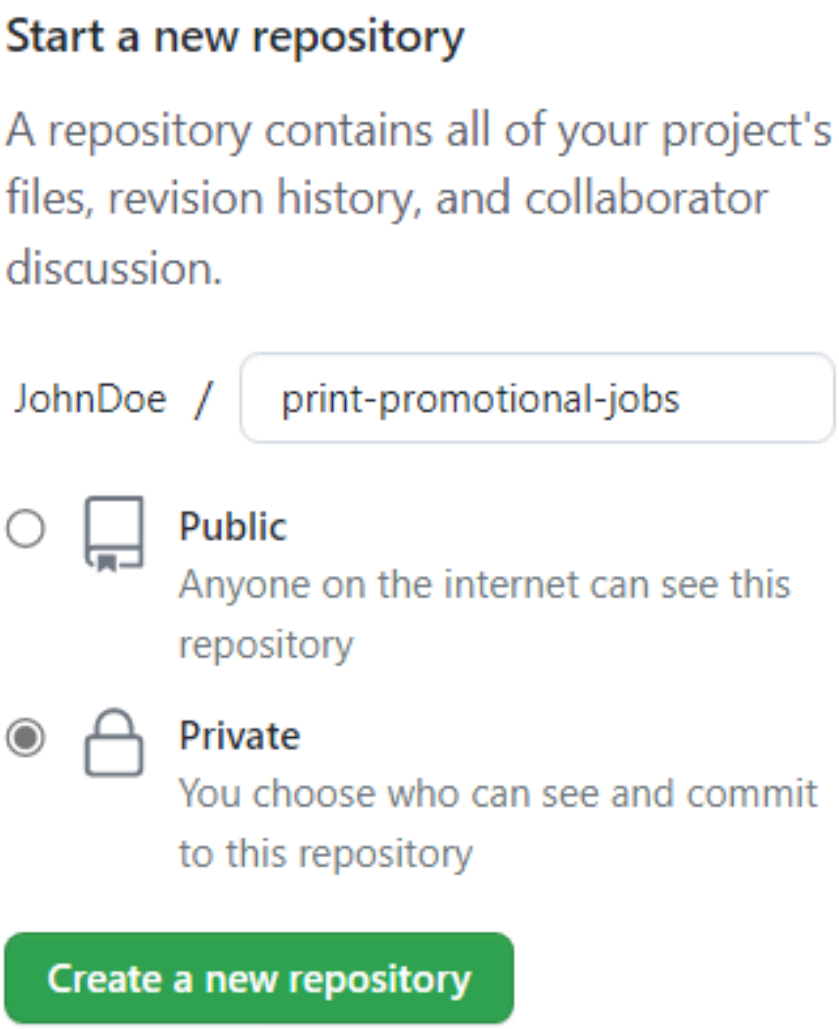
In the Designer, select **Window > Preferences** in the menu, go to **Versioning** and provide the required information under **Remote repository**.

Creating a cloud-based versioned project

Once you have created a Git repository in the cloud, you can *clone* that project in OL Connect and keep it in sync with the online project.

Here are the steps to create and clone a cloud-based versioned project:

1. Log on to your Git repository hosting service and create a new project that will host the files for a new project. In BitBucket, for example, it looks like this:



2. Make sure that the remote repository contains at least one file. If it is completely empty, add one file to it. This could be an empty *readme.txt* or *.gitignore* file.
   With BitBucket this isn't necessary; a new Bitbucket repository always has a *.gitignore* file.

   Note: A remote repository that is completely empty will cause an error when cloning it in OL Connect. This will be fixed in a future release.

3. Once the new repository has been created, you are presented with the URL of the new project. Copy the URL; you will need it in a following step.
4. Open OL Connect Designer and select the **Clone** option in the **Project** menu.

   The first time you create a project you may be prompted to enter in your name and email address. This is required by Git. Your name is used to identify changes in the

project history.
This setting is also accessible via the menu: **Window > Preferences > Versioning**.

5. Enter in the **URL** of the remote repository.
6. In the **Destination** field, enter the name of the folder where the local copy of the versioned project should be stored.
   The software suggests to store all versioned projects in the **OL Connect** folder that is located in the user's **Documents** folder, with each project having its own folder under **Documents/OL Connect**.
   The folder name will actually become the name of the project. By default, the local repository is created with the same name as the online repository. For example, if the URL to the online repository is 'https://github.com/jdoe/promotional-print-project' then the destination folder will be 'C:\Users\<username>\Documents\OL Connect\promotional-print-project '. You can adjust the local project name if you want to.
7. Click the Refresh icon next to the **Branch** field, to update the list of branches that are available in the remote repository, and select the branch that you want to clone from the drop-down list.
8. Click OK to clone the project. The project will be stored in the destination folder.

   Now you can create a new template, data mapping configuration or print preset (**File > New**) or open an existing one (**File > Open**) .

   As soon as you **save** a file in the project folder, it becomes part of the project.

   Note: Versioned projects have a hidden *.git* folder. Do not remove this folder. It contains the version history and, in the case of an online versioned project, information about the remote repository.

Tip: "Sample Projects" on page 1185 are locally stored versioned projects. To turn a Sample Project into an online versioned project, you could create a new project in your online repository and add all of the Sample Project files to the online project before cloning it in OL Connect. Include the local *.gitignore* file, but don't upload the existing (hidden) *.git* folder.

Keeping the local and online projects in sync

Before synching your local repository with the online repository, select **Project > Commit**. This records the project files as a new version in the local repository. Remember to save any

changes before you commit.

Publishing changes

Publishing the local changes to your online repository is done via the **Publish Commits...** option in the **Project** menu. A confirmation dialog is displayed, stating that the changes have been published, or, if there are new changes in the remote repository, that you have to download the changed files first.

> **Note:** The *.gitignore* file in the project folder tells Git which subfolders and files should not be committed.

Tags that you add to versions have to be published to the remote repository via the **Project > Publish tags...** option.

See also: "Using tags" on page 258.

Downloading remote changes

When you select **Project > Check and apply changes** OL Connect retrieves changed files from the remote repository and downloads them to your local project. This updates your local project with the most recent version of files in the remote repository.

You can also check for remote changes without immediately downloading the changed files: select **Project > Check remote changes** from the menu bar.

Conflicts

A 'conflict' arises when both the remote version and the local version of a file have been changed. If there are conflicting files when you download remote changes, you will be asked whether you want to overwrite the local files ("Keep Remote") or discard the remote changes ("Keep Local"). Before a local file is overwritten, a backup of the local file is made.

Preventing conflicts

If multiple people have access to the files in a remote project, check for remote changes on a regular basis, to make sure that you work with the latest files. It is important to prevent conflicts in files, since there is no way or tool yet to resolve conflicts in OL Connect files like templates and data mapping cofigurations.

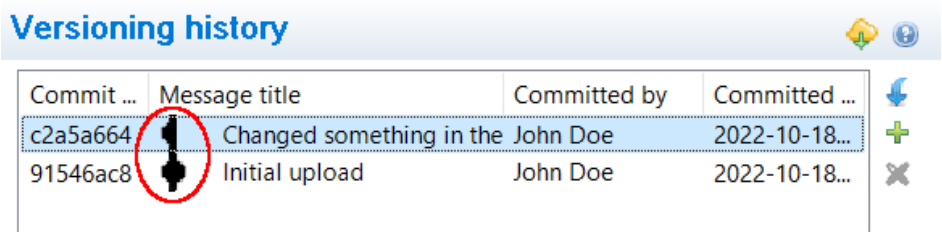
OL Connect automatically checks for remote changes and displays a message when one or more files in the remote repository have changed. The setting **Minutes between remote checks** determines how often OL Connect will check the remote repository for changes if a cloud-based versioned project is open. You can find this setting in the preferences; from the menu at the top, select: **Window > Preferences > Versioning**.

Synchronization information in the project history

Whenever you publish commits to, or pull changes from the online repository, the versioning history panel (**Project > Versioning history**) is updated to reflect whether the local and remote versions are in sync.

A full circle before the message means the local and remote versions of that commit are in sync.

Half a circle on the left hand side of the axis indicates the commit was made locally but has yet to be published to the online repository.



If someone were to make changes to the online version that the local version doesn't have yet, then that half circle would be on the right hand side of the axis.

See also: "Viewing project history" on page 256.

Sample Projects

A Sample Project generates a small Connect solution that is ready to be tested and deployed. The solution contains a specific Workflow configuration, as well as the Connect templates, data mapping configurations, and any Job Presets and Output Presets that are used in that configuration.

This chapter describes the Sample Projects and the projects that they install. It will help you install, comprehend and customize the projects.

For generic information about Connect solutions and their components, see "OL Connect projects" on page 250.

> **Note:** The projects require that the Connect Server and Connect Workflow be installed on the local machine.

These are the projects that can be installed with a wizard.

- **Print promotional jobs**. This project generates a batch of basic personalized letters, in the format that was selected in the wizard: PDF, PCL or PostScript Level 3. (See: "Sample Project: Print Promotional Jobs" on page 279.)
  The Workflow configuration uses the All In One plugin (see "Print processes with OL

Connect tasks" on page 306).

- **Print transactional jobs**. This project creates print content items - invoices - and uses multiple Create Output tasks to generate various print output variants:
  - A single PDF for the entire job (in which the invoices are grouped per customer).
  - One PDF per customer.
  - One PDF per invoice.

  (See: "Sample Project: Print Transactional Jobs" on page 285.)

  The Workflow process implements the typical Print plugins (see "Print processes with OL Connect tasks" on page 306).
- **Basic web page**. This project serves a simple web page, personalized via URL parameters. (See: "Sample Project: Serving a Web Page" on page 296.)
- **Submitting data with webforms**. This project illustrates how to serve multiple web pages, one of them a web form, using a single Workflow process. It stores submitted data in the Data Repository. (See "Sample Project: Submitting Data with Web Forms" on page 291.)
- **Basic email setup**. This project sends email messages with attachments. Pre-rendered PDFs are attached based on the provided sample data. (See "Sample Project: Basic Email" below.)
- **Capture OnTheGo Timesheets**. This project deploys and serves COTG Timesheet forms. Additional processes capture the submitted data and output them in JSON, XML and PDF. In order to create the PDF, the data are merged with a Connect template. (See: "Sample Project: COTG Timesheets" on page 272.)

Tip: All projects created with the Sample Projects wizard are *local versioned projects*. Any time you *commit* files in the project, a snapshot of the project is saved as a *version*. See "Versioned projects" on page 253 for more information about this feature.

Sample Project: Basic Email

The Basic Email Sample Project creates an OL Connect project that sends **emails** with two attachments: a Return and Refund Policy (PDF) which is the same for all emails, and a delivery note (PDF) which is attached to the email dynamically. The email has a *mailto* link.

Installing the project

To start the Sample Project, select **File > New > Sample Projects > Basic Email** from the menu. (See also: "Basic Email - Sample Project" on page 1186.)

Alternatively, open the **Welcome** screen, click **New** at the left, then **Project** at the right and select the correct type of project.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

Note: In order to use the project, OL Connect Server and OL Connect Workflow must be installed on the local machine.

The wizard lets you select the **folder** in which you want the solution to be installed. Since Sample Projects are *versioned* (see "Versioned projects" on page 253), the software suggests to store it in the **OL Connect** folder that is located in your **Documents** folder. By default, all versioned project have their own folder under **Documents/Connect**.

In the selected folder, the Sample Project will create two subfolders: Configurations and Workspace.
The project's resource files are saved to the **Configurations** folder.
The **Workspace** folder is used for debugging or running the solution. It has an In folder that may be used to monitor incoming data and an Out folder to write output files to.

The selected folder's path is saved to a global variable in the Workflow configuration (see "The Workflow configuration" on page 268).

The variable is used in the data mapping configuration (see "Project details" on the next page).

In addition you have to make the settings for outgoing mail (for the details see "Basic Email - Sample Project" on page 1186).
These settings are copied to the Create Email Content task in the Workflow process.

Finally, enter the **username** and **password** that will allow the software to access the Connect Server.

Testing and running the project

Once the Sample Project has finished the installation, the project is ready to be tested.

1. If the **templates** and **data mapping configurations** haven't been sent to Workflow yet, do that first (see "Sending files to Workflow" on page 593). This requires that the **Workflow service** is running (see Starting the Workflow service in Workflow's Online

Help).

2. Locate the Workflow configuration in the project folder and open it in OL Connect Workflow.
3. Select the **em_basic_sending_email** process
4. Open the Debug ribbon and click **Run**.

The process should send the email messages along with a delivery note and the Return and Refund Policy to the email address entered in the Sample Project. This is the sender's address.

Running the project

Having tested the project, you will be ready to send it to Workflow; see Saving and sending a Workflow Configuration in Workflow's Online Help.
The project will run when you copy the **Sample Data.xml** file from the Configurations\Data folder to the **Workspace\In** folder.

By default the project will still send all the emails to the sender's address. To change this:

1. Open the Create Email Content task and select the Email Info tab; then uncheck the Send emails to sender (test mode) option.
2. Replace the test data with real data:

a. Open the **Sample Data.xml** file. You will find it in the Configurations\Data folder.

b. Replace the email addresses in this file by real email addresses that you have access to.

c. Copy the modified sample data file from the Configurations\Data folder to the **Workspace\In** folder.

The emails should now be sent to the email addresses that you entered in the sample data file.

Project details

The email template

The email's design is in the **EM_BASIC Email Message** template. It contains one **Email section** (see "Email templates" on page 669).

Styling is done via **style sheets** (see "Local formatting versus style sheets" on page 879). The style rules are in the context_htmlemails_styles.css file. Note how they use HTML tags, IDs and classes to select elements. (See also: "Selectors in OL Connect" on page 1083.)

The @media rules make the email responsive (see also: "Designing an Email template" on page 660).

> **Note:** Even though email clients do not read CSS files, CSS files can be used with the Email context in the Designer. See "Using CSS files with HTML email" on page 662.

Two attachments are added to the email, in different ways.

- The Return and Refund Policy PDF is stored in the template (in the Images folder). Right-click the Email section and select **Attachments** to see how this is attached to the email.
- The delivery note is a dynamic attachment, based on a data field. It is attached to the email by the **Attachment** script. To learn how to create dynamic attachments, see "Dynamic attachments: creating file names based on data fields" on page 680.

All scripts are listed on the Scripts pane. You can double-click to open them.

> **Tip:** Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

- Some data are added to the email via simple **text scripts**. Such scripts search the template in order to replace a certain text that is surrounded by @ (e.g. @First@) by the correct data. (See: "Variable data in text: scripts and placeholders" on page 945.)
- The To and Subject scripts apply to email fields. (Click **Email Fields** at the top of the email to expand all email fields.) For information about this kind of scripts, see: "Email header settings" on page 672.
- Finally, there are two custom scripts:
  - The **Personalize Support link** script adds the order number to the 'support team' link (which is a *mailto* link).
  - The **Year** script puts the current year in the footer.

  For more information about writing custom scripts, see: "Writing your own scripts" on page 1064.

The Workflow configuration

The project's Workflow configuration, **Sending Email**, contains just one process. It is a simple OL Connect Email process (see "Email processes with OL Connect tasks" on page 305).

To open the configuration, open Workflow, click the Workflow button, select **Open** and browse to the project's **Configurations\Resources** folder.

Double-click on a task to see its properties.

- The **Folder Capture** task reads the project's Workspace path from a global variable. The value of that variable is set by the Sample Project when it installs the project.
- The **Execute Data Mapping** task extracts data from the sample data file and outputs them in the Metadata.
- The **Create Email Content** task creates the emails, using the Metadata as data source.
  Note that it sends the emails in **test mode**, which means it will send the emails to the sender.
  The settings on the Email Info tab are taken from the Sample Project.
- The **Delete** task is an Output task that does nothing, actually; it doesn't even remove anything. This step is sometimes useful when running a project step by step in **Debug** mode, as it forces preceding tasks to return their output to the Workflow process. The Create Email Content task, however, does not return the generated emails to the Workflow process.

The data mapping configuration

The Workflow process uses a data mapping configuration, made with "The DataMapper" on page 336: **EM_BASIC Data**.

To open the data mapping configuration, open the Designer, select **File > Open** from the menu and browse to the **Configurations\Resources** folder.

This data mapping configuration will of course only work with the appropriate data files. It was designed for XML files that are structured like this file: **Sample Data.xml** (in the **Configurations\Resources\Data** folder).
The sample data yields 5 records with customer data including a detail table with invoice details.

Much of the information in the extracted records isn't used in the email, but was used to create the delivery notes.

The email addresses are used in the template (in the **To** field), but ignored in the Workflow process because it sends emails in test mode (see "The Workflow configuration" on the previous page).

Using a Workflow variable

The path to the delivery note is constructed via JavaScript:
automation.variables.em_basic_workspace + '\\Delivery Notes\\' +
record.fields.OrderNumber + '.pdf';

The script uses a **property**: automation.variables.em_basic_workspace, which is defined in the **Preprocessor** step. (See also: "Properties and runtime parameters" on page 410.)
In this case, the variable holds the project's path. The order number in the path is extracted from the sample data.

> **Note:** In order to use the value of a Workflow variable - an 'Automation variable' - as a property in the Preprocessor step, the property and the Workflow variable must have the exact same name.

To see the script, click on the DeliveryNote field in the Data Model pane at the right; then take a look at the Step Properties pane (in the middle).

To see where the property is defined, click the Preprocessor step on the Steps pane (at the left); then look at the Properties section on the Step Properties pane.

Customizing the project

A project is a great starting point for building an OL Connect solution. This part explains how to make changes to the project.

Do you intend to expand the project into a solution where Workflow runs on a different machine that also has a different regional setting? Then indicate the desired encoding in the Designer preferences (see "Sample Project deployment settings" on page 1058) **before** installing the project.

Input data

If your input data is in a file, but the file is of a different type or has a different structure, create a new data mapping configuration to match it. (See "Creating a new data mapping configuration" on page 338.)
When it's finished, send the new data mapping configuration to Workflow (see "Sending files to Workflow" on page 593).
Then open the Workflow configuration, double-click the **Execute Data Mapping** task to open it and select the new data mapping configuration.

To capture input data from a different **source**:

1. Replace the Folder Capture Input task by the appropriate Input task. See: Input tasks in Workflow's Online Help.
2. Add a **Send to Folder** task directly after the new Input task and set its output folder to the Workspace\Debug folder (`%{global.pr_prom_workspace}\Debug`). This task writes the **job file** to a file, which can then be used as sample data file when creating a data mapping configuration and debugging the Workflow process.
3. Run the process once (see "Testing and running the project" on page 266) to capture the job file that contains the input data.
4. Use that file to create a new data mapping configuration, send the data mapping configuration to Workflow, and adjust the Workflow configuration, as described above.

Note: If the input data is JSON, you don't need a data mapping configuration: JSON data can be used in a template as is. See: "Adding JSON sample data" on page 937. However, if you want the data to be saved in the OL Connect database, put the XML/JSON Conversion plugin before the Send to Folder task to convert the JSON to XML, and create an XML data mapping configuration to extract the data.

The email

Often this type of email has a **Call to Action** button which redirects the user to the website of the vendor, where the user could log on to the site and follow an online payment process, for instance.
To get an example (or to start over) you could use one of the template wizards that start an 'action email'; see "Email Template Wizards" on page 665.

You could add text, images and other elements to the email (see "Content elements" on page 761) and change the layout (see "Styling and formatting" on page 879). Keep in mind though that there are special design standards for HTML email (see "Designing an Email template" on page 660).

In order to further **personalize** the email, open your data mapping configuration (or JSON data, if the input data will be in that format; see "Adding JSON sample data" on page 937) and use the data fields to personalize the email. (See: "Personalizing content" on page 923.)

Tip: The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.

Once the template is ready, send it to Workflow (see "Sending files to Workflow" on page 593).

Finally, in Workflow, adjust the process: double-click the **Create Email Content** task to open it, and select the new template. This is only necessary when the file name has changed.

Workflow configuration

The current Workflow configuration is very simple. In reality, a process that generates email output will be part of a larger project, in which, for example, invoices are produced in a separate process, stored in a folder and attached to an email at a later time.

For general information about processes in Workflow see About Processes and Subprocesses, in the Online Help of Workflow.

Sample Project: COTG Timesheets

The COTG Timesheets Sample Project creates a **Capture OnTheGo project** that creates a **web page** with a web **form** and serves it to the COTG app. When the form is submitted, the project handles the submitted data by creating an XML, JSON and PDF output file in a folder.

Installing the project

From the menu, select **File > New > Sample Projects > COTG Timesheets** to start the wizard.

Alternatively, open the **Welcome** screen, click **New** at the left, then **Project** at the right and select the project type.

> **Tip:** Click the Home icon at the top right to reopen the Welcome screen.

The wizard lets you select the **folder** in which you want the solution to be installed. Since Sample Projects are *versioned* (see "Versioned projects" on page 253), the software suggests to store it in the **OL Connect** folder that is located in your **Documents** folder. By default, all versioned project have their own folder under **Documents/Connect**.

In the selected folder, the Sample Project will create two subfolders: Configurations and Workspace.
The project's resource files are saved to the **Configurations** folder.
The **Workspace** folder is used for debugging or running the solution. It has an In folder that may be used to monitor incoming data and an Out folder to write output files to.

The selected folder's path is saved to a global variable in the Workflow configuration (see "The Workflow configuration" on page 276).

You also need to enter the **host**. This address will be used by the COTG app to download forms and submit data to the OL Connect Workflow service.
If you want to use localhost, the OL Connect Server, Workflow and COTG app must be installed on the local machine.

A valid **COTG account**, **Repository ID** and **password** are also necessary. You could use dummy data to create the Sample Project and finish the installation, but you won't be able to test the project.

For more information about the options, see: "COTG Timesheets - Sample Project" on page 1187.

Finally, enter the **username** and **password** that will allow the software to access the Connect Server.

Tip: If you own OL Connect (Professional or Enterprise edition), free COTG trial licenses may be available to you; see http://www.captureonthego.com/en/promotion/.

Note: Your network setup may make it impossible for the COTG app to communicate with the OL Connect Workflow service. The app needs to be able to communicate with OL Connect Workflow in order to download forms and submit data. Network and firewall settings may block these requests. Please contact your local IT department for advise on how this can be resolved in your setup.

Testing the project

Once the Sample Project has finished the installation, the project is ready to be tested.

1. If the templates and data mapping configurations haven't been sent to Workflow yet, do that first (see "Sending files to Workflow" on page 593).
   This requires that the Workflow service is running (see Starting the Workflow service in Workflow's Online Help).
2. Send the Workflow configuration to the Workflow service; see Saving and sending a Workflow Configuration in Workflow's Online Help.
3. Open the Workflow configuration from the project's Resources\Workflow folder.
4. Select the **cotg_ts_deploy_form** process, open the **Debug** ribbon and click **Run** to execute the process. This deploys a number of forms to the COTG account entered in the Wizard.

> **Note:** The process is pre-configured to use the sample data file, which comes with the project, when it runs in Debug mode.
> Alternatively you could copy the sample data file to the In folder.

5. Launch the COTG app and log in using the COTG account that you entered in the Sample Project.
6. In the app, open the Repository and download one of the forms.
7. From the app's Library, open the form.
8. Enter data and draw a signature.
9. Submit the form.

> **Note:** If the COTG app is unable to submit the data, you may have to change your network setup or remove the built-in restrictions in Windows 10 that prevent applications from sending data to the same computer. See "Installing the project" on page 272.

Project details

The templates

The form

The **COTG Timesheet Form** template contains a **Web context** with one Web section: **Section 1** (see "Web pages" on page 688 and "Forms" on page 842). The form has regular Form elements as well as COTG elements (see "Form Elements" on page 847 and "COTG Elements" on page 835).

The template was started with the Time Sheet Wizard (see "Capture OnTheGo template wizards" on page 717), which also provides the necessary JavaScript files and style sheets.

Most of the **scripts** are simple Text Scripts (see "Using the Text Script Wizard" on page 948). They put data in one of the form's fields, selecting the field by its ID.

> **Tip:** Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

- The **guid** script adds data to the `guid` field. This is a hidden field. It is only visible in the Source view.
- The result of the **form action** script is added to the form's `action` attribute. To see how that is done, double-click the script; then click **Options**.

- The **Year** script puts the current year in the footer of the page. This is a custom script. For more information about writing scripts, see: "Writing your own scripts" on page 1064.

The report

The **COTG Timesheet Report** template contains a **Print context** with one Print section: **Section 1** (see "Print sections" on page 629) and one **Master Page** (see "Master Pages" on page 648).

Personalization is mostly done via simple **text scripts**. Such scripts look for a text surrounded by @ (e.g. @city@) and replace that by the correct data. (See: "Variable data in text: scripts and placeholders" on page 945.)

The template also contains a **dynamic table** which is filled and expanded dynamically by the scripts in the **Table** folder. To learn how to insert and edit such a table, see "Dynamic Table" on page 974.

Note that this table does not use one of the default table styles, and that the style sheet with the default table styles is not present in the template. To add that style sheet to the template, insert a table using the Dynamic Table wizard.

The data mapping configurations

This project has two data mapping configurations, made with "The DataMapper" on page 336. To open one of them, select **File > Open** from the menu in the Designer and browse to the **Configurations\Resources** folder.

COTG Timesheet Form

The COTG Timesheet Form data mapping configuration is designed to extract data from the sample data file (Sample Data.xml).
The configuration also adds two fields to each record, using JavaScript: a unique ID (guid) and a field that contains the action of the Form.
Note that the COTG account is the same in all of the records in the sample data. It is inserted into the sample data by the Sample Project.

COTG Timesheet Report

The COTG Timesheet Form data mapping configuration is designed to extract the data that are submitted via the app. It first extracts the common fields, and then the information from the rows in the form, in a loop. For information about how to extract a variable number of similar elements from an XML file, see: "From an XML file" on page 399.

In order to create this data mapping configuration, the input data needed to be saved to a file first (see "Saving input as sample data" on page 278).
The sample data file is located in the **Configurations\Data** folder, but you will also see it when you open the data mapping configuration itself.

The Workflow configuration

There are four processes in the COTG Timesheets Workflow configuration. Some of them use the OL Connect tasks that are normally used in a Web process or a Print process (see "Web processes with OL Connect tasks" on page 309 and "Print processes with OL Connect tasks" on page 306).

Creating the forms

The **cotg_ts_deploy_form** process is triggered when any file enters the Workspace\In folder, but its Execute Data Mapping task expects an XML file that is structured exactly like the Sample Data.xml file (in the Resources\Data folder).
The Execute Data Mapping task outputs records in the Metadata.
Subsequently, the **Metadata Sequencer** task creates a loop; the rest of the process is performed as many times as there are records in the Metadata, for one record at a time:

- The **Set Job Infos and Variables** task reads data from the (current record in the) Metadata and puts them into variables.
- The **Create Web Content** task merges the record with the COTG Timesheet Form template.
- The **Send to Folder** task saves the form to the Forms folder in the project folder, using the value of one variable as the file name.
- Finally, the **Output to Capture OnTheGo** task sends information about the form to the COTG Server. Double-click the task to see how variables are used in its settings.

The **Delete** task is an Output task that does nothing, actually; it doesn't even remove anything. However, this step is useful when running the project step by step in **Debug** mode. Tasks will only return their output to the process - where it can be viewed - as long as they are followed by another task.

Serving a form

When the user chooses to download a form in the app, the app sends a request using the URL that was set for the form in the properties of the Output to Capture OnTheGo task (on the **Deposit** tab). The URL contains one parameter: a unique ID.
The **cotg_ts_serve_form** process receives the request and loads the form (if it exists). The

Load External File task replaces the job file with the form. At the end of the process the Server Input task returns the job file (the form) to the app.

Capturing data and generating output

When the user submits a form, the **cotg_ts_capture_data** process writes the job file - the request in the form of XML - to a Temp folder in the Workspace and sends a response code to the app. This way there is no need for the COTG user to wait while output is being generated. The response to the app can be sent right away.

The **cotg_ts_generate_report** process picks up the XML file from the Temp folder, converts it to JSON and saves it to a JSON file. Note how the **Branch** task keeps a backup of the job file. It is the backup file - the XML - that is handed down to the next Branch, where it is saved as an XML file.
Finally, the process extracts data from the job file and merges them with the COTG Timesheet Report template, generating a PDF.
All output files are stored in the Workspace\Out folder

Customizing the project

A project is a great starting point for building an OL Connect solution. This part explains how to make changes to the project.

Do you intend to expand the project into a solution where Workflow runs on a different machine that also has a different regional setting? Then indicate the desired encoding in the Designer preferences (see "Sample Project deployment settings" on page 1058) **before** installing the project.

The form

If you want to add inputs to the form and extract the submitted data, here's how to do that.

1. Open the Web section and add elements to the Web Form (see "Using COTG Elements" on page 729 and "Using Form elements" on page 698).
   You could also add text, images and other elements (see "Content elements" on page 761) and change the layout of the page (see "Styling and formatting" on page 879).
2. Save the template and send it to Workflow (see "Sending files to Workflow" on page 593).
3. Upload the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help) and let the process save the input data to a file (see "Saving input as sample data" on the facing page).

4. Use the saved file to expand the data mapping configuration (see "Opening a data mapping configuration" on page 342). Send the data mapping configuration to Workflow.
5. Send the Workflow configuration to the server.

Saving input as sample data

Saving input as sample data

Testing a process in Debug mode is only possible with a sample data file. The process is pre-configured to use the Sample Data.xml file located in the Configurations\Data folder. To create your own sample data file:

1. Locate the Workflow configuration in the **Configurations\Workflow** folder and open it in Connect Workflow.
2. Select the process.
3. Enable the **Send to Folder** step (step 2 in the process).
4. Send the Workflow configuration to the Workflow service (see Saving and sending a Workflow Configuration) and run it again, with a custom name value.
   The Send to Folder step will now write the input data - the job file - to a file in the **Workspace\Debug** folder.

When you select the file as sample file (on the Debug ribbon), it can be used to debug the process.

> **Tip:** The saved file can be used to create a data mapping configuration.

The report

Using different data in the report requires changing the **COTG Timesheet Report** template (see "Personalizing content" on page 923).

> **Tip:** The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.

You could also:

- Add text, images and other elements (see "Content elements" on page 761)
- Change the layout (see "Styling and formatting" on page 879)

- Change the "Media" on page 652. If you have an image of the stationery the PDF will be printed upon, add it to the Media to make designing the template little easier.
- Change the "Master Pages" on page 648
- Add sections (see "Print context" on page 624)

Once the template is ready, send it to Workflow (see "Sending files to Workflow" on page 593) and send the Workflow configuration to the server again.

Workflow configuration

To transform the project into a real solution, the Workflow configuration has to be adapted in two important respects.

- The timing and manner of the production of forms will need to change. The Input task of the **cotg_ts_deploy_form** process must be adapted to the actual input data. You will also need to create an appropriate data mapping configuration for that data, which probably means saving the input as sample data first (see "Saving input as sample data" on the previous page).
- How the submitted data must be handled depends entirely on what they are used for. You may have to write them to a database and use them in an email, for example. The **cotg_ts_generate_report** process demonstrates only a few of the things that you can do with the submitted data.

  Tip: The Submitting Data with Web Forms Wizard shows how to use a data selection function - `xmlget()` - to read data from the request XML and how to store data in the Data Repository (see "Sample Project: Submitting Data with Web Forms" on page 291).

For general information about processes in Workflow see About Processes and Subprocesses, in the Online Help of Workflow.

Sample Project: Print Promotional Jobs

The Print Promotional Jobs Sample Project creates a simple, yet complete OL Connect project that produces **promotional print output**.

The project extracts data from an XML file and uses that data to personalize a promotional letter. The output is a single file containing all the letters, in the format that was selected in the wizard (PDF, PCL or PostScript Level 3).

Installing the project

Select **File > New > Sample Projects > Print Promotional Jobs** from the menu to start the Print Promotional Jobs Sample Project. (See also: "Promotional Print Jobs - Sample Project" on page 1189.)

Alternatively, open the **Welcome** screen, click **New** at the left, then **Project** at the right and select the project type.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

Note: In order to use the project, OL Connect Server and OL Connect Workflow must be installed on the local machine.

The wizard lets you select the **folder** in which you want the solution to be installed. Since Sample Projects are *versioned* (see "Versioned projects" on page 253), the software suggests to store it in the **OL Connect** folder that is located in your **Documents** folder. By default, all versioned project have their own folder under **Documents/Connect**.

In the selected folder, the Sample Project will create two subfolders: Configurations and Workspace.
The project's resource files are saved to the **Configurations** folder.
The **Workspace** folder is used for debugging or running the solution. It has an In folder that may be used to monitor incoming data and an Out folder to write output files to.

The selected folder's path is saved to a global variable in the Workflow configuration (see "Workflow configuration" on page 283).

That variable is used in the settings of the Capture Folder and Send to Folder tasks.

Next, you have to select the desired output type: PDF, PCL or PostScript Level 3. Each of the output types is available in two variants. The 'Stationery' variants will include the virtual stationery (stored in the OL Connect template) in the output. This is a setting in the Output Preset (see "Print settings" on page 282).

Finally, enter the **username** and **password** that will allow the software to access the Connect Server.

Testing and running the project

Once the Sample Project has finished the installation, the project is ready to be tested.

1. Locate the Workflow configuration in the **Configurations\Workflow** folder and open it in OL Connect Workflow.

2. Select the **pr_prom_generate_output** process.
3. Open the **Debug** ribbon and click **Run**.

In Debug mode, the first Input task is skipped and the process is executed using a **sample data file**. This project is pre-configured to use the file: **Sample Data.xml**.

A successful test run results in a subfolder in the **Workspace\Out** folder, named after the current month and year. The Print output is written to that subfolder. It consists of one file that should have the proper file extension.

Running the project

Having tested the project, you will be ready to send it to the Workflow service; see Saving and sending a Workflow Configuration in Workflow's Online Help.
To test the project, which then runs on the server, copy the **Sample Data.xml** file from the Configurations\Data folder to the **Workspace\In** folder. The output should appear in the same folder as before.

Project details

The letter template

The promotional letter is designed in the **PR_PROM Letter** template. It contains one **Print section** (see "Print sections" on page 629).

There are two **Master Pages** that are each applied to different pages in the Print section (see "Applying a Master Page to a page in a Print section" on page 651).

The **Media** contains the virtual stationery. Whether this is included in or omitted from the final output depends on a setting in the Output Preset.

Personalization is mostly done via simple **text scripts**. Such scripts look for a text surrounded by @ (e.g. @city@) and replace that by the correct data. (See: "Variable data in text: scripts and placeholders" on page 945.)

A few scripts are a little bit more sophisticated.

- The **sales_rep** script combines some data fields and adds a line break. Double-click the script to open the Text Script Wizard with which this script was made. (See: "Using the Text Script Wizard" on page 948.)
- The **Promo** script shows or hides a paragraph (with the ID 'promo') depending on whether the customer's address is in Montréal. (See: "Showing content conditionally" on page 958.)

- The **year** script changes the year in the conditional paragraph to the current year. This script only has to look for @year@ in an element that has the ID 'promo', instead of in the entire letter, which makes it run faster.
- The **Dynamic Signature** script switches the signature, with a file name based on a data field. (See: "Dynamic images" on page 972.)
- The sender's address is adjusted depending on where the customer lives. The two different sender's addresses are saved in **snippets**. (See: "Loading a snippet via a script" on page 1091.)

Styling is done via **CSS**. The context_all_styles.css file contains style rules for certain HTML elements (level 1 and 2 headings) and for elements with a certain class or ID.

The data mapping

The data with which the template is merged come from an XML file: **Sample Data.xml**. They are extracted from the XML file with a data mapping configuration: , made with the DataMapper module in the Designer.
The data mapping configuration is very straightforward. It extracts all data without modifications and it does no pre- or post-processing. Simple data mapping configurations like this are made with the XML Data Mapping Wizard (see: "Using the wizard for XML files" on page 353). The wizard creates a configuration with only one step: the **Extract All** step.

Of course, this will only work with the appropriate data files. This data mapping configuration was designed for XML files that are structured like this file: **Sample Data.xml**. The sample file is located in the **Configurations\Data** folder, but you will also see it when you open the data mapping configuration itself: select **File > Open** from the menu; browse to the **Configurations\Resources** folder and select the data mapping configuration: **PR_PROM Data XML**.

Print settings

The Print context and Print sections can have their own print settings, such as Duplex printing or binding style (see "Print settings in the Print context and sections" on page 626). But in this template, they don't have any. The way the letter is outputted is determined (mostly) by an **Output Preset**.

The Sample Project installs six Output Presets; two for each type of output. Per output type, all settings in the Output Presets are the same, except one: Print Virtual Stationery. The ones that have this option enabled will include the Media - the Virtual Stationery - in the output. The other presets will not print the Media.
Only the selected Output Preset is put to use in the Workflow configuration.

To see the exact settings, open an Output Preset in the Designer: first select **File > Open** from the menu; then browse to the preset.
Note that the **Output Type**, on the Print Options page in the Output Creation dialog, is set to **Prompt for file name**. This setting is overruled in the Workflow configuration (see below).

Workflow configuration

Whenever new input data appears in the **Workspace\In** folder, the letter template is automatically merged with it and then printed. That is, if the Workflow server is running with the Workflow configuration installed by the Sample Project.

This project's Workflow configuration: **Print Promotional Jobs**, contains just one process: a basic OL Connect Workflow Print process (see "Print processes with OL Connect tasks" on page 306).

It consists of three plugins:

- The **Folder Capture** Input task. This task retrieves all XML files that enter the Workspace\In folder.
- The **All In One** plugin. This plugin uses the data mapping configuration and template that come with the project, and the Output Preset selected in the Sample Project. The plugin is set to handle the output 'Through Workflow'. This means that the task will return the output file to the Workflow process.
- The **Send to Folder** Output task. This task writes the output file to a subfoler in the Workspace\Out folder. It makes use of system variables (see Standard variables) to dynamically create a subfolder based on the current month and year (%M_%y). The name of the file consists of the original file name (%O), the current time (%h%n%s) and the file extension. The file extension (.pdf, .pcl or .ps) matches the output technology selected in the Sample Project.

Both the Folder Capture and the Send to Folder task read the project's Workspace path from a global variable. The value of that variable is set by the Sample Project when it installs the project.

Customizing the project

A project is a great starting point for building an OL Connect solution. This part explains how to make changes to the project.

Do you intend to expand the project into a solution where Workflow runs on a different machine that also has a different regional setting? Then indicate the desired encoding in the Designer preferences (see "Sample Project deployment settings" on page 1058) **before** installing the project.

Input data

Here's how to adjust the project to input data that has a different structure or file type or comes from a different source.

1. Create a new data mapping configuration to match your input data. (See .)
2. When it's finished, send the new data mapping configuration to Workflow (see "Sending files to Workflow" on page 593).
3. Open the Workflow configuration: Print Promotional Data.
4. Double-click the **Folder Capture** Input task and change the file mask, or replace the task by the appropriate Input task. See: Input tasks in Workflow's Online Help.
5. Double-click the **All In One** task and select the new data mapping configuration on the **Data Mapper** tab.

Note: If the input data is JSON, you don't need a data mapping configuration: JSON data can be used in a template as is. See: "Adding JSON sample data" on page 937. However, if you want the data to be saved in the Connect database, let the XML/JSON Conversion plugin convert the JSON to XML and create an XML data mapping configuration to extract the data.

Template

There are countless ways to customize the template to meet your exact requirements. You could, for example:

- Add text, images and other elements (see "Content elements" on page 761)
- Change the layout (see "Styling and formatting" on page 879)
- Change the "Media" on page 652
- Change the "Master Pages" on page 648
- Add sections (see "Print context" on page 624)

In order to further **personalize** the letter, you need to open your data mapping configuration. (See: "Personalizing content" on page 923.)

Tip: The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.

When the template is ready, send it to Workflow (see "Sending files to Workflow" on page 593).

Finally, in Workflow, adjust the process: double-click the **All In One** task to open it, and select the new template on the **Content Creation** tab. This is only necessary when the file name has changed.
Send the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help).

Print output

To save the output to another kind of file, you could use one of the other Output Presets. To do that, adjust the process in Workflow: double-click the **All In One** task to open it, and select the Output Preset of your choice on the **Output Creation** tab.

To change the settings in an Output Preset, open it in the Designer: select **File > Open** from the menu and browse to the preset.

All of the presets provided by the wizard save the output to one file. You might want to save the output to a number of files, for instance, one per customer. The Transactional Print Jobs project shows you how to do that.

Sample Project: Print Transactional Jobs

The Print Transactional Jobs Sample Project creates an OL Connect project that produces **transactional print output**. It prints invoices to:

- A single PDF for the entire job (in which the invoices are grouped per customer).
- One PDF per customer.
- One PDF per invoice.

Installing the project

Select **File > New > Sample Projects > Print Transactional Jobs** from the menu to start the wizard. See also: "Transactional Print Jobs - Sample Project" on page 1190.)

Alternatively, open the **Welcome** screen, click **New** at the left, then **Project** at the right and select the project type.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

Note: In order to use the project, OL Connect Server and OL Connect Workflow must be installed on the local machine.

The wizard lets you select the **folder** in which you want the solution to be installed. Since Sample Projects are *versioned* (see "Versioned projects" on page 253), the software suggests to store it in the **OL Connect** folder that is located in your **Documents** folder. By default, all versioned project have their own folder under **Documents/Connect**.

In the selected folder, the Sample Project will create two subfolders: Configurations and Workspace.
The project's resource files are saved to the **Configurations** folder.
The **Workspace** folder is used for debugging or running the solution. It has an In folder that may be used to monitor incoming data and an Out folder to write output files to.

The selected folder's path is saved to a global variable in the Workflow configuration (see "Workflow configuration" on page 289).

That variable is used in the settings of the Capture Folder task.
The path is also copied to the Output Presets which are used in the Create Output tasks.

Finally, enter the **username** and **password** that will allow the software to access the Connect Server.

Testing and running the project

Once the Sample Project has finished the installation, the project is ready to be tested.

1. Locate the Workflow configuration in the **Configurations\Workflow** folder and open it in OL Connect Workflow.
2. Select the **pr_tran_generate_output** process.
3. Open the **Debug** ribbon and click **Run**.

The debugger *always* skips the first Input task. It needs a **sample data file** to work with. Normally you'd have to select a sample data file in Workflow. However, the project is pre-configured to use this file: **Sample Data.xml** (located in the project's Configurations\Data folder). The dates in the file are based on the date on which the project is installed.

A successful test run results in the following output in the **Workspace\Out** folder:

- One PDF containing all invoices, grouped by customer number.
- A subfolder containing PDFs that have one invoice each.
- A subfolder containing one PDF per customer. (Some customers have more than one invoice.)

Running the project

Having tested the project, you will be ready to send it to the Workflow service; see Saving and sending a Workflow Configuration in Workflow's Online Help.
To test the project when it runs on the server, copy the **Sample Data.xml** file from the Configurations\Data folder to the **Workspace\In** folder. The same output should appear in the same folders as before.

Project details

The invoice template

The invoice is designed in the **PR_TRAN Invoice** template. It contains one **Print section** (see "Print sections" on page 629) and one **Master Page** (see "Master Pages" on page 648).

The **Media** (virtual stationery) is included in the output - it is printed on the page's background - , according to a setting in the Output Presets.

Styling is done via **style sheets** (see "Local formatting versus style sheets" on page 879). The style rules are in the context_print_styles.css file. Note how they combine the HTML tag, ID and class of elements to select them. (See also: "Selectors in OL Connect" on page 1083.)

Scripts

Scripts personalize content. Most of the scipts in the **Information** folder (on the Scripts pane) are made with the **Text Script Wizard** (see "Using the Text Script Wizard" on page 948).

> **Tip:** Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

Text that should be replaced by data is not only surrounded by @ (e.g. @date@) but also wrapped in a **span** (see "Span" on page 826). Each span has an ID, or a class if the same text occurs more than once, to use that as the selector of the script. The selector of the Invoice Total script, for instance, is a class: .inv-total.

Wrapping elements in a box (see "Boxes" on page 823) or in a semantic HTML element makes it easier to target them in a script or in a style sheet. Place the cursor in the element or select multiple elements. Then, on the menu, click **Insert > Wrap in Box**. You can now use the wrapper element as a script's or style's selector; see "Using the Text Script Wizard" on page 948 and "Styling and formatting" on page 879.

The Text Script Wizard has an Expand button that opens the **Script Editor**, where you can edit the code or add your own. (For an introduction, see "Writing your own scripts" on page 1064).
When you double-click on the **Invoice Meta** script, the Script Editor opens immediately. This

script shows how to add data using the replace() method (see String replace).
Note that this is a native JavaScript function. Functions that are only available in Designer scripts are listed in the "Designer Script API" on page 1479.

Dynamic Table

The table in the invoice is a Dynamic Table. It is filled and expanded dynamically by the scripts in the **Products** folder. To learn how to insert and edit such a table, see "Dynamic Table" on page 974.

Note that this table does not use one of the default table styles, and that the style sheet with the default table styles is not present in the template. To add that style sheet to the template, insert a table using the Dynamic Table wizard.

The data mapping configuration

The template is merged with data from an XML file: Sample Data.xml. They are extracted from the XML file with a data mapping configuration, made with the DataMapper (see "The DataMapper" on page 336).
The data mapping configuration first extracts the common invoice fields, and then the transactional data, in a loop. For information about how to extract transactional data from an XML file, see: "From an XML file" on page 399.

Of course, this will only work with the appropriate data files. This data mapping configuration was designed for XML files that are structured like this the sample file: **Sample Data.xml**.
It is located in the **Configurations\Data** folder, but you will also see it when you open the data mapping configuration itself: select **File > Open** from the menu; browse to the **Configurations\Resources** folder and select the data mapping configuration: **PR_TRAN Data XML**.

Print settings

A Print context and Print sections can have their own print settings (see "Print settings in the Print context and sections" on page 626). The only print setting that the Print section has in this template, is the Duplex setting. (Right-click the section and select **Sheet Configuration**. See also: "Sheet Configuration dialog" on page 1208.)

All other print settings are in the three **Output Presets**. These are used by the Create Output tasks in the Workflow process.
To see the exact settings, open an Output Preset in the Designer: first select **File > Open** from the menu; then select the preset.

The three Output Presets all create PDF output, add the template's name to the PDF's meta data, and save the file in a certain directory. They all have the Print Virtual Stationery option enabled. So, what are the differences between these presets?

Groups and separation

The **PR_TRAN PDF Full Job** preset outputs only one file.
The **PR_TRAN PDF per Invoice** preset outputs one file per invoice, separating the output by document.
The **PR_TRAN PDF per Customer** preset outputs one file per customer, separating the output by document set.

The last output preset needs the invoices in the job to be **grouped** in document sets first (by customer number, in this case). That is what the **Job Preset** does. This preset is used by the Create Job task in the Workflow configuration.

Variable file names

The Job Preset does something else, too: it attaches extra information - meta data - to the documents and to the document sets. Both Output Presets that produce multiple files use that information in the output file names.
They do that by using a variable in the file output mask field. The variable refers to certain meta data attached to items at a certain level (the document or document set, respectively). For more information see "Print output variables" on page 1659.

Workflow configuration

Whenever new input data appears in the **Workspace\In** folder, the invoices are automatically merged with the data and printed to one file, one file per customer, and one file per invoice. That is, if the Workflow server is running with the Workflow configuration installed by the Sample Project. (See "Running the project" on page 287.)

This project's Workflow configuration contains just one process. It is a typical OL Connect Print process (see "Print processes with OL Connect tasks" on page 306), with the difference that it does not have one Create Output task, but three. Each of them uses its own Output Preset (see "Print settings" on the previous page).

Note that the **Branch** tasks back up the job file and job information, so that the main branch continues with a job file that is unaffected by what happens to the job file inside the branch.

The **Folder Capture** task reads the project's Workspace path from a global variable. The value of that variable is set by the Sample Project when it installs the project.

Customizing the project

A project is a great starting point for building an OL Connect solution. This part explains how to make changes to the project.

Do you intend to expand the project into a solution where Workflow runs on a different machine that also has a different regional setting? Then indicate the desired encoding in the Designer preferences (see "Sample Project deployment settings" on page 1058) **before** installing the project.

Input data

Here's how to adjust the project to input data that has a different structure or file type or comes from a different source.

1. Create a new data mapping configuration to match your input data. (See .)
2. When it's finished, send the new data mapping configuration to Workflow (see "Sending files to Workflow" on page 593).
3. Open the Workflow configuration: Print Promotional Data.
4. Double-click the **Folder Capture** Input task and change the file mask, or replace the task by the appropriate Input task. See: Input tasks in Workflow's Online Help.
5. Double-click the **All In One** task and select the new data mapping configuration on the **Data Mapper** tab.

Note: If the input data is JSON, you don't need a data mapping configuration: JSON data can be used in a template as is. See: "Adding JSON sample data" on page 937. However, if you want the data to be saved in the Connect database, let the XML/JSON Conversion plugin convert the JSON to XML and create an XML data mapping configuration to extract the data.

Template

There are countless ways to customize the template to meet your exact requirements. You could, for example:

- Add text, images and other elements (see "Content elements" on page 761)
- Change the layout (see "Styling and formatting" on page 879)
- Change the "Media" on page 652
- Change the "Master Pages" on page 648
- Add sections (see "Print context" on page 624)

In order to further **personalize** the invoice, you need to open your data mapping configuration. (See: "Personalizing content" on page 923.)

> **Tip:** The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.

Note that a "Dynamic Table" on page 974 can't exist without a detail table in the data.

When the template is ready, send it to Workflow (see "Sending files to Workflow" on page 593).

Finally, in Workflow, adjust the process: double-click the **Create Print Content** task to open it, and select the new template. This is only necessary when the file name has changed. Send the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help).

Print output

To save the output to another **type** of file, or to send it to a printer, you must change the Output Presets. Output Presets are edited in the Designer: select **File > Open** from the menu and select the preset. Click Next and adjust the **Printer** and **Output** options.

To **separate** the output differently, for example, by city in which the customers live, you need to change the Output Preset as well as the Job Preset.

1. Open the Job Preset in the Designer: select **File > Open** from the menu and select the preset.
2. Change the **Grouping Options** by selecting the relevant fields on the **Job**, **Job Segment**, and/or **Document Set** tab.
3. Adjust the **Separation Options** in the Output Preset accordingly.

> **Tip:** The Execute Data Mapping task, Create Job and Create Output task may be replaced with the All In One task, with its Output Creation set to None. Using the All In One instead of the separate tasks makes the process a little faster.

Sample Project: Submitting Data with Web Forms

The Submitting Data with Web Forms Sample Project creates an OL Connect project that responds to a request by serving a **web page** with a web **form**. When the form is submitted, it handles the submitted data, saving them to the Data Repository.

Installing the project

From the menu, select **File > New > Sample Projects > Submitting Data with Web Forms** to start the wizard. See also: "Submitting Data with Web Forms - Sample Project" on page 1191.)

Alternatively, open the **Welcome** screen, click **New** at the left, then **Project** at the right and select the project type.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

Note: In order to use the project, OL Connect Server and OL Connect Workflow must be installed on the local machine.

The wizard lets you select the **folder** in which you want the solution to be installed. Since Sample Projects are *versioned* (see "Versioned projects" on page 253), the software suggests to store it in the **OL Connect** folder that is located in your **Documents** folder. By default, all versioned project have their own folder under **Documents/Connect**.

In the selected folder, the Sample Project will create two subfolders: Configurations and Workspace.
The project's resource files are saved to the **Configurations** folder.
The **Workspace** folder is used for debugging or running the solution. It has an In folder that may be used to monitor incoming data and an Out folder to write output files to.

The selected folder's path is saved to a global variable in the Workflow configuration (see "The Workflow configuration" on page 294).

Finally, enter the **username** and **password** that will allow the software to access the Connect Server.

Testing and running the project

Once the Sample Project has finished the installation, the project is ready to be tested.

1. If the templates and data mapping configurations haven't been sent to Workflow yet, do that first (see "Sending files to Workflow" on page 593). This requires that the Workflow service is running (see Starting the Workflow service in Workflow's Online Help).
2. Send the Workflow configuration to the Workflow service; see Saving and sending a Workflow Configuration in Workflow's Online Help.
3. Access the web page by entering the following URL in a browser on the machine that runs Workflow: `http://localhost:9090/form`. (The process starts with a NodeJS

Server Input task, and Workflow's NodeJS Server listens on port 9090 by default.)

4. Fill in some data and submit the form. You will now be presented with a Thank You web page.

Saving input as sample data

Saving input as sample data

Testing a process in Debug mode is only possible with a sample data file. The process is pre-configured to use the Sample Data.xml file located in the Configurations\Data folder. To create your own sample data file:

1. Locate the Workflow configuration in the **Configurations\Workflow** folder and open it in Connect Workflow.
2. Select the process.
3. Enable the **Send to Folder** step (step 2 in the process).
4. Send the Workflow configuration to the Workflow service (see Saving and sending a Workflow Configuration) and run it again, with a custom name value.
   The Send to Folder step will now write the input data - the job file - to a file in the **Workspace\Debug** folder.

When you select the file as sample file (on the Debug ribbon), it can be used to debug the process. Note however that the process will not return the web page to a browser, as it wasn't started by a browser.

> **Tip:** The saved file can be used to create a data mapping configuration.

Project details

The web templates

Both web pages are designed in the **WEB_FORM Web Page** template. It contains a **Web context** with two Web pages: **form** and **thank_you** (see "Web pages" on page 688).

Styling is done via **style sheets** (see "Local formatting versus style sheets" on page 879). The style rules are in the context_web_styles.css file. Note that they use the HTML tag (e.g. section), ID (#theID) and/or the class (.theclass) of elements to select them. (See also: "Selectors in OL Connect" on page 1083.)

Dynamic content is inserted by **scripts**, which are listed on the Scripts pane. You can double-click a script to open it.
The scripts in the **Thank you** folder only affect the **thank_you** Web page; on the **form** Web page, nothing matches their selectors.

Tip: Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

- Most of the **scripts** in the **Thank you** folder directly insert data that were submitted via the Web form; see "Variable data in text: scripts and placeholders" on page 945.
- The **Snippets** script (also in the Thank you folder) loads two snippets, depending on the values of two data fields (see "Loading a snippet via a script" on page 1091). Note that it selects the table, in order to insert the snippet *after* the table (via `results.after()`; see "after()" on page 1560).
- The **Year** script puts the current year in the footer of both Web pages.

For more information about writing scripts, see: "Writing your own scripts" on page 1064.

The Workflow configuration

The project's Workflow configuration contains just one process: an OL Connect Web process (see "Web processes with OL Connect tasks" on page 309).
Its NodeJS Server Input task has two **HTTP Action** entries: form and thank_you. This means the process will be triggered when the server receives a request that ends with either /form or /thank_you.

The request is saved to an XML file, which becomes the job file.

The **Text Condition** task checks if the request's header is thank_you. (For the function that is used to read the header from the job file, see: XML data selections.)
If it is, the tasks in the branch run. The submitted data are merged with the thank_you Web section and saved in Workflow's **Data Repository**.
Otherwise the main branch renders and returns the web form.
Note how the **Section** setting in the Create Web Content tasks determines which web page is outputted (double-click the task to open the properties).

The **Delete** task is an Output task that does nothing, actually; it doesn't even remove anything. However, this step is useful when running the project step by step in **Debug** mode. When it is followed by another task, the Create Web Content task returns its output to the Workflow process, where it can be viewed (click **View as HTML**).

The data mapping configuration

To extract the submitted data from the job file (the request XML), the process uses the data mapping configuration: **WEB_FORM Data**.
To open it, select **File > Open** from the menu and browse to the **Configurations\Resources** folder.

In order to create the data mapping configuration, the input data needed to be saved to a file (see "Saving input as sample data" on page 293).
The sample data file is located in the **Configurations\Data** folder, but you will also see it when you open the data mapping configuration itself.

For more information about data mapping configurations see "The DataMapper" on page 336.

Customizing the project

A project is a great starting point for building an OL Connect solution. This part explains how to make changes to the project.

Do you intend to expand the project into a solution where Workflow runs on a different machine that also has a different regional setting? Then indicate the desired encoding in the Designer preferences (see "Sample Project deployment settings" on page 1058) **before** installing the project.

The form

If you want to add inputs to the form and use the submitted data in the Thank You web page, here's how to do that.

1. Open the **form** Web section and add inputs to the Web Form (see "Using Form elements" on page 698).
2. Save the template and send it to Workflow (see "Sending files to Workflow" on page 593).
3. Upload the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help) and let the process save the input data to a file (see "Saving input as sample data" on page 293).
4. Use the saved file to add the new data to the data mapping configuration (see "Opening a data mapping configuration" on page 342). Send the data mapping configuration to Workflow.
5. Open the **thank_you** Web section and use the new data fields to personalize the page (see: "Personalizing content" on page 923). Then send the template to Workflow again.

   Tip: The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.
6. Send the Workflow configuration to the server.

Web pages

Apart from the form, there are lots of ways to modify the template. You could add text, images and other elements (see "Content elements" on page 761) and change the layout of the web pages (see "Styling and formatting" on page 879).

Once the template is ready, send it to Workflow (see "Sending files to Workflow" on page 593).

Finally, in Workflow, adjust the process: double-click the **Create Web Content** task to open it, and select the new template. This is only necessary when the file name has changed. Send the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help).

> **Note:** If you want the web pages to be **responsive** (have a different layout on screens of different sizes), you could add Media Queries to the style sheet (see Media Queries on MDN Web Docs), or start a new template with a wizard; see "Creating a Web template with a Wizard" on page 683.

Workflow configuration

Serving the two web pages could also be achieved using **separate processes**, but in fact it is more efficient to have a single process, as activity needs to be monitored for each process.

In real life the **submitted data** will probably not be stored in the Data Repository, but used differently. This means that the Push to Repository task will need to be replaced by the appropriate tasks, but that won't change the way the submitted data is *retrieved*.
To see how the current process does that, double-click the Push to Repository task and take a look at the task's properties. The xmlget() function, used in the Value fields, is a data selection function (see Data Selections). Data selection functions are available in many plugins, in any field that accepts variables. Try right-clicking a field and select **Get Data Location** from the contextual menu. (See also: Variable Task Properties.)

For general information about processes in Workflow see About Processes and Subprocesses, in the Online Help of Workflow.

Sample Project: Serving a Web Page

The Serving a Web Page Sample Project creates an OL Connect project that responds to a request by serving a **web page**. This project extracts data from a parameter in the given URL and shows the value on the web page.

Installing the project

To start the Sample Project, select **File > New > Sample Projects > Serving a Web Page** from the menu. (See also: "Serving a Web Page - Sample Project" on page 1191.)

Alternatively, open the **Welcome** screen, click **New** at the left, then **Project** at the right and select the project type.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

Note: In order to use the project, OL Connect Server and OL Connect Workflow must be installed on the local machine.

The wizard lets you select the **folder** in which you want the solution to be installed. Since Sample Projects are *versioned* (see "Versioned projects" on page 253), the software suggests to store it in the **OL Connect** folder that is located in your **Documents** folder. By default, all versioned project have their own folder under **Documents/Connect**.

In the selected folder, the Sample Project will create two subfolders: Configurations and Workspace.
The project's resource files are saved to the **Configurations** folder.
The **Workspace** folder is used for debugging or running the solution. It has an In folder that may be used to monitor incoming data and an Out folder to write output files to.

The selected folder's path is saved to a global variable in the Workflow configuration (see "The Workflow configuration" on page 299).

Finally, enter the **username** and **password** that will allow the software to access the Connect Server.

Testing and running the project

Once the Sample Project has finished the installation, the project is ready to be tested.

If the **templates** and **data mapping configurations** haven't been sent to Workflow yet, do that first (see "Sending files to Workflow" on page 593).

This requires that the **Workflow service** is running (see Starting the Workflow service in Workflow's Online Help).

To test the project:

1. Send the **Workflow configuration** to the OL Connect Workflow service; see Saving and sending a Workflow Configuration in Workflow's Online Help.

2. Access the web page by entering the following URL in a browser on the machine that runs Workflow: `http://localhost:9090/hello`.
3. Follow the instructions on the page to see how values in the URL change the text of the page.

Saving input as sample data

Saving input as sample data

Testing a process in Debug mode is only possible with a sample data file. The process is pre-configured to use the Sample Data.xml file located in the Configurations\Data folder. To create your own sample data file:

1. Locate the Workflow configuration in the **Configurations\Workflow** folder and open it in Connect Workflow.
2. Select the process.
3. Enable the **Send to Folder** step (step 2 in the process).
4. Send the Workflow configuration to the Workflow service (see Saving and sending a Workflow Configuration) and run it again, with a custom name value.
   The Send to Folder step will now write the input data - the job file - to a file in the **Workspace\Debug** folder.

When you select the file as sample file (on the Debug ribbon), it can be used to debug the process. Note however that the process will not return the web page to a browser, as it wasn't started by a browser.

> **Tip:** The saved file can be used to create a data mapping configuration.

Project details

The web template

The web page is designed in the **WEB_HELLO Web Page** template. It contains one **Web section** (see "Web pages" on page 688).

Styling is done via **style sheets** (see "Local formatting versus style sheets" on page 879). The style rules are in the context_web_styles.css file. Note how they combine the HTML tag, ID and class of elements to select them. (See also: "Selectors in OL Connect" on page 1083.)

There are only two **scripts** in the template, as you can see on the Scripts pane. You can double-click to open them.

Tip: Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

- The **My name is** script looks for an element that has the ID: `hero`. Inside that element it looks for the text: `@name@` and replaces that with either the default name ("John Doe") or the name given in the URL.
- The **Year** script puts the current year in the footer.

For more information about writing scripts, see: "Writing your own scripts" on page 1064.

Tip: You don't have to write a script yourself if you just want to insert some data directly into the template. You could simply drag-and-drop the data on the template or use a wizard (see: "Variable data in the text" on page 924).

The Workflow configuration

The project's Workflow configuration, **Serving a Web Page.OL-workflow**, contains just one process. It is a simple OL Connect Web process (see "Web processes with OL Connect tasks" on page 309).

The NodeJS Server Input task's **HTTP action** is set to `hello`. This means that this process will be invoked whenever someone tries to access the Workflow server with this URL: `http://{server's address}/hello`.

The **Delete** task is an Output task that does nothing, actually; it doesn't even remove anything. However, this step is useful when running the project step by step in **Debug** mode. If it wouldn't be followed by another task, the Create Web Content task would not return its output to the Workflow process, where it can be viewed (click **View as HTML**).

The data mapping configuration

When the browser sends the request to Workflow, the Workflow process starts and puts the request into XML, including the values that were given in the URL. To extract those values from the 'job file' (the XML) the process uses a data mapping configuration, made with the DataMapper: **WEB_HELLO Data**.

To open the data mapping configuration, select **File > Open** from the menu and browse to the **Configurations\Resources** folder.

To create the data mapping configuration, the input data needed to be saved to a file first (see "Saving input as sample data" on the previous page).
The sample data file is located in the **Configurations\Data** folder, but you will also see it when you open the data mapping configuration itself:

For more information about data mapping configurations see "The DataMapper" on page 336.

Customizing the project

A project is a great starting point for building an OL Connect solution. This part explains how to make changes to the project.

Do you intend to expand the project into a solution where Workflow runs on a different machine that also has a different regional setting? Then indicate the desired encoding in the Designer preferences (see "Sample Project deployment settings" on page 1058) **before** installing the project.

Input data

If you want to add other parameters to the URL and use those data in the template, here's how to do that:

1. Run the project, add more parameters to the URL and let the process save the input data to a file (see "Saving input as sample data" on page 298).
2. Use the file to create a new data mapping configuration that extracts the additional data. (See "Creating a new data mapping configuration" on page 338.)
3. Send the new data mapping configuration to Workflow (see "Sending files to Workflow" on page 593).
4. Open the Workflow configuration, double-click the **Execute Data Mapping** task to open it and select the new data mapping configuration.
5. Use the data in the template (see: "Personalizing content" on page 923), send the new template to Workflow and select it in the properties of the **Create Web Content** task.
6. Send the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help).

Alternatively you could use input data from a Web form. This is demonstrated by another Sample Project: .

Web page

There are countless ways to modify the template. You could add text, images and other elements (see "Content elements" on page 761) and change the layout (see "Styling and formatting" on page 879).

In order to further **personalize** the web page, open your data mapping configuration (or JSON data, if the input data will be in that format; see "Adding JSON sample data" on

page 937) and use the data fields to personalize the page. (See: "Personalizing content" on page 923.)

> **Tip:** The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.

Once the template is ready, send it to Workflow (see "Sending files to Workflow" on page 593).

Finally, in Workflow, adjust the process: double-click the **Create Web Content** task to open it, and select the new template. This is only necessary when the file name has changed. Send the Workflow configuration to the server (see Saving and sending a Workflow Configuration in Workflow's Online Help).

> **Note:** If you want the web page to be **responsive** (have a different layout on screens of different sizes), it is recommended to start a new template with a wizard; see "Creating a Web template with a Wizard" on page 683.

Workflow configuration

The current Workflow configuration is very simple. In a real-life solution, a process that generates a web page could be part of a larger solution, in which, for example, emails are generated with a link to a personal landing page.

For general information about processes in Workflow see About Processes and Subprocesses, in the Online Help of Workflow.

Workflow processes in OL Connect projects

About Workflow processes

A Workflow configuration consists of one or more **processes**. Each of these processes waits until it is time to run or until it receives the necessary input, and then does its job. A process is made up of **tasks** (also referred to as **plugins**). In the Workflow configuration tool they are divided into groups. For example:

- **Input** tasks look for certain input data on a certain channel and start a process when they find it. The data that enter the process are handed from one plugin to the next in the form of a file: the 'job file'.

- **Process logic** tasks help you to control the process, for example by branching it, creating a loop, or referring to another process or a subprocess.
- **Action** tasks perform a wide variety of operations.
- Metadata - data about a job, kept in the process - can be accessed and manipulated using **Metadata** tasks.
- **OL Connect** tasks enable the process to communicate with the Connect Server and/or database. (See: "OL Connect tasks" on the next page.)
- A process ends with an **Output** task.

Note: Workflow was originally developed - and is still used - as part of PlanetPress Suite. Nevertheless, most plugins are just as useful in Connect as in PlanetPress Suite. Where plugins are restricted to one software package or the other, it is indicated in Workflow's Online Help.

Common OL Connect Workflow processes

In an OL Connect project there are typically a number of Workflow processes that communicate with the Connect Server and/or database through one or more of the **OL Connect** tasks.

These are the most common Workflow processes in Connect solutions.

- **Print process**. A process that results in print output implements either the typical OL Connect Print plugins, or the All in One plugin which combines the Print plugins; see "Print processes with OL Connect tasks" on page 306.
- **Email process**. An email process outputs one or more emails; see "Email processes with OL Connect tasks" on page 305.
- **Web process**. A web process serves one or more web pages. The output can vary from a plain and simple web page to an extensive web form. See: "Web processes with OL Connect tasks" on page 309.
- **COTG processes**. The process that creates a form for the Capture OnTheGo app is in essence a web process, but a COTG solution consists of more than one process. The typical COTG processes are described here: "Capture OnTheGo Workflow processes" on page 310.

Tip: Sample Project

The OL Connect software comes with a number of Sample Projects that generate a Workflow configuration, and any Connect templates, data mapping configurations, and Print Presets required to make the project work. For more information, see "Sample Projects" on page 1185 in the online help.

In each process, **runtime parameters** can pass variables from OL Connect Workflow to the template, data mapping configuration or Job Preset. For more information see "Runtime parameters" on page 1687.

OL Connect tasks

In Workflow there is one set of plugins developed especially for OL Connect: the **OL Connect** tasks. These plugins are designed to communicate to the OL Connect Server and database, and work with components that are only available within OL Connect: templates, data mapping configurations, and Print Presets.

This topic describes the available OL Connect tasks, which are commonly used in Workflow processes in OL Connect projects (see "Workflow processes in OL Connect projects" on page 301).

Data extraction

The Execute Data Mapping task is likely to appear in a lot of OL Connect Workflow processes. It generates a record set in the OL Connect database by executing a data mapping configuration on a data source.

Output creation

Merging the records with a template is the job of one of the following tasks.

Email

The Create Email Content task creates email content items and sends them to the recipient. See "Email processes with OL Connect tasks" on page 305.

Web

The Create Web Content task can handle only one record at a time and returns a web page. See "Web processes with OL Connect tasks" on page 309.

Print

The Create Print Content task generates a set of Print content items in the OL Connect database, but it is not an Output task. Following this task, the Create Job task assembles

Print content items into a Print job. The Create Output task turns the job into a print output file.

The All In One task combines the four tasks involved in a Print process. If a process's only goal is to create Print output, this task is often the better choice because it is faster and more efficient. (See "Print processes with OL Connect tasks" on page 306.)
However, sometimes using the separate tasks is inevitable.
For example when you want to merge jobs with the Merge Jobs task, or when Print content items created at different times should be put into the same job, or when the same extracted data should be used to create different kinds of output.

The Create Preview PDF task generates a PDF preview for a single record as fast as possible. This preview is typically used for PDF previews embedded in web pages.

PlanetPress Suite Documents

The Create PDF/VT task creates PDF/VT files from a PlanetPress Suite Document (created with PlanetPress Design). This PDF/VT is compatible with the **Create Print Content** task directly, without the use of an OL Connect template (PDF/VT mode).

OL Connect database

The following tasks let you act directly upon the OL Connect database:

- The Set Properties task adds properties as tags to items/sets in the OL Connect database.
- The Retrieve Items task retrieves items (records, or content items, etc.) or sets of items from the OL Connect database, by ID or by property.

  Note: Combined, the Set Properties and Retrieve Items tasks make it possible to *batch* and *commingle* Print content items.

The following two tasks use the Metadata (data about the current job, maintained in the Workflow process; see Metadata) to do something with records in the Connect database:

- The Mark Connect Sets for Deletion task marks the sets found in the current Metadata as 'to be deleted' in the OL Connect database, so that they will be deleted the next time the database's Cleanup process runs.
- The Update Data Records task updates records in the OL Connect database using values from the current Metadata.

File Store

The OL Connect Server has a File Store which it uses for transient files. This File Store is managed by the Cleanup service who takes care of removing obsolete files when those files are not marked as permanent.
You can make use of the File Store in your own Workflow processes. The Workflow configuration tool has three tasks to that end: the File Store - Upload File task, File Store - Download File task, and File Store - Delete File task.

Email processes with OL Connect tasks

Sending out emails, based on a Connect template, via an automated process requires you to design a Connect email process in the Workflow configuration tool.
This topic explains which tasks and files are used in a Connect email process.

> **Tip:** An easy way to create a Connect email project, including the Workflow configuration and the files that it needs, is to use a **Sample Project**. See "Sample Project: Basic Email" on page 265.

The structure of an OL Connect email process

In an OL Connect Email process one of two plugins is essential: the **Create Email Content** plugin or the **Render Email Content** plugin.

Both plugins can create a set of emails using the Email context in a Connect template.

The difference between the two is that the Create Email Content plugin sends the emails to the email server immediately, whereas the Render Email Content plugin creates the emails and stores them, so that they can be sent later, via SMTP or via an Email Service Provider such as SendGrid or Mailjet.

You have to select the Connect template in the task's properties.

If the template doesn't need any data, you can set the Data Source of the plugin to JSON and enter an empty JSON string: {}.
However, if the template should be merged with data, you will need to add one or more tasks to provide the required data. The plugin must receive either Metadata containing information regarding a valid Record Set, or JSON data. This can be the output of tasks like:

- An **Execute Data Mapping** task which retrieves data from the job file (such as the request XML). In addition to creating records in the Connect database, this task can output the (IDs of the) records in the Metadata.

- A **Retrieve Items** task which retrieves an existing record set from the Connect database and outputs it in the Metadata or as JSON data.
- A **Create File** task that creates a JSON file.
- Etc.

Which task or tasks fit best, depends on where the data come from.

The Create Email Content and Render Email Content plugins can be found on the OL Connect tab of the Plug-In Bar in Workflow.
For a description of all OL Connect tasks, see "OL Connect tasks" on page 303.

Files used in a Connect email process

Before creating the Email process in the Workflow configuration tool, you will need to create:

- A template with an Email context. (See "Creating a template" on page 589)
- If the email should contain variable data, you might need to make a data mapping configuration (see "Creating a new data mapping configuration" on page 338). If the input is going to be JSON data, you could add them to the design using a JSON file (see "Adding JSON sample data" on page 937).

Templates and data mapping configurations must be sent from the Designer to OL Connect Workflow before they can be used in the Workflow process; see "Sending files to Workflow" on page 593.

The next step is to create a process in OL Connect Workflow that generates Email output, using these files.
For more information about how to create a process in Workflow, please refer to the Online Help of Workflow.

Print processes with OL Connect tasks

Generating print output from a template via an automated process requires you to design a print process in the OL Connect Workflow configuration tool.
This topic explains which tasks and files are used in a print process.

Tip: An easy way to setup a print project in OL Connect, including the print process and the files that it needs, is to use a **Sample Project**. There are two Sample Projects that create a sample print project. See "Sample Projects" on page 1185.

There is also a **Walkthrough sample** that helps you build a Print process for Connect documents in the Workflow Configuration tool by yourself, step-by-step: Creating a Print process in Workflow.

The structure of a print process

In its simplest form, such a process may consist of only two plugins: an **Input** task and the **All In One** plugin.

The **All In One** task combines the following four **OL Connect** tasks:

- The **Execute Data Mapping** task, or the **Retrieve Items** task.
  The Execute Data Mapping task extracts data from the job data file and puts them in a record set. This task works according to the instructions in a data mapping configuration, made with the DataMapper (see "Data mapping configurations" on page 337).
  The Retrieve Items task can be used to retrieve records from the OL Connect database.
- The **Create Print Content** task. This task merges data with a template, resulting in Print Content Items. Templates are made with the Designer (see "Templates" on page 589).

  Note: Content creation can be skipped completely if there is no need to merge the extracted data with a template, and if the input data is paginated (e.g. a PDF). Both the All in One task and the Execute Data Mapping task have an option to bypass content creation.
- The **Create Job** task. This task turns a set of Print Content Items into a print job. Any settings are passed on to Workflow via a Job Preset.
- The **Create Output** task. This task creates the actual output file(s). Any settings are passed on to Workflow via an Output Preset.

The **All In One** task is the fastest and most efficient way to create print output. The task exchanges less data with the server than the separate plugins do and it has multi-threading support: it can produce the data set and content items in parallel.

Nevertheless, the separate plugins would be used in the following situations:

- The record set, created by the Execute Data Mapping task, is also needed to create another kind of output in the same process.

- The input is JSON data, which can be used directly. In this case there is no need to use the Execute Data Mapping task or Retrieve Items task.
- The necessary Print Content Items have already been created, whether in the same or in another Workflow process. Print Content Items can be retrieved from the OL Connect database using the **Retrieve Items** task. Subsequently, the Create Job and Create Output tasks can generate print output from them.
  Note that at the time the Print Content Items are created, their ID or the Print Content Set ID needs to be stored somewhere where the print process can access them, in order to retrieve the Print Content Items.

The tasks mentioned here can all be found on the OL Connect tab of the Plug-In Bar in Workflow. For a description of each task, see "OL Connect tasks" on page 303.

Files used in a print process

Before creating the print process in the Workflow configuration tool, you will need to create:

- A **template** with a Print context. (See "Creating a template" on page 589)

In addition, the process may use:

- A **data mapping configuration**, or a data model at the very least, if the documents should contain variable data. (See "Creating a new data mapping configuration" on page 338.)
- A **Job Preset** (see "Job Presets" on page 1646). A Job Preset defines where the output goes and makes it possible to filter and sort records, group documents, and add metadata.
- An **Output Preset** (see "Output Presets" on page 1648). An Output Preset can split a print job into smaller print jobs, and set printing options such as binding, OMR markings and the like.

All of these files are made with the Designer and need to be sent to OL Connect Workflow before they can be used in the Workflow process (see "Sending files to Workflow" on page 593.

When the necessary files have been created and sent to Workflow, the next step is to create a process in OL Connect Workflow that generates Print output, using these files (see "Print processes with OL Connect tasks" on page 306).

Tip: For more information about how to create a process in Workflow, please refer to the Online Help of Workflow.

Web processes with OL Connect tasks

Serving a web page, made with a template, via an automated process requires you to design a web process in the OL Connect Workflow configuration tool.
This topic explains which tasks and files are used in a web process.

Tip: An easy way to start an OL Connect web project including the web process and the files that it needs, is to use a **Sample Project**. There are two Sample Projects that generate a sample web project. See "Sample Projects" on page 1185.

Note: With a trial or reseller license, Connect Web output is limited to the *localhost*. This means that the Connect Server and Workflow must be on the same workstation in order to create Web output.

The structure of a web process

In a web process two plugins are essential:

- An **Input** task.
- The **Create Web Content** plugin.

If the **Input** task is a Server Input task, that task will return the output of the process - the web page - to the caller.
The **HTTP Action** of the Server Input task determines how the process is triggered. If, for example, the HTTP Action is /hello, the process will be invoked when the Workflow server receives a request for a resource called hello. (You could, for example, trigger the process by typing the following URL in the address bar of your browser: http://localhost:9090/hello, if the browser runs on the same machine as the Workflow server and the Input task is a NodeJS Server Input task. The protocol could be HTTPS, depending on the settings of the Input task.)
To trigger the process from a Web Form, set the **Action** of the Form to the **HTTP Action** of the Input task. (See "Forms" on page 842.)

The **Create Web Content** task creates an HTML output file from the Web context in a template.
Select the Web template in the task's properties.
If the Web template doesn't need any data, you can set the Data Source of this task to JSON and enter an empty JSON string: {}, or set it to Record ID and enter 0 (zero).
However, if the template should be merged with data, the task needs either a valid Record ID or a JSON object. In that case, one or more steps must be added to the process to provide the required data. For example:

- An **Execute Data Mapping** task that retrieves data from the job file (such as the request XML). This task outputs one or more records to the Metadata. By default, the Create Web Content task uses the first Record ID from the Metadata.
- A **Retrieve Items** task that retrieves an existing record from the OL Connect database and outputs it in the Metadata or as JSON object.

Which task or tasks fit best, depends on where the data must come from.

Of course, numerous other tasks could be added to the process. If you'd want to save the output of the Create Web Content task - the web page - to a file, for example, the task would have to be followed by a Send to Folder task.

The Create Web Content task can be found on the OL Connect tab of the Plug-In Bar in Workflow.
For a description of all mentioned OL Connect tasks, see "OL Connect tasks" on page 303.

Files used in a web process

Before creating the web process in the Workflow configuration tool, you will need to create:

- A template with a Web context. (See "Creating a template" on page 589.)
- If the web page should contain variable data, you might need to make a data mapping configuration (see "Creating a new data mapping configuration" on page 338). If the input is going to be JSON data, you could add them to the design using a JSON file (see "Adding JSON sample data" on page 937).

Templates and data mapping configurations need to be sent from the Designer to OL Connect Workflow before they can be used in the Workflow process; see "Sending files to Workflow" on page 593.

The next step is to create a process in OL Connect Workflow that generates Web output, using these files.
For more information about how to create a process in Workflow, please refer to the Online Help of Workflow.

Capture OnTheGo Workflow processes

A Capture OnTheGo solution requires no less than three basic processes in a Workflow configuration:

- One process that generates a document (most often, a form), stores it and notifies the COTG app of the document's existence. Note that this process doesn't send the actual

document. It sends a *ticket* to the Capture OnTheGo Server so that when the intended user logs in, the name of the document will appear in the app's Repository, signaling to the user that the document can be downloaded.
The key task in this process is the Output to Capture OnTheGo plugin.

- One process that authenticates and replies to document requests from COTG users. This process returns the actual document directly to the app. It starts with a Server Input task (the NodeJS Server Input task or the older HTTP Server Input task), which returns the final output of the process to the caller.
- One process that receives and handles the data when a COTG user submits a form. This process starts with a Server Input task as well.

The three basic processes are described in more detail in the Capture OnTheGo manual: Basic processes involved in COTG.

Tip: An easy way to create a COTG solution, including the Workflow configuration and other files that it needs, is to use the **COTG Timesheets Sample Project**. See "Sample Project: COTG Timesheets" on page 272.

Batching and commingling

A Connect Print process in its simplest form merges data with a template and creates the print job(s) in one go, as shown in "Print processes with OL Connect tasks" on page 306.

In reality, however, a company might want to create letters and invoices during the week, for example, and then print them out only over the weekend. For such a scenario, it would be better to separate the data mapping and content creation process from the production of the actual print job(s), and work with print **batches**.

Batching means gathering and then printing previously created print content items. **Commingling** refers to documents originating from different templates being combined into one print batch.

This topic explains how to batch, commingle, and - not the least importantly - *sort* and *separate* items into print batches with Connect.

The Workflow tasks mentioned here can all be found on the OL Connect tab of the Plug-In Bar in Workflow. For descriptions of the tasks, see "OL Connect tasks" on page 303.

Batching

Batching refers to creating print jobs from print content items that were created earlier. A batching print process is split up in 2 phases:

- Phase 1: Content is created by merging a print template with data. Newly created **print content items** are always automatically saved in the Connect database[1]. Information about the **set** of items (the "content set"), including the Content Set ID, gets stored in the database as well.
- Phase 2: Print content items are retrieved from the Connect database using the Retrieve Items task. The Create Job and Create Output tasks transform these items or sets into one or more print jobs.

In order to retrieve print content from the Connect database in the second phase, some preparatory work must be done in the first phase. What needs to be done exactly depends on what you want to retrieve, and how.
Sorting and separating the output also requires some preparation (see "Sorting and grouping items in print batches" on page 314).

[1]Database entries get cleaned up automatically as well. Make sure that the **Cleanup Service** is configured appropriately, so that it doesn't remove relevant record sets and content sets before batching starts (see "Clean-up Service preferences" on page 1035).

What to retrieve: content sets or content items

The Retrieve Items task can retrieve only one type of entity from the Connect database at a time: records, record sets, content items and so forth. Since the Create Job task can only work with print *content items* or *content sets*, in a batching print process the choice is narrowed down to these two possibilities.

Here are a few things to consider:

- If you want the **Create Job** task to use a **Job Preset**, you must retrieve content sets. A Job Preset can filter, sort, and group print content items, add meta data and make finishing settings (see "Print Presets" on page 1644).
- From a performance perspective, working with content sets is better. Retrieving content sets is faster than retrieving content items and thus can improve performance. Also, working with Job presets is more efficient than working with individual content items in Workflow.
- If you decide to retrieve content sets, you may still have to do preparatory work on the content item level to enable a Job Preset to sort the output (see "Sorting and grouping items in print batches" on page 314).

How to retrieve: selection methods

The Retrieve Items task can retrieve items/sets by **ID** or based upon **conditions**.

Retrieving items/sets by ID

New print content items are automatically saved in the Connect database when the **Create Content** task creates them. The task returns the IDs of those items, as well as the ID of the content set, to the Workflow process via **Metadata** (see About Metadata).
In order to retrieve (sets of) items by ID later on, you will have to get the IDs from the Metadata and store them somewhere else. Workflow's Data Repository would be a good place to store them in; you can use the Push to Repository task to store the IDs there.

Retrieving items or sets using conditions

When the **Retrieve Items** task retrieves items or sets based upon conditions, the respective database entities are matched against the **values** of data fields or by the values of **properties** set on entities in the Connect database.

- **Values**: Print content items and sets don't contain data fields, but they *do* have a link to the data record with which they were created, so selecting and sorting them by value is still a possibility.
- **Properties** are key/value pairs that can be set on entities in the Connect database. There are two ways to do that:
    - Using the Set Properties task.
      Ideally, the **Set Properties** task directly follows the Create Print Content task in a Workflow process. The Create Print Content task returns the IDs of the content items as well as the ID of the content set to the process via the Metadata. Using those IDs, the Set Properties task can either set properties on all new **Content Items** or on the **Content Set** that was just created.
      Use two consecutive Set Properties tasks to set properties on both levels.
    - Through a Post Pagination script in the template. (See "Post Pagination Scripts" on page 1114 and "contentitem" on page 1620.)

In order to retrieve (sets of) items by data **values**, you may have to adjust the data mapping configuration (see "Data mapping configurations" on page 337).

Note that a **property** can only be used to retrieve entities on the level on which the property was explicitly set. For example, when you set a property on a *content set*, you can only use that property to retrieve the content set in its entirety (i.e. all content items that belong to that content set); it isn't possible to retrieve individual content items using that property. Properties added to a content set are not propagated to the individual items in that content set.
To label both a set and the items in it, use the Set Properties task twice.

Note: Data field values and properties of items in the Connect database are also used when sorting the output. Make sure to set properties that are necessary for sorting on the content item level.

Commingling

When documents in one print batch originate from different templates, that is called **commingling**.

Commingling isn't very different from batching in how it's done. The content creation phase will likely be a little different, as there will probably be a separate process or branch for each of the templates used to create the print content.

However, **sorting and grouping** can become more important when print content items are commingled. For instance, when invoices and letters are commingled, they probably need to be grouped per customer, and sorted in a certain way. Ensure that there is a unique ID in the different data sets that can be used to group items together that belong to the same mail piece.

Sorting and grouping items in print batches

Print content items can be grouped and sorted by the **values** of data fields or by the values of **properties** set on them in the Connect database.

- **Values**: Print content items and sets don't contain data fields, but they *do* have a link to the data record with which they were created, so selecting and sorting them by value is still a possibility.
- **Properties** are key/value pairs that can be set on entities in the Connect database. There are two ways to do that:
    - Using the Set Properties task.
      Ideally, the **Set Properties** task directly follows the Create Print Content task in a Workflow process. The Create Print Content task returns the IDs of the content items as well as the ID of the content set to the process via the Metadata. Using those IDs, the Set Properties task can either set properties on all new **Content Items** or on the **Content Set** that was just created.
      Use two consecutive Set Properties tasks to set properties on both levels.
    - Through a Post Pagination script in the template. (See "Post Pagination Scripts" on page 1114 and "contentitem" on page 1620.)

Sorting by **property** is only possible if the property has been set explicitly on the respective **content items** (not content sets) in the Connect database.

It is also important to note that in order to use a data field or property in a sorting rule, it should be present in **all** content items.
You may have to set properties on content items (not content sets) or add data fields to a data model (see "Data mapping configurations" on page 337), for no other reason than to sort the items by them eventually.

In Connect there are two places where you could group and sort print content after it has been retrieved from the Connect database:

- In a Job Preset.
- On the Batching/Commingling tab of the Retrieve Items task.

Job Preset

Based on the settings in a Job Preset, the Connect Server can filter, sort, and group print content items, add meta data and make finishing settings (see "Job Presets" on page 1646). Compared with the Batching/Commingling tab of the Retrieve Items task, a Job Preset offers a *lot* more options. It is also usually more efficient; that's why it's the recommended method, for larger jobs at least.

A Job Preset can only take **content sets** as input. So, if you want to use a Job Preset, the **Retrieve Items** task must retrieve content sets from the Connect database, not content items.
This also means that if you want to use properties in conditions to retrieve the correct items, those properties must be set on the content set level.
However, any properties that you want to be used for filtering, grouping or sorting must be set on the content items.

Batching/Commingling tab of Retrieve Items task

The Batching/Commingling tab of the **Retrieve Items** task allows you to group and sort print content on two levels. You can:

- Bundle content items into "documents" (mail pieces) and sort the items within each document.
- Put documents in "groups", and define how documents are sorted within a group.

Note that this tab is only available when the Retrieve Items task is configured to retrieve **content items** - not content sets.

Separating the output

Print output in batches can be separated just like all other print output, using an Output Preset in the **Create Output** task (see "Output Presets" on page 1648).

This requires that the output be grouped beforehand, either via a Job Preset or via the Batching/Commingling tab of the Retrieve Items task.

Note that the terminology used on the Batching/Commingling tab and in an Output Preset is slightly different.

- Select **Document Set** in the Separation Settings to separate the "documents" defined on the Batching/Commingling tab.
- Select **Job Segment** in the Separation Settings to separate the "groups" defined on the Batching/Commingling tab.

OL Connect Automate

Node-RED is a programming tool for wiring together hardware devices, APIs and online services. It was originally developed by IBM as an open source project and evolved into a general purpose IoT/IIoT (Internet of Things/Industrial Internet of Things) programming tool. It provides a browser-based editor that makes it easy to wire together **flows** using the wide range of **nodes** in the **palette**.

With the OL Connect Automate stack, Upland OL has added its own set of nodes to this palette: **OL Connect nodes** and a couple of supporting nodes. The OL Connect nodes allow Node-RED to communicate with the OL Connect server and access the OL Connect Database and File Store. The supporting nodes perform common tasks like triggering a flow when a new file arrives or list the files in a specific folder.

For a complete overview of all OL Connect Automate packages, see OL Connect Automate packages.

Automate vs. Workflow

The OL Connect Automate stack for Node-RED was born out of the need to provide users of OL Connect with a modern, web-based, cloud-friendly, unicode-aware automation tool.
For years, this job has been handled by OL Connect Workflow (aka Workflow), whose first version dates back to the turn of the century.
However, as powerful and efficent as Workflow is, its evolution is somewhat hampered by its original design: a native-code, 32-bit, on-premise application that only partially supports unicode. Consequently, we decided to explore other automation methods that could, at first complement, and ultimately replace Workflow. The packages that are part of the OL Connect Automate stack are the first step in this direction.

> **Note:** The OL Connect Automate stack is a **technology preview**. The stack is not (yet) a replacement for OL Connect Workflow; in its current inception, it should be considered a complement to it.

The information in this chapter will help you get started with OL Connect Automate. It focuses on the OL Connect nodes, the settings that need to be made to communicate with the OL Connect server, and the most common flows in OL Connect solutions with Node-RED. Tips and techniques regarding standard nodes in such solutions can be found in another topic: "Node-RED: nodes and common techniques" on page 320.

For general information about Node-RED, please refer to Node-RED's website: nodered.org.

Installing and updating nodes

For newcomers to OL Connect Automate, utilize the Windows installer available on OL Resource Center (https://olresourcecenter.uplandsoftware.com/downloads/?olc-product=olconnect). This installer streamlines the process by automatically installing necessary components and the latest collection of OL Connect Automate nodes.

1. Follow the instructions on Downloading and installing Node.js and npm | npm Docs (npmjs.com).
2. Start the Node-RED editor.
3. Download and install the OL Connect nodes and other Automate nodes that you wish to use.

   You can do this using the Palette Manager: in the right-hand sidebar, click **Manage Palette** and look for **@objectif-lune**.

   > **Tip:** You can also use the Palette Manager to update nodes.

   Or, run a command in your Node-RED user directory - typically `~/.node-red`. For example, to install the OL Connect package:

   ```
   npm install @objectif-lune/node-red-contrib-connect
   ```

   For a complete overview of all OL Connect Automate packages, see OL Connect Automate packages.

   See also: https://flows.nodered.org/search?term=objectif-lune.

> **Note:** Obviously, you must have OL Connect installed on a server that can be reached from the one where Node-RED is installed, otherwise you won't be able to perform any automation tasks.

> **Tip: Example flows** are provided by OL Connect Automate, once installed. The examples appear in the *flows* folder in the Import dialog in Node-RED which allows you to import an example flow into a new or existing flow; see Importing and Exporting Flows in the Node-RED documentation.
> Each example comes with a **Comment** node (Read me) with instructions.

Connection settings for OL Connect Server

In order to setup an OL Connect Server connection:

1. Add one of the OL Connect nodes to a flow.
2. Double-click the node to open its Properties window and click the **Edit Server** icon (a pencil).
3. Enter the URL, user name and password to use to connect to the OL Connect Server.
4. Click **Done**.

The next time you add an OL Connect node to this flow, the server will appear in the list of servers in that node's Properties window.

OL Connect resources in Node-RED

OL Connect resources are files created with the Designer or DataMapper. In order to use them in the OL Connect nodes in a flow in Node-RED they need to be deployed to the OL Connect Server first.

Deployed resources are registered in the OL Connect database and saved in the File Store. OL Connect nodes can either refer to them by name or by using the internal database ID (also referred to as the *managed file ID*). The simplest approach is to use the resource name in the respective field of the node.

There are two ways to deploy OL Connect resources to the File Store: using the **Designer** or from within a **flow** in Node-RED's editor.

- For instructions on sending files to the File Store from the Designer, see "Sending files to Connect Server or to another server" on page 594.
- OL Connect's **file upload** node sends files to the File Store from within a flow. When used in a Startup flow the uploaded files will be available to all flows in the project. See: "OL Connect startup flow" on page 325.

Flows in an OL Connect application

These are some of the typical flows in a Node-RED OL Connect solution.

- **Startup flow**

  A Startup flow initializes variables in Node-RED's *context* - through which information is shared between nodes without using the messages that pass through a flow - and deploys Connect resources for the project. See "OL Connect startup flow" on page 325 for a detailed description.

- **Print flow**

  A print flow creates and outputs paginated content. See "An OL Connect print flow in Node-RED" on page 328.

- **Email flow**

  An email flow creates email content and stores it in the File Store or sends it immediately using an email service provider. See "An OL Connect email flow in Node-RED" on page 327.

- **Web (form) flow**

  A web flow creates web content and stores it in the File Store or serves it immediately. See "An OL Connect web flow in Node-RED" on page 333.

- **PDF preview flows**

  A PDF preview flow provides a PDF preview, usually at the request of a browser. See "An OL Connect preview PDF flow in Node-RED" on page 331.

Node-RED: nodes and common techniques

The nodes and common techniques described in this chapter will help you to start creating flows for an OL Connect application in Node-RED.

The typical flows that an OL Connect application may have are described separately; see "Flows in an OL Connect application" on the previous page.

For the user documentation of Node-RED, please refer to Node-RED's website: nodered.org.

This chapter uses a startup flow for the examples. See also: "OL Connect startup flow" on page 325

Nodes used in OL Connect flows

In addition to the OL Connect nodes, these standard nodes will often be used in OL Connect applications:

- The **inject** node triggers the flow. It can be setup in a way that it triggers the flow whenever it is deployed or when Node-RED is started (see "OL Connect startup flow" on page 325). It can also be used to inject the full path to a (sample) data file into a flow.

  > **Tip:** Adding multiple **inject** nodes to your flow is a great way to test the flow with data files of different sizes.

- The **HTTP in** node creates an HTTP end-point for creating web services.
- Any flow that starts with an HTTP in node must have a path to an **HTTP response** node otherwise requests will eventually timeout.

- The **read file** node loads a file. If the file is a JSON file, the result is a JSON string.
- The **write file** node writes the content of `msg.payload` to a file.
- The **JSON** node parses a JSON string and adds the keys with their values as properties to `msg.payload`.
- The **change** node can initialize variables and change their value.
- The **function** node executes JavaScript code. See https://nodered.org/docs/user-guide/writing-functions.
- The **debug** node displays the content of the `msg` object, which is passed between nodes in a flow, in the debug message console.

  > **Tip:** Add a **debug** node - one for each property of the `msg` object - after another node to verify that the properties are changed as expected.

- The **split** node turns a single `msg.payload` consisting of an array into a series of messages, each of which has a payload containing one of the array's elements. Note that the Split node only works on `msg.payload` and not on its child objects.

  > **Tip:** The **join** node can be used to continue a flow after iterations made with the **split** node are done.

- The **switch** node can have multiple output channels and these channels can be linked back to a node on the primary/main branch downstream. The **switch** node allows messages to be routed to different branches of a flow by evaluating a set of rules for each message.

Reading a JSON file

In order to load a JSON file you can use a **read file** node. Set the **Filename** property to the full path.
Note that when Node-RED's project feature is used, the path is relative to the project location under Node-RED's user directory (`%userprofile%/.node-red/`).

Example

When triggered, a startup flow must read the manifest.json file.

1. Add a **read file** node after the **inject** node that triggers the flow, and connect their ports.
2. Double-click the **read file** node to view its properties.

3. Paste or enter the full path to manifest.json in the **Filename** property. For example: `C:\projects\project-a\resources\manifest.json`, or, when using Node-RED's project feature: `node-red\projects\project-a\manifest.json`.
4. You can change the **Name** of the node, for example into: Read manifest.
5. Click **Done**.

Parsing a JSON string

When a JSON file is read, the result is a JSON string. In order to access its data, that string needs to be parsed to its JavaScript Object representation. Node-RED provides the **JSON** node to do this conversion.
The parsed data is written to the `payload` property of the `msg` object, which is passed on to the next node. Keys in the JSON become properties of the payload.

Example

After reading the manifest.json file using a **read file** node, a startup flow must parse the JSON string in order to add the keys and values to `msg.payload`. The manifest.json file has the following content:

```
{
        "email": "Laura from OL Acme <laura@ol-acme.com>",
        "someApi": "http://localhost/myendpoint",
        "workspace": "C:\\workspace",
        "resources": [
                "olsg-invoice.OL-template",
                "olsg-invoice-XML.OL-datamapper"
  ]
}
```

1. Add a **JSON** node after the **read file** node. Make sure the **JSON** node is connected to the output port of the **read file** node.
2. Add a **debug** node and connect the **JSON** node to the input port of that **debug** node so that the result can be viewed in the debug message console, once the flow is deployed.

After the JSON file is parsed, the `msg` object will have the following properties: `msg.payload.email`, `msg.payload.someApi`, `msg.payload.workspace` and `msg.payload.resources`.

Using variables

There are various ways to set and get the values of (global and other) variables, for example:

- Via the **change** node. A 'Set' rule can set the value of one variable to a hard-coded value or to the value of another variable (e.g. 'Set' `global.email` to `msg.email` object).
- Via the **function** node, using JavaScript. The node has access to the `msg` object as well as to the different contexts: `context` (i.e. the node's local context), , `flow` and `global`.

The **Context** sidebar in the editor of Node-RED displays the contents of the *context* data store and allows to view the stored variables. The view consists of three sections, one for each *context*: *node*, *flow* and *global*.

Variables stored in the *global context* of Node-RED are visible to all nodes, in all flows, on all tabs. Through these variables, information is shared between nodes without using the messages that pass through a flow.

Example

After parsing a JSON string, a startup flow must initialize global variables to the values of properties of `msg.payload`.

1. Insert a **change** node after the **JSON** node (wire the input port and the output port).
2. Double-click the **change** node to view its properties.
3. Create a rule for each of the values. For example:

   'Set': `global.email`

   To: `msg.payload.email`
4. Click **Done**.
5. Add a **debug** node and connect the **change** node to the input port of that **debug** node so that the result can be viewed in the debug message console.

Setting and moving `msg` properties

There are various ways to set and move values of properties in the `msg` object.

- Via the **change** node. Select 'Set' to set a value; select 'Move' in order to move a value from one property to another property.

- Via the **function** node. The value of a property can be set or replaced using JavaScript.

Example

After reading and parsing a manifest.json file, a startup flow needs to move the names of some OL Connect resources from `msg.payload.resources` to `msg.payload`

1. Add a **change** node to the flow.
2. Create a rule to 'Move' `msg.payload.resources` to `msg.payload`.

Iterating over items in an array

If items in an array need to be passed on individually, the flow will need to iterate over the items. This kind of loop is handled by Node-RED's **split** node. The **split** node turns a single `msg.payload`, consisting of an array, into a series of messages, each of which has a payload containing one of the array's elements.

Note that the **split** node only works on `msg.payload` and not on its child objects.

Example

A startup flow needs to iterate over the array in `msg.payload` that contains the names of OL Connect resources.

1. Insert a **split** node into the flow.
2. Add a **debug** node to verify that Node-RED creates a new message for each resource name.

Concatenating strings

To concatenate two strings in different variables, one could use a **function** node and script it, or use a **change** node.

Example

A startup flow needs to upload an OL Connect resource to the OL Connect server, but the resource name in `msg.payload` lacks the path. The path is stored in `msg.resourceFolder`. To construct the full path and pass it via `msg.fileName`, the flow can use a **change** node.

1. Add a **change** node and a **file upload** node.
2. Double-click the **change** node and create a rule to 'Set' `msg.fileName` to `msg.resourceFolder & payload`. The latter is a **JSONata** expression.

OL Connect startup flow

A startup flow is used in Node-RED projects for OL Connect to **initialize global variables** and to **deploy OL Connect resources** (templates, data mapping configurations, job presets, output presets) for the project to OL Connect's File Store.

A typical startup flow has the following nodes:

- **inject**
- **folder listing**
- **split**
- **file store upload**.

The **file store upload** node is part of the OL Connect package; see OL Connect nodes.

The **folder listing** node can be found in the Files and Folders package. See Files and Folders nodes.

The **inject** and **split** nodes are standard nodes. Other standard nodes that may be useful are the **read file**, **JSON**, **change**, and **debug** node (see "Nodes used in OL Connect flows" on page 320).

> **Tip:** An example startup flow is provided by OL Connect Automate, once installed. The example appears in the *flows* folder in the **Import** dialog in Node-RED which allows you to import an example flow into a new or existing flow; see Importing and Exporting Flows in the Node-RED documentation.
> Instructions are given in the **Comment** node and in the topic: Sample startup flow.

Triggering a startup flow

The **inject** node can trigger a flow in Node-RED. It can be set to do that when the flow is deployed or when Node-RED is started, by selecting the **Inject once after** option in its Properties window.

In order to validate the setup you could add a **debug** node and connect this to the **inject** node (i.e. join the output port of the inject node with the input port of the debug node). Deploying the flow triggers the **inject** node, which by default adds a timestamp to the `payload` of the `msg` object which is passed between nodes in a flow. The timestamp becomes visible on the **debug** tab when you click the Debug icon in the sidebar.
When the flow is deployed for the first time, a suffix appears after the inject node's name, e.g. "1". The suffix indicates the number of times the flow has been triggered.

Initializing global variables

Variables stored in the *global* context of Node-RED are visible to all nodes, in all flows on all tabs. The values could be set in a **change** node or scripted in a **function** node, for example (see "Using variables" on page 323).

Instead of hard-coding the values, they could be read from a JSON file. The flow would need a **read file** node and a **JSON** node to read the JSON file and transform the keys in the JSON into properties of the `msg.payload`. Next, a **change** node can 'Set' global variables to those properties.

> **Tip:** OL Connect's **capture folder** node can read the name of the folder to be monitored for incoming files from a global variable.

Deploying OL Connect resources

> **Tip:** For an example of this type of flow, see Sample startup flow. This sample is provided by OL Connect Automate, once installed.

OL Connect's **folder listing** node returns an array of file paths in `msg.payload`.

To iterate over this array and upload the resources one by one, place a **split** node before the **file store upload** node. (See "Iterating over items in an array" on page 324.)

OL Connect's **file store upload** node uploads a single file to the File Store. This node requires the path to the resource. This should be the full path to the resource or a path relative to either the Node-RED installation or a path relative to the current Node-RED project.

Make sure to check the **Mark as permanent** option. A file marked as permanent will not be removed by OL Connect's Clean-Up Service.

Also make sure to select the OL Connect Server to which the resources must be uploaded in the settings of the node, or enter the connection details (see "Connection settings for OL Connect Server" on page 318).

> **Tip:** Some useful tips:
>
> - Set the mask of the **folder listing** node to *.OL-* to get all OL Connect resources in the selected folder.
> - The **split** node only works on `msg.payload`, not on its child objects. If you are not using the **folder listing** node, you may need to move

an array that contains resource names to the `payload`.
See "Setting and moving msg properties" on page 323.

- To construct the path you may need to concatenate strings. See "Concatenating strings" on page 324.

An OL Connect email flow in Node-RED

This topic explains which nodes and files are used in a typical OL Connect email flow in Node-RED.

Tip: Example email flows are provided by OL Connect Automate, once installed. The examples appear in the *flows* folder in the **Import** dialog in Node-RED which allows you to import an example flow into a new or existing flow; see Importing and Exporting Flows in the Node-RED documentation. Instructions are given in the **Comment** node.

Alternatively, an easy way to get the files that the typical OL Connect email flow needs, is to create a **Sample Project** (see "Sample Project: Basic Email" on page 265). You would use all files except the Workflow configuration file.

The structure of an OL Connect email flow

In its simplest form, an email flow may consist of a node that captures data, the **email content** node to generate email content and a node to send the messages via an Email Service Provider (ESP) like SendGrid or Mailjet, or via Microsoft Outlook Mail.

The **email content** node creates a set of emails, using the Email context in a Connect template.

The data to merge the template with can be the output of (a combination of) nodes like:

- An OL Connect **folder capture** node.
- An OL Connect **data get** node which retrieves an existing record set from the Connect database.
- An OL Connect **data mapping** node which retrieves data from a data file.
- A standard node like the **http in** node, the **inject** node, or a **create file** node that creates a JSON file.
- Etc.

Which node or nodes fit best, depends on where the data come from.

Tip: If the template doesn't need any data, set `msg.payload` of the **email content** node to an empty JSON string: `{}`.

Tip: A number of nodes accept **runtime parameters**. These can be passed via the `parameters` property of the `msg` object which is passed between nodes. For example, a runtime parameter named *brandId* would be passed via `msg.parameters.brandId`. Use a **function** node or **change** node to set the property.

Files used in an OL Connect email flow

Before creating the email flow in the editor of Node-RED you will need to create:

- A template with an Email context. (See "Creating a template" on page 589.)
- If the email should contain variable data, you may need to create a data mapping configuration (see "Creating a new data mapping configuration" on page 338). If the input is going to be JSON data, you could add them to the design without a data mapping configuration, using a JSON file (see "Adding JSON sample data" on page 937).

Note: Note that before templates and data mapping configurations can be used in a flow, they must be sent to the OL Connect server separately. This can be done from the Designer (see "Sending files to Connect Server or to another server" on page 594) or in a startup flow (see "OL Connect startup flow" on page 325).

An OL Connect print flow in Node-RED

This topic gives an overview of the nodes and files that are used in a typical OL Connect print flow in Node-RED.

An easy way to create some example files to work with is to use a Sample Project. There are two Sample Projects that create a sample print project; see "Sample Projects" on page 1185. For a print flow in Node-RED you would use all files *except* the Workflow configuration file.

If the objective of the flow is to serve a preview of a PDF, take a look at this topic: "An OL Connect preview PDF flow in Node-RED" on page 331.

Tip: Example print flows are provided by OL Connect Automate, once installed. The examples appear in the *flows* folder in the **Import** dialog in Node-RED which allows you to import an example flow into a new or existing flow; see Importing and Exporting Flows in the Node-RED documentation. Instructions are given in the **Comment** node.

Alternatively, an easy way to get the files that the typical OL Connect print flow needs, is to use a Sample Project. There are two Sample Projects that create a sample print project; see "Sample Projects" on page 1185. You would use all files except the Workflow configuration file.

The structure of a print flow

Generate paginated print output for XML data using the **all in one** node. This node extracts data from the data file, merges the data with a Connect Designer template, and creates print output.\n\nIn its simplest form, a print flow may consist of a node that captures a data file, such as a **folder capture** or **read file** node, the **all in one** node and a node to save the result to disk.

In its simplest form, a print flow may consist of a node that captures a data file, such as a **folder capture** node or **read file** node, the OL Connect **all in one** node, and a node to save the result to disk.

The **all in one** node combines the following four OL Connect nodes:

- The **data mapping** node that extracts data from a file and stores a record set in the database, or the **data get** node that retrieves previously extracted data from the database.
    - The **data mapping** node needs a data mapping configuration, made with the DataMapper (see "Data mapping configurations" on page 337), and a data file of course.
    - All the **data get** node needs is a Record Set ID.
- The **paginated content** node. This node merges the data with a template, resulting in Print Content Items. Templates are made with the Designer (see "Templates" on page 589).
- The **paginated job** node. This node turns a set of Print Content Items into a print job. Any job settings are passed on via a Job Preset which is made with the Designer (see "Job Presets" on page 1646).
- The **paginated output** node. This node creates the actual output file(s) and stores them in the File Store. Any settings that it needs are passed on via an Output Preset which is made with the Designer (see "Output Presets" on page 1648).

The nodes listed here are OL Connect nodes; see OL Connect nodes.

The OL Connect nodes are built are built in such a way that the output of a node is automatically recognized as input by the next node.

- The **data mapping** node outputs a Data Set ID in `msg.dataSetId` which is used by the **paginated content** node to retrieve the data and create a set of content items.
- The **paginated content** node outputs a Content Set ID in `msg.contentSetId` which is used by the **paginated job** node to create a print job.
- The **paginated job** node outputs a Job Set ID to `msg.jobSetId` which is used by the **paginated output** node to generate output.

To build the flow you also need some nodes that are available by default or have been made by other parties. See "Nodes used in OL Connect flows" on page 320.

The **all in one** node is the fastest and most efficient way to create print output, as in that case multi-threading is supported: the data set and content items are produced in parallel.

Tip: For an example of this type of flow, see Sample print flow with all in one node. This sample is provided by OL Connect Automate, once installed.

The separate nodes would be of use in situations like the following:

- The record set, created by the Execute Data Mapping task, is also needed to create another kind of output in the same flow.
- The input is JSON data which can be used directly and doesn't need to be stored in the database. In this case there is no need to use the **data mapping** node or **data get** node.
- The Print Content Items have already been created, either in the same flow or in another flow. Print Content Items can be retrieved from the OL Connect database using the **data get** node. Subsequently, the **paginated job** and **paginated output** nodes can generate print output from them.
  Note that at the time the Print Content Items are created, their Print Content Set ID needs to be stored somewhere where the print flow can access them, in order to retrieve the Print Content Items. You could for example use MongoDB.
- If there is no need to merge the extracted data with an OL Connect template, the **document mapping** node can be used to create a Content Set as well as a Record Set, so that content creation - normally performed by the **paginated content** node - can be skipped. This requires paginated input data, for example a PDF or PS file, and a data mapping configuration.

Retrieving the output

The **print output** node outputs two things: the ID of the output location in the File Store and an array containing the output file(s). Connect the **print output** node with the OL Connect **file download** node to download a ZIP archive of the entire job based on the

`msg.managedFileId` provided by the **print output** node. The data is read into a data buffer which allows it to then be written to disk using the **write file** node.

The target location can set by entering the full path in the Filename field of the file node. The other option is to set it dynamically in `msg.filename`, using a **change** or **function** node.

Downloading individual output files

The array that the **print output** node outputs can be used to download the files individually. You will need a **split** node to iterate over the array in `msg.payload` and generate a new message for each file, setting `msg.payload` to the respective file name. See "Iterating over items in an array" on page 324.

Make sure to clear the Filename field in the **write file** output node before deploying the flow. A value configured in the node takes precedence over the msg property - `msg.filename` in this case - and will be used for every file, overwriting the previous file.

Files used in a print flow

Before creating a print flow, you will need to create:

- A template with a Print context. (See "Creating a template" on page 589.)

In addition, the flow may use:

- A data mapping configuration, if the documents should contain variable data that is extracted from some data source. (See "Creating a new data mapping configuration" on page 338.)
- A Job Preset. (See "Job Preset" on page 1359.) A Job Preset defines where the output goes and makes it possible to filter and sort records, group documents, and add metadata.
- An Output Preset. (See "Output Presets" on page 1375.) An Output Preset can split a print job into smaller print jobs, and set printing options such as binding, OMR markings and the like.

All of these files are made with the Designer and need to be sent to the OL Connect server *before* using them in a flow; see "OL Connect resources in Node-RED" on page 319.

An OL Connect preview PDF flow in Node-RED

This topic gives an overview of the nodes and files that are used in a OL Connect PDF preview flow in Node-RED.

Tip: An example preview PDF flow is provided by OL Connect Automate, once installed. The example appears in the *flows* folder in the **Import** dialog in Node-RED, which allows you to import an example flow into a new or existing flow; see Importing and Exporting Flows in the Node-RED documentation. Instructions are given in the **Comment** node.

Alternatively, an easy way to get the files that the typical OL Connect preview PDF flow needs, is to use a Sample Project. There are two Sample Projects that create a sample print project; see "Sample Projects" on page 1185. For a PDF preview flow in Node-RED you would use the template and possibly the data mapping configuration; you would not use the Workflow configuration file.

The structure of a preview PDF flow

Serving a preview to a browser

Typically, a PDF preview flow consists of at least three nodes: an **http in** node that receives the request, the **preview pdf** node which is one of the OL Connect nodes, and the **http response** node to serve the preview.

The **http in** and **http response** nodes are standard nodes. See "Node-RED: nodes and common techniques" on page 320.

The **HTTP in** node will pass any query parameters via the payload as JSON data. Query parameters are key/value pairs attached to the end of an URL. The first parameter is appended to the URL using a ? (question mark). If there are multiple parameters, an & (ampersand) is added in between them.
For example: http://localhost:1881/hello**?first=John&last=Doe**.

The **preview pdf** node accepts JSON data provided via the `msg.payload` property. Note that in order to merge the data with a template, the parameter names must match field names in the template's data model.

The template to use can be configured in the **preview pdf** node or passed via the `msg.template` property. (See: "Setting and moving msg properties" on page 323.)

The **preview pdf** node accepts **runtime parameters**. These can be passed via the `parameters` property of the `msg` object which is passed between nodes.
For example, a runtime parameter named *brandId* would be passed via `msg.parameters.brandId`.

Creating and saving multiple PDF files

A PDF preview is usually needed in online solutions, but the preview PDF node can also be used in situations where a PDF file can be produced without an Output Preset.

In such situations the flow could for example have an OL Connect **data get** node (see OL Connect nodes) that outputs an array of Data Record IDs, followed by a standard **split** node, OL Connect's **preview pdf** node and a standard **write file** node.

The **preview pdf** node can only take one data record ID at a time. That's why you will need the **split** node to iterate over the array in `msg.payload` and generate a new message for each file, setting `msg.payload` to the respective Data Record ID. (See "Iterating over items in an array" on page 324.)

Make sure to clear the Filename field in the **write file** output node before deploying the flow. A value configured in the node takes precedence over the msg property - `msg.filename` in this case - and will be used for every file, overwriting the previous file.

Files used in a preview PDF flow

Before creating a preview PDF flow, you will need to create:

- A template with a Print context. (See "Creating a template" on page 589.)

In addition, the flow may use:

- A data mapping configuration, if the document(s) should contain variable data that is extracted from some data source. (See "Creating a new data mapping configuration" on page 338.)

These files are made with the Designer and need to be sent to the OL Connect server *before* using them in a flow. For instructions see "OL Connect resources in Node-RED" on page 319.

An OL Connect web flow in Node-RED

This topic explains which nodes and files are used in a typical OL Connect web flow in Node-RED.

> **Tip:** Example web flows are provided by OL Connect Automate, once installed. The examples appear in the *flows* folder in the **Import** dialog in Node-RED which allows you to import an example flow into a new or existing flow; see Importing and Exporting Flows in the Node-RED documentation. Instructions are given in the **Comment** node.

Alternatively, an easy way to get the files that the typical OL Connect web flow needs, is to create a **Sample Project** (see "Sample Project: Serving a Web Page" on page 296.). You would use all files except the Workflow configuration file.

The structure of an OL Connect web flow

In an OL Connect web flow only one plugin is essential: the **html content** plugin.

The **html content** node creates a set of web pages, using the Web context in a Connect template, and stores them in the File Store or serves it.
If the template doesn't need any data, set `msg.payload` to an empty JSON string: `{}`.

If the template should be merged with data, the data can be the output of another node. Nodes that output data that can be used by the html content node are:

- An **HTTP in** node. It will pass any query parameters via the payload. Query parameters are key/value pairs attached to the end of an URL. The first parameter is appended to the URL using a ? (question mark). If there are multiple parameters, an & (ampersand) is added in between each. Example: http://localhost:1881/hello?first=Peter. Note that in order to use them as data, the parameter names must match field names in the template's data model.
  The flow should end with a **http response** node which serves the HTML output.
- An OL Connect **data mapping** node which retrieves data from the job file (such as the request XML). In addition to creating records in the Connect database, this task can output the (IDs of the) records.
- An OL Connect **data get** node which retrieves an existing record set from the Connect database.
- A standard **create file** node that creates a JSON file.
- Etc.

Which node or nodes fit best, depends on where the data come from.

The html content node accepts **runtime parameters**. These can be passed via the `parameters` property of the `msg` object which is passed between nodes. For example, a runtime parameter named *brandId* would be passed via `msg.parameters.brandId`.

Tip: Use a **change** node or **function** node to set a property. See also: "Node-RED: nodes and common techniques" on page 320.

Files used in an OL Connect web flow

Before creating the web flow in the editor of Node-RED you will need to create:

- A template with a Web context. (See "Creating a template" on page 589.)
- If the HTML file should contain variable data, you may need to create a data mapping configuration (see "Creating a new data mapping configuration" on page 338). If the input is going to be JSON data, you could add them to the design using a JSON file (see "Adding JSON sample data" on page 937).

Note that before templates and data mapping configurations can be used in a flow, they must be sent to the OL Connect server separately. This can be done from the Designer (see "Sending files to Connect Server or to another server" on page 594) or in a startup flow (see "OL Connect startup flow" on page 325).

The DataMapper

The DataMapper is the tool to create a data mapping configuration. Data mapping configurations are used to extract data and transpose that data into a format that can be shared amongst different layouts and outputs created with the OL Connect Designer and Workflow.

The original data, located in a file or database outside of OL Connect, is called a data source.
The first step in the data extraction process is making settings for the input data (see "Data source settings" on page 370), including boundaries for each record inside the data sample. When you define the boundaries, you are actually defining a series of records inside your data sample file.
After configuring these settings you can start working on the logic to extract data from each of those records. You need to identify and extract data from each record. To achieve this, you will create a data mapping workflow, consisting of multiple steps (extractions, loops, conditions and more) (see "Data mapping workflow" on page 368 and "Extracting data" on page 390).

When this process is complete, the result is a Data Model. This model contains the necessary information to add variable data to OL Connect Designer templates. (see "The Data Model" on page 419 for more information). It has a generic format with an emphasis on content, free from any restrictions imposed by the file types or the origin of the data. This allows a same layout or output to be populated with data from different sources and formats without the need to modify it.

DataMapper basics

Connect's DataMapper lets you build a data mapping workflow to extract data from a variety of data sources. The data mapping workflow consists of multiple 'steps' which process and extract data from each record of a data source and store it in a new, extracted record set. The data mapping workflow is saved in a data mapping configuration.

1. **Create a new data mapping configuration**.

   Run the Designer and start creating a data mapping configuration by selecting a data source. See "Data mapping configurations" on the next page.

2. **Configure settings for the data source**.

   The data source can be a file (CSV, PDF, TXT, XML, JSON) or a particular database. Configure how the data source is read by the DataMapper and create a record structure. See "Data source settings" on page 370.

3. **Build the data mapping workflow**.

   A data mapping workflow always starts with the **Preprocessor** step and ends with the **Postprocessor** step. You can add as many steps as you like and edit the Data Model of the extracted data as required. See "Data mapping workflow" on page 368 and "The Data Model" on page 419.

What's next?

Use the data mapping configuration in the Designer module to create templates for personalized customer communications. To learn more, see "The Designer" on page 587.

In Workflow, a data mapping configuration can be used to extract data. The extracted data can then be merged with a Designer template to generate output in the form of print, email or a web page. See Workflow Help and "OL Connect tasks" on page 303.

Data mapping configurations

A data mapping configuration file contains the information necessary for data mapping: the settings to read the source file (Delimiter and Boundary settings), the data mapping workflow with its extraction instructions ('Steps'), the Data Model and any imported data samples.

Data mapping configurations are used in the **Designer** to help add variable data fields and personalization scripts to a template. In fact, only a Data Model would suffice (see "Importing/exporting a Data Model" on page 421). The advantage of a data mapping configuration is that it contains the extracted records to merge with the template, which lets you preview a template with data instead of field names.
It is also possible to generate output of a data mapping configuration directly from the Designer (see "Generating output" on page 1637).

Ultimately data mapping configurations are meant to be used by the OL Connect Server to extract data from a particular type of data file. Typically the extracted data is then merged with a template to generate output in the form of print, email and/or a web page. To make this happen, the data mapping configuration as well as the template and any Print Presets have to be sent to the Server.

If Workflow is used as automation tool, the files need to be sent to Workflow; see "Sending files to Workflow" on page 593. Next, they have to be linked to "OL Connect tasks" on page 303 (i.e. the Execute Data Mapping task or All in One task) in OL Connect Workflow processes.

Alternatively the files can be sent to the Connect Server directly; see "Sending files to Connect Server or to another server" on page 594.

Note: The option to use **AFP input** is included only in the Enterprise edition of OL Connect.

AFP input is dependent on a third party library (CDP), which allows OL Connect to run up to 4 AFP input processes on a given machine, at any given time. This license requires a stable MAC address, and therefore AFP input is not compatible with virtual servers or cloud environments that cannot provide a network interface with a stable MAC address.

Creating a new data mapping configuration

A new data mapping configuration can be made with or without a wizard. When you open a data file with a DataMapper wizard, the wizard automatically detects a number of settings. You can adjust these settings. Next, the wizard automatically extracts as many data fields (or metadata, in case of a PDF/VT or AFP (Enterprise edition only) file) as it can, in one extraction step.
Without a wizard you have to make the settings yourself, and configure the extraction workflow manually.

Note: The DataMapper doesn't use the data source directly, rather it uses a copy of that data: a **data sample**. Although the data sample is a copy, it is updated each time the data mapping configuration is opened or whenever the data sample is selected. More samples can be added via the Settings pane; see "Data samples" on page 469.

From a file

To start creating a data mapping configuration *without* a wizard, first select the data file.

1. Click the **File** menu and select **New**.
2. Click the **Data mapping Configuration** drop-down and select **Files** and then the file type:
    - CSV, XLSX or XLS (Comma Separated Values or Excel),
    - JSON File
    - Microsoft Access
    - PDF, PS, PCL or AFP file
    - Text File
    - XML File

Note: The option to use **AFP** input is included in the Enterprise edition of OL Connect.

The option to use **PCL** input can be purchased with the Professional and Enterprise editions of OL Connect.

3. Click **Next**.
4. Click the **Browse** button and open the file you want to work with.
5. Click **Finish**.

After opening the file, you have to make settings for the input data (see "Data source settings" on page 370) to make sure that the source data is parsed correctly and divided into logical units of data the way you want. Then you can start building the data extraction workflow.

Note: The first line.

- **Excel** files saved in "Strict Open XML" format are not supported yet.
- **PCL** and **PostScript (PS)** files are automatically converted to PDF format by the Connect Server. To allow for this, the default Connect Server and (if it is secured) an authenticated user must be configured via the Preferences (see "Connect Servers preferences" on page 1061).
  Note that when used in a production environment (e.g. a Connect Workflow process) the conversion to PDF may influence the processing speed, depending on the available processing power.
- Some **advanced PCL to PDF** options are available by calling LincPDF (OL Connect's PCL to PDF converter) command line module; see "Advanced PCL to PDF options" on page 356.
- Extracting data from a PDF that comes from a **Windows printer queue** (a PDF converted to PostScript, converted back to PDF by an Input task in Workflow) might not work. The problem lies in the conversion of the original PDF to PostScript. PostScript code is usually optimized to print and not to keep text searchable, by maintaining character identities for example.
  The rule of thumb is: if copy-paste from Acrobat works, so will data mapping; if not, the DataMapper won't either. If the original is a

PDF, it is recommended to find an alternative way to get it in the system instead of going through a print operation.

- **Rotated pages** in a PDF are supported (if rotated 0/90/180/270 degrees). The Extract step will be able to extract data from horizontal and vertical lines of text on rotated pages. Motion steps (such as the Repeat step and the Goto step) however, can only work as expected if text on a page has the same orientation as the page, not when text has been rotated after the page was rotated. The page number and rotation of a page are shown in the status bar at the bottom, next to the region selection information.
- The XML option also allows to select a JSON file. The JSON file will be converted to and treated like XML. If there is no root element in the JSON it will be added and each object without a named parent element is called an 'item'.
  Note that in addition to being valid, the JSON should follow naming rules for XML elements. For example, "adress_line_1:" is a valid key name in JSON, but it cannot be converted to a valid element name in XML because the colon is reserved for namespaces. For XML naming rules and best naming practices, see: XML elements on W3Schools.

With a wizard

Data mapping wizards are available for PDF/VT, AFP, XML, JSON, CSV/XLSX/XLS and database tabular files, because these files are structured in a way that can be used to automatically set record boundaries.

Note: The option to use **AFP input** is included only in the Enterprise edition of OL Connect.

AFP input is dependent on a third party library (CDP), which allows OL Connect to run up to 4 AFP input processes on a given machine, at any given time. This license requires a stable MAC address, and therefore AFP input is not compatible with virtual servers or cloud environments that cannot provide a network interface with a stable MAC address.

The wizard for PDF/VT and AFP files cannot extract data, only metadata. After opening such a file with the wizard, you can build the data extraction workflow for data and metadata in the PDF.

The other wizards use the Extract All step to extract data, but they cannot create detail tables, so they are less suitable for files from which you want to extract transactional data.

There are two ways to open a data file with a wizard: from the Welcome screen or from the File menu.

- From the **Welcome** screen
    1. Open the OL Connect **Welcome** page by clicking the Home icon at the top right or select the **Help** menu and then **Welcome**.
    2. Click **New** at the left.
    3. Click **Data** and select the appropriate file type.
- From the **File** menu
    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select the appropriate file type.

The steps to take with the wizard depend on the file type. See:

- "Using the wizard for CSV and Excel files" on page 344
- "Using the wizard for databases" on page 346
- "Using the wizard for PDF/VT or AFP files" on page 351
- "Using the wizard for XML files" on page 353. This wizard can also be used for JSON files.

Generating a counter

Instead of creating a data mapping configuration for a certain type of data file, you may create a data mapping configuration that only contains a series of sequential numbers.

> **Note:** You can't join this configuration to another data file. It is just a counter to be applied on a static template.
> In a template that is not static, the Handlebars helper: "counter" on page 1010 can be used to generate sequential numbers that update for each record in the main data set.

To generate a counter:

- From the **Welcome** screen
    1. Open the OL Connect **Welcome** page by clicking the Home icon at the top right or select the **Help** menu and then **Welcome**.
    2. Click **New** at the left.
    3. Click **Data** at the right and select **Counters**.
- From the **File** menu
    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select **Generate counters**.

You can set the following parameters:

- **Starting Value**: The starting number for the counter. Defaults to 1.
- **Increment Value**: The value by which to increment the counter for each record. For example, an increment value of 3 and starting value of 1 would give the counter values of 1, 4, 7, 10, [...]
- **Number of records**: The total number of counter records to generate. This is not the end value but rather the total number of actual records to generate.
- **Padding character**: Which character to add if the counter's value is smaller than the width.
- **Width**: The number of digits the counter will have. If the width is larger than the current counter value, the padding character will be used on the left of the counter value, until the width is equal to the set value. For example for a counter value of "15", a width of "4" and padding character of "0", the value will become "0015".
- **Prefix**: String to add before the counter, for example, adding # to get #00001. The prefix length is not counted in the width.
- **Suffix**: String to add after the counter. The suffix length is not counted in the width.

Opening a data mapping configuration

To open an existing data mapping configuration, in the "Menus" on page 465, select **File > Open**. Make sure that the file type is either DataMapper files or Connect files. Browse to the configuration file to open, select it and click **Open**.
Alternatively, click on **File > Open Recent** to select one of the recently opened configuration files.

Saving a data mapping configuration

A data mapping configuration file has the extension .OL-datamapper. The file contains the settings, the extraction workflow ('Steps'), the Data Model and the imported Data Samples (excluding database source files such as mySQL, oracle, etc).

To save a data mapping configuration:

- In the "Menus" on page 465, click on **File > Save**, or click on **Save As** to save a **copy** of a data mapping configuration under a different name.
- In the "Toolbar" on page 527, click the **Save** button.

If the data mapping configuration has not been saved before, you have to browse to the location where the data mapping configuration should be saved and type a name, then click **Save**.

Down-saving a data mapping configuration

The Connect software is backwards compatible: data mapping configurations that were made with an older version of Connect can always be opened with the newest version of the software.
But newer data mapping configurations cannot be opened with an older version of the software.
Connect, however, allows you to save a data mapping configuration in the format of a previous version of the software, so that you may restore a data mapping configuration that was accidentally opened and saved in a newer version, or share a data mapping configuration with users of a previous version of Connect.
Note that it may not always be possible to down-save a data mapping configuration to a an older version. For instance, a JSON-based data mapping configuration cannot be saved to a version earlier than 2021.1 because the JSON data type did not exist prior to that version.

To **down-save** a data mapping configuration, select **File > Save a Copy**, from the menu. Select the software version and click OK. Next you can select a location and give the data mapping configuration a name, as usual.

To **save a copy**, follow the same procedure without selecting a previous version of the software.

Note: Since it was not possible to create a data mapping configuration for a JSON file (without conversion to XML) until version 2021.1, these data mapping configurations cannot be saved in the format of an earlier version.

Using the wizard for CSV and Excel files

The DataMapper wizard for CSV and Excel files helps you create a data mapping configuration for such files. The wizard automatically detects delimiters and extracts all data in one extraction step. It also auto-detects the data type of each field (String, Integer, Boolean, Currency, Float or Date).

The wizard interprets each line in the file as a record. If your data file contains transactional data, you will probably want more lines to go in one record and put the transactional data in detail tables.

The wizard cannot create detail tables. If the file contains transactional data, create the data mapping configuration without a wizard (see "Creating a new data mapping configuration" on page 338), and add the extraction steps yourself.

There are two ways to open a CSV file or Excel file with a wizard: from the Welcome screen or from the File menu.

- From the **Welcome** screen
    1. Open the OL Connect **Welcome** page by clicking the Home icon at the top right, or select the **Help** menu and then **Welcome**.
    2. Click **New** at the left..
    3. Click **Data** at the right and select **CSV or Excel**.
    4. Click the **Browse** button and open the file you want to work with.
    5. Click **Next**.
- From the **File** menu
    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select **From CSV/XLSX/XLS File**.
    3. Click **Next**.
    4. Click the **Browse** button and open the file you want to work with.
    5. Click **Next**.

Note: Excel files saved in "Strict Open XML" format are not supported yet.

After selecting the file, take a look at the **preview** to ensure that the file is the right one and the encoding correctly reads the data. Click **Next**.

For an **Excel** file you can make the following settings:

- **First row contains field names**: Uses the first row of the Excel sheet as headers, which automatically names all extracted fields.
- **Sheet**: Select the sheet from which the data should be extracted.
- **Sort on**: Allows to select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.
- **Automatically detect data types**: Introduced 2024.2. Selecting this option loads the data with automatic detection of Data Types enabled. To use the old functionality (pre 2024.2), do not select the checkbox.

For a **CSV** file the wizard will display the different settings it has detected, allowing you to change them:

- **Encoding**: Defines which encoding is used to read the file.
- **Field Separator** group: Specifies which character separates each field in the file. The choice is between Tab, Comma, Semicolon or Other (to be manually specified) characters.
- **Comment Delimiter**: Defines which character starts a comment line.
- **Text Delimiter**: Defines which character surrounds text fields in the file. Separators and comment delimiters within text are not interpreted as separator or delimiter; they are seen as text.
- **Automatically detect data types**: Introduced 2024.2. Selecting this option loads the data with automatic detection of Data Types enabled. To use the old functionality (pre 2024.2), do not select the checkbox.
- **Ignore unparseable lines**: Ignores any line that does not correspond to the settings above.
- **First row contains field names**: Uses the first line of the CSV as headers, which automatically names all extracted fields.
- **Sort on**: Allows to select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

Tip: The **Sort on** option, combined with the **Stop data mapping** option of the "Action step" on page 417, allows to process only a group of items without having to examine all records. (See also: "Action step properties" on page 499.)

Verify that the data are read properly.

Only the first 10 rows of a data file are analyzed to detect a field's data type. If the data type of values in a column is not consistent throughout the file, an exception may be thrown if a value cannot be interpreted, and data loss may occur if the detected data type is more general than the data type in a subsequent row. For example, if an Integer is detected in the analyzed rows, but a subsequent row contains a Float value, the value of the Float value is truncated to an Integer.
The correct data type can be selected in the field's properties, see "Setting the data type" on page 429.

Finally click **Finish**. All data fields are automatically extracted in one extraction step.

Using the wizard for databases

The DataMapper wizard for database files helps you create a data mapping configuration for a database file. The wizard extracts the data in one extraction step.

The wizard cannot create detail tables. If the file contains transactional data, create the data mapping configuration without a wizard (see "Creating a new data mapping configuration" on page 338), and add the extraction steps yourself.

Opening a database file with a wizard

There are two ways to open an XML file with a wizard: from the Welcome screen or from the File menu.

- From the **Welcome** screen
    1. Open the OL Connect **Welcome** page by clicking the Home icon at the top right or select **Help > Welcome** on the menu.
    2. Click **New** at the left, then click **Data** at the right.
    3. Select **From database**.
    4. Use the drop-down to select the database type.
    5. Click **Next**.
- From the **File** menu
    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select **From databases**.
    3. Click **Next**.

4. Use the drop-down to select the database type.
5. Click **Next**.

Wizard settings for a database file

After opening a database file with a wizard there are a number of settings to make, depending on the database type (see below).
On the last page of the dialog, click **Finish** to close the dialog and open the actual data mapping configuration.

Note: After creating the initial data mapping configuration you may use a custom SQL query via the Input Data Settings; see "Data source settings" on page 370.

Tip: The **Sort on** option, combined with the **Stop data mapping** option of the "Action step" on page 417, allows to process only a group of items without having to examine all records. (See also: "Action step properties" on page 499.)

MariaDB, MySQL, SQL Server or Oracle

- **Server**: Enter the server address for the database.
- **Port**: Enter the port to communicate with the server. The default port is 3306.
- **Database name**: Enter the exact name of the database from where the data should be extracted.
- **User name**: Enter a user name that has access to the server and specified database. The user only requires **Read**access to the database.
- **Password**: Enter the password that matches the user name above.
- **Table name**: The selected database is a set of related tables composed of rows and columns corresponding respectively to source records and fields. Select a table from which you want to extract data.
- **Encoding**: Choose the correct encoding to read the file.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

Microsoft Access

- **Password**: Enter a password if one is required.
- **Table name**: The selected database is a set of related tables composed of rows and columns corresponding respectively to source records and fields. Select a table from which you want to extract data.
- **Encoding**: Choose the correct encoding to read the file.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

ODBC Data Source

Caution: ODBC drivers do not all work with Java applications. Since OL Connect is a Java application, it is strongly recommended to use JDBC instead.

- **ODBC Source**: Use the drop-down to select a 64-bit ODBC System Data Source. OL Connect is a 64-bit application and cannot access 32-bit data sources.
- **This ODBC source is MSSQL**: Check this option if the ODBC source is MSSQL (SQL Server). The options below appear under **MSSQL-ODBC advanced configuration**:
    - **Windows authentication**: Select to use the Windows user name and password that are used by the Connect Service.
    - **SQL Server authentication**: Select to use the User name and Password set below to connect to the SQL Server:
        - **User name**: Enter the SQL Server user name.
        - **Password**: Enter the password for the above user name.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

JDBC

Note: Since JDBC can connect to multiple types of databases, a specific database driver and path to this driver's JAR file must be specified.

- **JDBC Driver**: Use the drop-down to select which JDBC Driver to use for the database connection.
- **JAR file path**: Enter a path to the JAR file that contains the appropriate driver for the database.
- **Server**: Enter the server address for the database server.
- **Database name**: Enter the exact name of the database from where the data should be extracted.
- **User name**: Enter a user name that has access to the server and specified database. The user only requires **Read** access to the database.
- **Password**: Enter the password that matches the user name above.
- **Advanced mode**: Check to enable the **Connection String** field to manually enter the database connection string.
- **Connection string**: Type or copy in your connection string.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

Note: To instruct the SQL Server driver to not use encryption, the ";encrypt=false" parameter needs to be present in the connection string. For more information see "Known Issues" on page 126.

Using the wizard for JSON files

The DataMapper wizard for JSON files helps you create a data mapping configuration for a simple JSON file. The wizard provides limited options for specifying source records and boundaries, and adds one extraction step.

The wizard cannot create detail tables. If the file contains transactional data, create the data mapping configuration without a wizard (see "Creating a new data mapping configuration" on page 338), and add the extraction steps yourself.

The topic Setting up detail tables with JSON describes how to turn repeating items in an array or object in JSON into detail tables.

There are two ways to open a JSON file with a wizard: from the Welcome screen or from the File menu.

- From the Welcome screen

    1. Open the OL Connect **Welcome** page by clicking the Home icon at the top right or select the **Help** menu and then **Welcome**.
    2. Click **New** at the left..
    3. Click **Data** at the right and select **JSON**.
    4. Click the **Browse** button and select the file you want to work with. Click **Next**.

- From the File menu

    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select **From JSON File**.
    3. Click **Next**.
    4. Click the **Browse** button and select the file you want to work with. Click **Next**.

After selecting a JSON file, specify if and how the JSON file must be split into multiple records.
This is done by selecting an object or array as **parent element**. Its direct child elements - objects and arrays, *not* key-value pairs - can be seen as individual source records. If the root is selected, there will be only one source record.

Whether source records are output as individual records depends on the **trigger**. Either:

- Select **On element** to create a new record in the output for each object or array in the parent element.
- Select **On change** to create a new record each time the value in a certain key-value pair changes. Only key-value pairs that exist at the root of a child element can be evaluated.

When you click **Finish**, a data mapping configuration will be set up with one **Extract all** step.

All data found in child elements of the selected parent element are extracted to fields at the root of the Data Model. If a value consists of an object or array, the entire object or array is extracted to one data field.

Field names are derived from keys, objects and arrays in the *first* record, but those aren't necessarily the same in a subsequent record. If following records have a different structure, for example if a record has more child elements compared to the first record, some data may not get extracted.

Any elements outside the selected parent element are repeated in the source records, but they don't get extracted automatically.

JSON Path collection dialog

Dialog appears when a repeat step is added in the JSON Data Mapper.

When a repeat step is added to a JSON datamapping, the behavior will be changed as follows:

- If the selected item does not have any iterating content, then .* JsonPath will be used automatically.

If the selected item has iterations (ie: child nodes), the following additional JsonPaths will also be selectable in the dialog:

- **Absolute JsonPath** of the selected node (example: *$.anode.invoices[*]* or *$.anode.invoices.**).
- **Relative path** (eg: *.Products[*]*) will be offered if:
  - The current position in the editor is not the root.
  - The current position remains in a direct path from the root to the selected element.
- **Deep-scan path** (eg: *..Products[*]*) will be offered if the selected node is a JSON array with a valid/not empty name.
  In addition, the node selected must not be the first level after the root. It must be at least the second level or a deeper level.

Using the wizard for PDF/VT or AFP files

Note: The option to use **AFP input** is included only in the Enterprise edition of OL Connect.

AFP input is dependent on a third party library (CDP), which allows OL Connect to run up to 4 AFP input processes on a given machine, at any given time. This license requires a stable MAC address, and therefore AFP input is not compatible with virtual servers or cloud environments that cannot provide a network interface with a stable MAC address.

The pages in PDF/VT and AFP files can be grouped on several levels. Additional information, such as TLEs (a TLE is a **T**agged **L**ogical **E**lement) in AFP files, can be attached to each level in the structure. The structure and additional information are stored in the file's metadata.

The DataMapper wizard for PDF/VT and AFP files lets you select a level to trigger the start of a new record and it also enables you to extract the additional information from the metadata. You can extract data from the content afterwards.

Tip: How to extract information from the metadata in the extraction workflow itself is explained in: "Extracting metadata" on page 392.

If the data file doesn't contain any metadata, each page is a new record - in other words, a boundary is set at the start of a new page -, which is exactly what happens when you open the file without a wizard.

You can open a PDF/VT or AFP file with a wizard using the Welcome screen or the File menu.

- From the **Welcome** screen
    1. Open the OL Connect**Welcome** page by clicking the Home icon at the top right or select the **Help** menu and then **Welcome**.
    2. Click **New** at the left.
    3. Click **Data** at the right and select **PDF/VT or AFP**.
    4. Click the **Browse** button and open the PDF/VT or AFP file you want to work with. Click **Next**.
- From the **File** menu
    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select **From PDF/VT or AFP**.
    3. Click **Next**.
    4. Click the **Browse** button and open the PDF/VT or AFP file you want to work with. Click **Next**.

After selecting the file, select the following options in the **Metadata** page:

- **Metadata record levels**: Use the drop-down to select what level in the metadata defines a record.
- **Field List**: This list displays all fields on the chosen level and higher levels in the PDF/VT or AFP metadata. The right column shows the field name. The left column displays the level on which it is located. Check any field to add it to the extraction.

Note: NOP (No Operation) records in AFP files cannot be extracted.

Click **Finish** to close the dialog and open the actual Data Mapping configuration.
On the Settings pane, you will see that the boundary trigger is set to **On metadata**. The selected metadata fields are added to the Data Model.

Note: Extracting data from a PDF that comes from a **Windows printer queue** (a PDF converted to PostScript, converted back to PDF by an Input task in Workflow) might not work because character identity information usually gets lost in the process.
The rule of thumb is: if copy-paste from Acrobat works, so will data mapping; if not, the DataMapper won't either.

Note: Rotated pages in a PDF are supported (if rotated 0/90/180/270 degrees). The Extract step will be able to extract data from horizontal and vertical lines of text on rotated pages. Motion steps (such as the Repeat step and the Goto step) however, can only work as expected if text on a page has the same orientation as the page, not when text has been rotated after the page was rotated.
The page number and rotation of a page are shown in the status bar at the bottom, next to the region selection information.

Using the wizard for XML files

The DataMapper wizard for XML files helps you create a data mapping configuration for an XML file. The wizard lets you select the type of node and the trigger that delimit the start of a new record. Next, the wizard extracts the data in one extraction step.

This wizard can also be used to extract data from a JSON file. JSON files are automatically converted to XML.
If there is no root element in the JSON it will be added and each object without a named parent element is called an 'item'.
Note that in addition to being valid, the JSON should follow naming rules for XML elements. For example, "adress_line_1:" is a valid key name in JSON, but it cannot be converted to a valid element name in XML because the colon is reserved for namespaces. For XML naming rules and best naming practices, see: XML elements on W3Schools.

The wizard cannot create detail tables. If the file contains transactional data, create the data mapping configuration without a wizard (see "Creating a new data mapping configuration" on page 338), and add the extraction steps yourself.

There are two ways to open an XML/JSON file with a wizard: from the Welcome screen or from the File menu.

- From the Welcome screen

    1. Open the OL Connect **Welcome** page by clicking the Home icon at the top right or select the **Help** menu and then **Welcome**.
    2. Click **New** at the left..
    3. Click **Data** at the right and select **XML**.
    4. Click the **Browse** button and select the file you want to work with. For a JSON file, change the file type to JSON first. Click **Next**.

- From the File menu

    1. In the menu, click **File > New**.
    2. Click the **Data mapping Wizards** drop-down and select **From XML File**.
    3. Click **Next**.
    4. Click the **Browse** button and select the file you want to work with. For a JSON file, change the file type to JSON first. Click **Next**.

After selecting a file, you have to set the split level and trigger type:

- **XML Elements**: This is a list of node elements that have children nodes. Select the level in the data that will define the source record.
- **Trigger**: Select **On element** to create a record for each occurrence of the node element selected in the **XML Elements** field, or select **On change** to create a record each time the element is different. (Check the option to include attributes in the list of content items that can trigger a boundary.)

Note: The DataMapper only extracts elements for which at least one value is defined in the file. Attribute values are not taken into account.
Attribute values (prefixed with an @ sign in the Data Viewer) are not extracted automatically.

Click **Finish** to close the dialog and open the data mapping configuration.

Providing missing fonts for PDF

If a PDF input file uses a font that is not embedded in the PDF itself, this may cause problems in OL Connect. Not only can the output look bad, it can also cause DataMapper's text extraction to produce incorrect data. These issues may be resolved by providing OL Connect with the missing fonts or replacement fonts. This feature is available from version 2023.1.

Note that OL Connect will not use system fonts when a font is missing, because systems are not guaranteed to have the same fonts. This has the added benefit that you will definitely notice it when the input requires external fonts.

Providing missing fonts

1. Copy the fonts to the folder *C:\ProgramData\Objectif Lune\OL Connect\Resources\PDF\fonts* on the system on which the OL Connect Server is located.
2. Create a file named **Fontmap** in the parent folder: *C:\ProgramData\Objectif Lune\OL Connect\Resources\PDF*
3. In the Fontmap file, map each of the missing fonts to the appropriate font file in the *fonts* subfolder. The syntax is: `font "font name" "font file"` (one font per line).

   Example:
   font "GalanoGrotesque-Regular" "RENE_BIEDER-GALANO_GROTESQUE.OTF"
   font "GalanoGrotesque-Bold" "RENE BIEDER - GALANO GROTESQUE BOLD.OTF"
   font "GalanoGrotesque-Italic" "RENE BIEDER - GALANO GROTESQUE ITALIC.OTF"

Providing a substitute font

If you don't have the missing font, but you do have a font that is very similar - for example, the semi-bold italic version instead of the bold italic version - you can use this as a replacement.

1. Add the font you want to use as a substitute to your Fontmap file using the same procedure as for missing fonts.

   Example:
   font "GalanoGrotesque-SemiBoldItalic" "RENE BIEDER - GALANO GROTESQUE SEMIBOLD ITALIC.OTF"

2. On the next line, link the missing font to the substitute font.
   The syntax is: `alias "font name" "substitute font name"`.

   Example:
   alias "GalanoGrotesque-BoldItalic" "GalanoGrotesque-SemiBoldItalic"

Note: Even if fonts are likely very similar, it is advisable to check whether the result is acceptable and text is extracted correctly.

Advanced PCL to PDF options

Note: The option to use PCL input can be purchased with the Professional and Enterprise editions of OL Connect.

The Connect DataMapper uses **LincPDF** to convert PCL files to PDF files. By default, the only options it passes to **LincPDF** are the input and output file names. Therefore, everything depends on **LincPDF**'s own default settings. However, in order to extract data from a PCL file it might be necessary to take some of the options into account that were set to create the PCL spool file, such as *Edge-to-Edge*, *page X and Y offset values*, *page width* and *height* in the case of a custom page type, or the *number of lines per page* to generate the correct number of PDF pages.

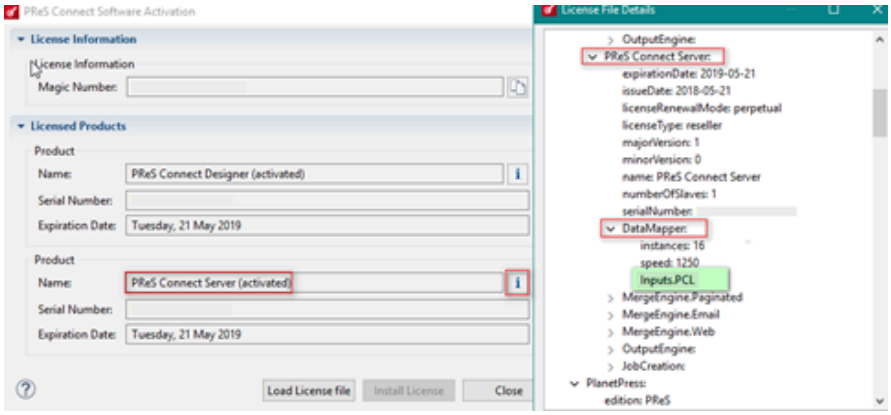
This topic's intention is to provide you with a method to make LincPDF take these advanced settings into account, so the generated PDF can be correctly read in the DataMapper.

Note: The method described in this topic can only be done using OL Connect Workflow.

Requirements

To be able to import PCL input files, you will need the PCL Input license, in addition to OL Connect.
To check if you have a PCL Input license:

1. Open the Connect Software Activation module.
2. Click on the "i" icon next to the OL Connect Server product.
3. When the **License File Details** window opens, expand the OL Connect Server, then expand the DataMapper node. If the *Inputs.PCL* entry is present in that node, you have a PCL Input license.



Calling LincPDFC via an external program

To set advanced conversion parameters, the PCL input file has to pass through an additional Workflow process before entering the process in which the data mapping takes place. The extra Workflow process should call the **LincPDF** command line module **LincPDFC** via the **External Program** plugin with the desired advanced conversion parameters. To create such a Workflow process:

1. Add a local variable to your process and name it **lincPDFOptions**.
2. Add another local variable named **workingDir** and give it a default Windows path (for example: C:\PCL2PDF\).
3. Add a plugin to capture the PCL File. You may use any input plugin that imports the PCL file into the process, such as Folder Capture, LPD Input, etc.
4. Using the Change Emulation plugin, change the Emulation to ASCII. This type of emulation allows to set the number of lines per page, remove or add lines or even remove some HP PCL escape sequences if necessary.
5. As the number of options you pass to **LincPDFC** may be high, it is recommended to assign all the desired options to the local variable **lincPDFOptions**. To do that you can use either the **Set Job Infos and Variables** plugin or a **Run Script** Action plugin, for example with the following JavaScript:

```
Watch.SetVariable("lincPDFOptions","-z:1 -*e -*f -u3 -s -pXOff:-0.155 -pYOff:0.15 -dAuthor:\"OL Connect\" -dTitle:\"" + Watch.Ex-
pandString("%O") + "\"");
```

This example sets the following options or parameters:

Key Name	Argument	Value Type	Description
EdgeToEdgePrinting	-z	0 or 1	Set the flag of "Allows edge-to-edge printing" (default: FALSE).
Constant Alpha	-n:num	0-1	Specify constant alpha value defined in PDF 1.4 (default: 0.5).
UsePolygons	*e	0 or 1	When enabled, Lincoln's PCL interpreter will output vector graphics in a simpler mode (default: FALSE).
OutputRgb	*f	0 or 1	When set (1), LincPDF will convert CMYK as RGB colorspace before writing (default: FALSE).

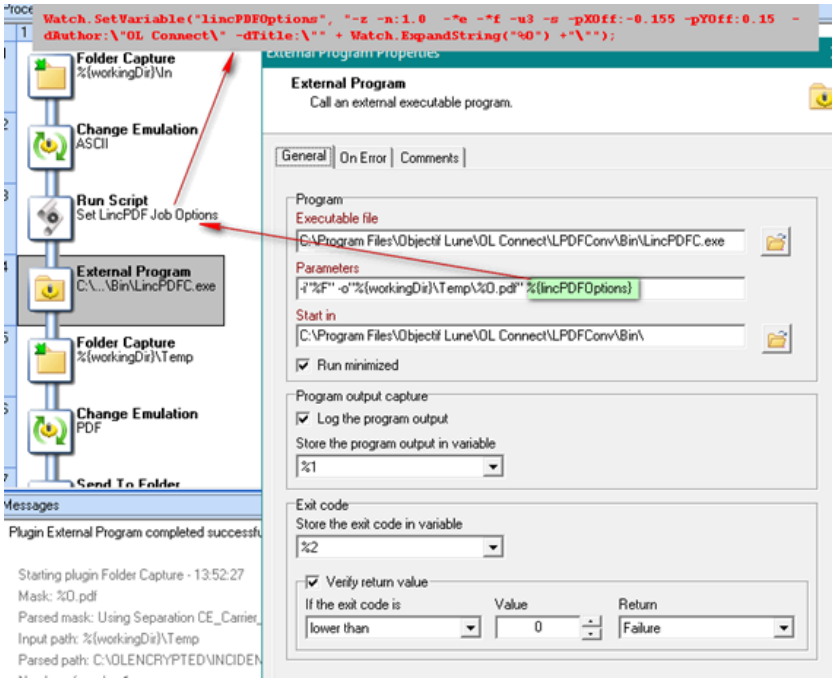
Key Name	Argument	Value Type	Description
	-u3		Emulate 300 DPI printers
	-s		Ignore Redundant Font ID
Page X Offset	-pXOff:num	number	Specifies page's X-direction offset (default: 0).
Page Y Offset	-p YOff:num	number	Specifies page's Y-direction offset (default: 0).
PDF Author	-dAuthor:$s	string	PDF document's author (default: null).
PDF Title	-dTitle:$s	string	PDF document's title (default: filename).

6. Next, use the External Program plugin to convert the PCL to PDF with the above **%{lincPDFOptions}** options.

   In the **General** tab:

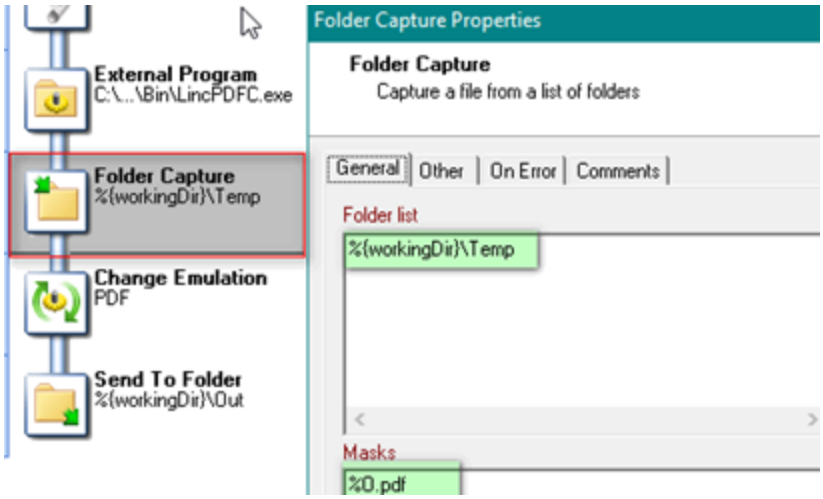
   - Set the Executable file to: *C:\Program Files\Objectif Lune\OL Connect\LPDFConv\Bin\LincPDFC.exe*
   - Set the Parameters to: *-i"%F" -o"%{workingDir}\Temp\%O.pdf" %{lincPDFOptions}*
   - As a minimum, **LincPDFC** requires an input file, which is set here via the parameter *–i* to which we are assigning the current job file path and name "**%F**", and an output file path and name set here using the *–o* parameter with a dynamic value of "**%{workingDir}\Temp\%O.pdf**".
   - Select the "*Run minimized*" option to run the program in the background
   - You may select to **Log** the program output for additional troubleshooting. This is optional; however when activated **LincPDFC** detailed conversion will be added to

the Workflow log.



7. Save the output of LincPDFC using the *–o* folder specifier in the parameter (*%{workingDir}\Temp*, in this case.)

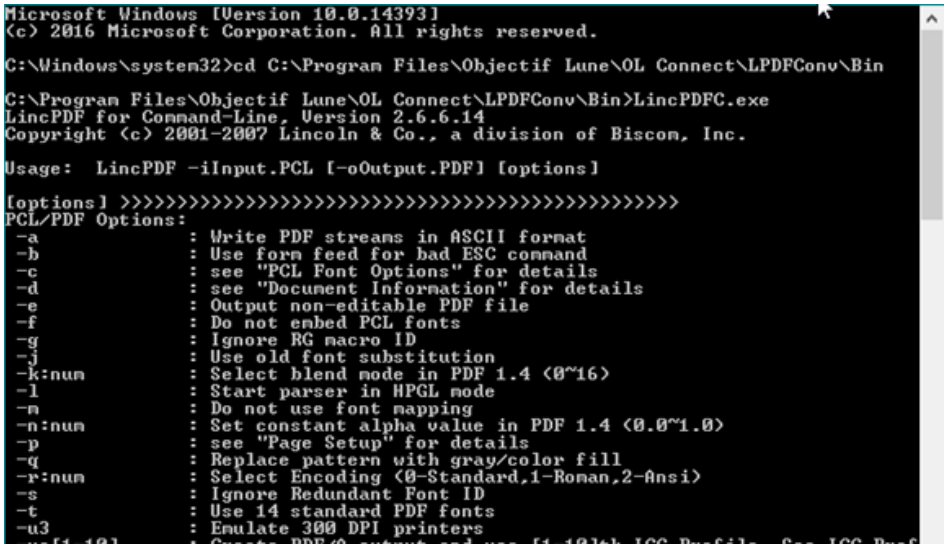
In another Workflow process, import the created PDF with the **Folder Capture** input plugin, specifying the output folder of the previous process (*%{workingDir}\Temp* in the example) as input folder, and *%O.pdf* as the file mask.



Once you have the PDF as job file, you may pass it to the Execute Data Mapping plugin for further processing.

LincPDFC Options

To view the available options that can be set in **LincPDF**, run the executable (**LincPDFC.exe**) in a command prompt window. It will display a help message with available options.




```
/*Open Windows Command Prompt
 Change directory
 cd C:\Program Files\Objectif Lune\OL Connect\LPDFConv\Bin
 C:\Program Files\Objectif Lune\OL Connect\LPDFConv\Bin\LincPDFC.exe
 LincPDF for Command-Line, Version 2.6.6.14
 Copyright (c) 2001-2007 Lincoln &amp;amp; Co., a division of Biscom, Inc.
 Usage: LincPDF -iInput.PCL [-oOutput.PDF] [options]
 [options]
 PCL/PDF Options:
 -a : Write PDF streams in ASCII format
 -b : Use form feed for bad ESC command
 -c : see "PCL Font Options" for details
 -d : see "Document Information" for details
 -e : Output non-editable PDF file
 -f : Do not embed PCL fonts
 -g : Ignore RG macro ID
 -j : Use old font substitution
 -k:num : Select blend mode in PDF 1.4 (0~16)
 -l : Start parser in HPGL mode
 -m : Do not use font mapping
 -n:num : Set constant alpha value in PDF 1.4 (0.0~1.0)
 -p : see "Page Setup" for details
 -q : Replace pattern with gray/color fill
```

-r:num : Select Encoding (0-Standard,1-Roman,2-Ansi)
-s : Ignore Redundant Font ID
-t : Use 14 standard PDF fonts
-u3 : Emulate 300 DPI printers
-ua[1-10] : Create PDF/A output and use [1-10]th ICC Profile. See ICC Profiles for details
-v : View the result PDF file
-w : see "Watermark Options" for details
-x : Duplex printing for even pages output
-y : see "PDF Security Options" for details
-z : Allow edge-to-edge printing
-*a# : Auto Scan Direction: 1 for on and 0 for off
-*b : Skip same macro RG w/o RG creation
-*c : Use SMPTE RGB colorspace
-*d : Suppress blank pages
-*e : Use Polygons
-*f : Output CMYK as RGB colorspace

Document Information:

-dTitle:$s : PDF Title
-dSubject:$s : PDF Subject
-dAuthor:$s : PDF Author
-dKeywords:$s : PDF Keywords
-dVersion:num : PDF Version (multiply by 10)

Page Setup:

-pWidth:num : Page Width (required only if Page Type is Custom)
-pHeight:num : Page Height (required only if Page Type is Custom)
-pXOff:num : Page X Offset (see also Measurement)
-pYOff:num : Page Y Offset (see also Measurement)
-pMeasure:num : Page Measurement (0-inch, 1-mm, 2-point)
-pOrient:num : Page Orientation (0-Portrait, 1-Landscape)
-pType:num : Page Type (0-Letter, 1-A4, 2-B5, 3-Legal, 4-Exec., 5~8-Env., 9-Tabloid, 10-A3, 11-Custom, 12-A5)

PCL Font Options:

-cPoint:num : PCL Font Point (0.25~999.75)
-cPitch:num : PCL Font Pitch (0.1~576)
-cNumber:num : PCL Font Number (0~1000)
-cSymbolSet:$s : PCL Font SymbolSet (e.g. 8U-US, 10U-EU)
-cSource:num : PCL Font Source (0-internal, 1-left, 2-right, 3-soft)
-cTextPath:num : PCL Font TextPath (0-horizontal, 1-vertical)

```
 -cEncoding:num : PCL Font Encoding (see Encoding in LincPDFContrl.doc)
 Watermark Options:
 -wApply:num : Watermark Apply to (0-Whole, 1-Por~Land, 2-Odd~Even, 3-
1st~Res
 t)
 -wType:num : Watermark Type (0-Text, 1-Image)
 -wFont:num : Watermark Font type (0~13)
 -wStr:$s : Watermark Text String
 -wSize:num : Watermark Font Size (1~1000)
 -wGray:num : Watermark Text Grayscale (0-darkest ~ 100-lightest)
 -wDir:num : Watermark Text Direction (0~7)
 -wPath:$s : Watermark Image File Path
 -wPos:num : Watermark Image Position (0~2)
 -wOnTop: Watermark on Top
 PDF Security Options:
 -yEnable : Enable PDF Encryption
 -yUserPassword:$s : Open PDF Password
 -yOwnerPassword:$s : Change PDF Permission Password
 -yPrintDocument: Enable Document Printing
 -yChangeDocument : Enable Document Changing
 -yAddAnnotations : Enable Adding Text Annotations
 -yCopyContents : Enable Copying Text and Graphics from Document
 -yUse128Bit : Use 128-bit Encryption
 -yAssembleDocument : Enable Assemble Document (128-bit encryption only)
 -yExtractText : Enable Text and Graphics Extraction (128-bit encryption
only)
 -yLowResolutionPrint : Enable Lower-level Resolution Printing (128-bit
encryption only)
 Tips ------------------------------------------------
 . using quotation mark for complicated string, for example,
 -dKeywords:"key1, key2"
 . you can convert multi- PCL files at one time, for instance,
 -iFile1.pcl -iFile2.pcl
 . for second watermark, please add '1' after each keyword, for example,
 -wSize:10 to set first watermark font size, -wSize1:12 for the second.
 . ICC Profiles for output intent are stored in the folder pointed to by
"ICCPath
 " defined
 in the register or in the default folder c:\program files\lincoln\lincpd
```

```
f and the
profiles info are described in the text file: CMYKICCProfiles.txt in the
same folder.
C:\Program Files\Objectif Lune\OL Connect\LPDFConv\Bin
*/
```

Preprocessing AFP input

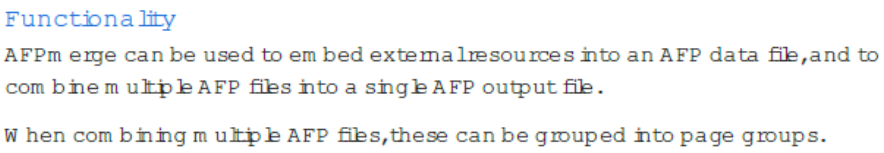
> **Note:** Option is only available in the **Enterprise** edition of OL Connect.

In some cases it is useful or even necessary to pre-process AFP input files before OL Connect extracts data from them. This topic explains why and how to preprocess AFP input.

Embedding external resources

AFP data streams often use external files: overlays, page segments, presentation pages, form definitions, page definitions, coded fonts, character sets, code pages, TrueType fonts, and more.

If resources are missing from AFP input files, it can make it impossible to extract data from them. The content will not look as intended and will therefore no longer be fit for further processing. In case of a missing font, for example, text may be converted into another font and become too long for a table cell or overlap with other text.


Functionality

AFPmerge can be used to embed external resources into an AFP data file, and to combine multiple AFP files into a single AFP output file.

When combining multiple AFP files, these can be grouped into page groups.

Assuming you have the corresponding source files, there are two solutions: you can put the source files in a folder where OL Connect will find them, or you can embed them in the AFP.

> **Note:** External resources are picked up immediately after they have been made available. There is no need to restart the Designer or OL Connect Server. However, if a *previously imported or processed* AFP input file is not formatted correctly after you add or embed source files, this is likely because the results of AFP input processing are cached. Both OL Connect Server and Designer (DataMapper) cache the results of AFP input processing. In that case, make sure to restart both.

Providing resources via a folder

The OL Connect Server will automatically use external AFP resources if it can find the files in the following folder:

C:\ProgramData\Objectif Lune\OL Connect\Resources\AFP

It will also search the following subfolders of that folder for resource files: FormDef, PageDef, Overlay, PageSeg, PresentPage, CodedFont, Charset, Codepage, TrueType.

Although these folder names suggest a certain resource type, it actually doesn't matter in which folder a particular resource type is stored.

The extension of a file name is ignored, so there is no need to change extensions of source files.

Note: This will also work in Designer (DataMapper) since all AFP processing is actually done by OL Connect Server.

Embedding resources with AFPMerge

AFPMerge is a command-line tool that can embed external resources in one or more AFP files, merge multiple AFP files into one, and indicate what form definition to include in the output AFP.

The tool is installed with OL Connect Enterprise when either the OL Connect Designer or Server component is installed.

AFPMerge can be found in the OL Connect installation folder:
C:\Program Files\Objectif Lune\OL Connect\AFPMerge\bin

To run AFPMerge with a single AFP file, open the Command Prompt and use the following command:

"C:\Program Files\Objectif Lune\OL Connect\AFPMerge\bin\AFPMerge.exe" -i <inputfile> -r <resource folder> -o myoutputfile.afp

- **-i** specifies an input file.

  AFPMerge can also be used to embed resources in a *collection* of AFP files which will then be merged into one file. How to do this is explained below, in "Merging AFP files" on the next page.

- **-r** specifies the folder in which the files to embed are stored.

  Files to be embedded do not have to have a file extension. AFPMerge looks up files by name, first without an extension, and then with file extensions that are accepted with the respective resource type.

- **-o** specifies the name of the output file.

Example:

```
"C:\Program Files\Objectif Lune\OL Connect\AFPMerge\bin\AFPMerge.exe" -
```

```
i c:\input\myinputfile.afp -r c:\myresourcefolder -o
c:\output\myoutputfile.afp
```

Tip: The command "C:\Program Files\Objectif Lune\OL Connect\AFPMerge\bin\AFPMerge.exe" **-h** (or **--help**) displays a list of all of AFPMerge's command line options and what they do.

Merging AFP files

Extracting data from many small AFP files that have the same structure will be less efficient than when those files are combined in a single AFP file.

AFPMerge, a command-line tool that is designed to embed external resources into AFP files, can merge AFP files at the same time.

To run AFPMerge on a collection of AFP input files:

1. Create a *manifest file*. This is a text file that contains the list of the AFP files that must be merged and, optionally, the grouping key of each (tab-delimited). See "Manifest file format" on the facing page.
2. Open the Command Prompt and use the following command:

   "C:\Program Files\Objectif Lune\OL Connect\AFPMerge\bin\AFPMerge.exe" -l <manifest file> -r <resource folder> -o myoutputfile.afp

   - **-l** (lowercase L) specifies the manifest file.
   - **-r** specifies the folder in which the files to embed are stored. This is required; however, if the given resource folder does not contain any resource files, the tool will proceed and only merge the files listed in the manifest file.
   - **-o** specifies the name of the file into which the listed files are merged.

   Example:

   ```
   "C:\Program Files\Objectif Lune\OL
   Connect\AFPMerge\bin\AFPMerge.exe" -l c:\input\mymanifest.txt -r
   c:\myresourcefolder -o c:\output\myoutputfile.afp
   ```

   Tip: In the Command Prompt, a . (dot) at the start of (or instead of) a folder name indicates the current folder.

Manifest file format

The manifest file (used with the -l option of AFPMerge) must be a UTF-8 encoded file that lists one file per line.

Example:
print75X-1.AFP
print75X-12.AFP
print75X-3.AFP
print75X-34.AFP
print75X-5.AFP
print75X-56.AFP
print75X-7.AFP
print75X-78.AFP

AFPMerge will wrap each file in a page group, and that page group gets a Tagged Logical Element (TLE) named GROUP, with its value set to the input filename.
Because DataMapper can set boundaries using AFP page groups, the combined AFP files can be easily processed into one record per input file.

Optionally, the file can have two columns separated by a tab.

Example:
print75X-1.AFP doc1
print75X-12.AFP doc1
print75X-3.AFP doc1
print75X-34.AFP doc2
print75X-5.AFP doc2
print75X-56.AFP doc3
print75X-7.AFP doc3
print75X-78.AFP doc3

When a second column is present, AFPMerge uses it as a **grouping key** and puts files with the same grouping key in the same page group. Each page group gets a TLE named GROUP that holds the value of the grouping key.

The above example will create a combined AFP file with 3 page groups that will map to 3 records in DataMapper.

Tip: TLEs can be mapped to fields in DataMapper, so it is easy to pick up the grouping key and use that in subsequent processing.

Specifying a form definition

AFP files can have multiple form definitions of which only one can be included in the output. AFPMerge allows to select the form definition that is to be included in the output with the option **-f** or **--formdef**.

For example:

"C:\Program Files\Objectif Lune\OL Connect\AFPMerge\bin\AFPMerge.exe" -l c:\input\mymanifest.txt -r c:\myresourcefolder -o c:\output\myoutputfile.afp -f F1MYFDEF

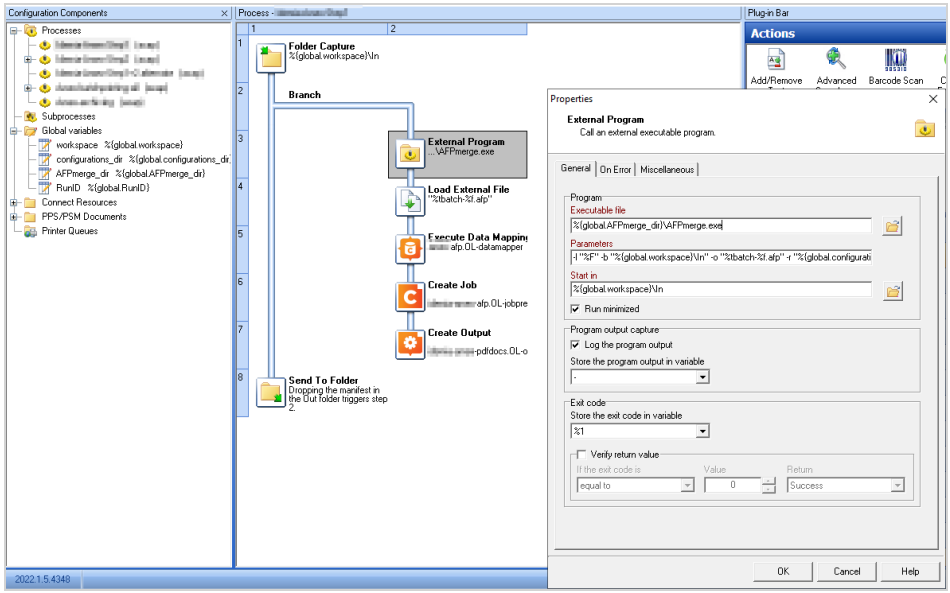
Preprocessing AFP files in a Workflow process

To preprocess AFP files in a Workflow process, AFPMerge can be run with the **External Program** action task.

- In the task properties, select to run the program minimized.
- Enter the command line options in the **Parameters** field. Enclose file and folder names in double quotes.
- If you want log messages to be included in OL Connect log files, check the **Log the program output** option.

> **Note:** AFPMerge has several log levels. For information on setting the log level, see "AFPMerge's log level options" on the facing page.

The output of AFPMerge can be loaded using the **Load External File** action task, which can pass the file to the OL Connect **Execute Data Mapping** task.



AFPMerge's log level options

When AFPMerge is invoked without a log level option, error and warning messages are logged, as well as info messages that occur once per run. Info messages that are repeated for each input file or even more often, are not shown.

The following command line options change the log level.

- -q, --quiet puts logging in quiet mode. When invoked with the -q option, AFPmerge shows no log messages.
- -v, --verbose turns on verbose logging. When invoked with the -v option, AFPmerge logs error, warning, and info messages.
- -d, --debug turns on detailed logging. When invoked with the -d option, AFPmerge logs error, warning, info, debug, and trace messages.

Data mapping workflow

A **data mapping workflow** is a series of extraction instructions, called **steps**. These steps process and extract the data from the source and store them in records whose structure is determined in the Data Model (see "The Data Model" on page 419). Together with the data source settings, the Data Model, and the sample data, this is what makes a data mapping configuration (See "Data mapping configurations" on page 337).

The data mapping workflow is shown on the Steps pane at the left (see "Steps pane" on page 486).

Creating a data mapping workflow

A data mapping workflow always starts with the **Preprocessor** step and ends with the **Postprocessor** step. These steps allow the application to perform actions on the data file itself before it is handed over to the data mapping workflow ("Preprocessor step" on page 410) and after the Data Mapping workflow has completed ("Postprocessor step" on page 418).
When you create a new data mapping configuration, these steps are added automatically, but they don't actually do anything until you configure them.
In between the Preprocessor and Postprocessor step, the workflow can contain as many steps as needed to extract the required data.

Defining properties and runtime parameters

Before adding steps, define the properties and runtime variables that you will need in the data mapping configuration.

A data mapping configuration's **properties** hold data that can be used throughout the data mapping workflow to compare against in conditions or to complement the existing data. **Runtime parameters** are properties that will pass information from OL Connect Workflow to the data mapping configuration.

All properties are defined in the **Preprocessor** step.

For more information see "Properties and runtime parameters" on page 387.

Adding steps

Extracting data is the main way to build a data mapping workflow; see "Extracting data" on page 390.
Extract steps, Condition steps and Repeat steps can be added after selecting data in the Data Viewer.
All steps can be added via the Steps pane:

1. In the extraction workflow on the **Steps** pane, select the step after which to add the new step.
2. Right-click on the Steps pane and select **Add a Step**; then select one of the step types.

To add a first Step inside a Repeat or Condition structure, click on the initial step of the Repeat/Condition structure. This highlights the entire structure and any Step you add now would be positioned immediately after the structure. Click once more on the structure's initial step to highlight it alone. The next Step added will now be positioned as the first child inside the structure.

Editing steps

The properties of each step in the extraction workflow become visible in the **Step properties** pane when you select that step in the Steps pane.

The **name** of each step is shown in the Steps pane. You can change it under **Description** in the Step properties pane.

The other properties are different per step type; see "Steps" on page 409.

Rearranging steps

To rearrange steps, simply drag & drop them somewhere else on the colored line in the Steps pane.
Alternatively you may right-click on a step and select **Cut Step** or use the Cut button in the **Toolbar**. If the step is **Repeat** or **Condition**, all steps inside it will also be placed on the clipboard. To place the step at its destination, right-click any step and select **Paste Step**, or

use the Paste button in the toolbar. The pasted steps will be positioned below the selected step.

Keep in mind that steps may influence each other, so you may have to move other steps as well to ensure that the workflow continues to function properly. In a Text file for example, an Extract step may require a Goto step to be positioned immediately before it to move the current position to a certain location in the source data.

Deleting steps

To delete a step, right-click on it in the Steps pane and select **Delete Step**.

Testing the extraction workflow

The extraction workflow is always performed on the current record in the data source. When an error is encountered, the extraction workflow stops, and the data field on which the error occurred and all data fields related to subsequent steps will be greyed out. Click the **Messages** tab (next to the Step properties pane) to see any error messages.

If you want to test a data mapping workflow on **all** records that are displayed, and see how the steps perform, read Testing a data mapping workflow.

> **Note:** At design time, steps automatically error out after the timeout value has been reached that is set in the preferences. This does not apply to the Preprocessor and Postprocessor steps and when the boundaries are being set. See "Common DataMapper preferences" on page 1039.

Data source settings

After opening a data file you have to make a number of settings to make sure that the source data is interpreted and grouped the way you want. These settings are found on the **Settings** pane at the left.

- "Input data settings (Delimiters)" on the next page help the DataMapper read the data source and recognize data correctly.
- "Record boundaries" on page 373 mark the start of a new record. They let you organize the data, depending on how you want to use them.
- "Data format settings" on page 373 define how dates, times and numbers are formatted by default in the data source.

Input data settings (Delimiters)

The **Input Data** settings (on the **Settings** pane at the left) specify how the input data must be interpreted. These settings are different for each data type. For a CSV file, for example, it is important to specify the delimiter that separates data fields. PDF files are already delimited naturally by page, so the only input data settings for PDF files are instructions on how to parse the text on each page.
For an overview of all options, see: "Input Data settings" on page 472.

Settings for a CSV File

In a CSV file, data is read line by line, where each line may contain multiple fields, separated by a **delimiter**. Even though CSV stands for comma-separated values, fields may be separated using any character, including commas, tabs, semicolons, and pipes.
The **text delimiter** is used to wrap around each field just in case the field values contain the field separator. This ensures that, for example, the field “Smith; John” is not interpreted as two fields, even if the field delimiter is the semicolon.
For an explanation of all the options, see: "Input Data settings" on page 472.

Settings for an Excel File

For an Excel file you have to specify which sheet to use, or provide a dynamic sheet name that will be populated at runtime. You can also set how many lines should be skipped, if the first row contains field names or not, and how the data should be sorted. See: "Excel file Input Data settings" on page 473.

Excel has its own way to display dates. You can specify if the Data Viewer should display dates just as Excel does, or not. That's important because extracting a Date value will only be successful if the expected date format matches the actual format of a date in the Data Viewer.
This Editor Data Format setting is found on the Settings pane; see: "Editor Data Format" on page 470.
The expected date format is set somewhere else; see: "Default Data Format" on page 471.

Settings for a PDF File

PDF files have a clear and unmovable delimiter: pages. So, the Input Data settings are not used to set delimiters. Instead, these options determine how words, lines and paragraphs are detected when you select content in the PDF to extract data from it.
For an explanation of all the options, see: "PDF file Input Data settings" on page 474.

Settings for a database

Databases all return the same type of information. Therefore the Input Data options for a database refer to the tables inside the database. Clicking on any of the tables shows the first line of the data in that table.
If the database supports stored procedures, including inner joins, grouping and sorting, you can use **custom SQL** to make a selection from the database, using whatever language the database supports. The query may contain variables and properties, so that the selection will be dynamically adjusted each time the data mapping configuration is actually used in a Workflow process; see "Using variables and properties in an SQL query" on page 486.
For an explanation of all the options, see: "Database Input Data settings" on page 476.

Settings for a text file

Because text files have many different shapes and sizes, there are a lot of input data settings for these files. You can add or remove characters in lines if it has a header you want to get rid of, or unwanted characters at the beginning of your file, for example; you can also set a line width if you are still working with old line printer data; etc.
It is important that pages be defined properly. This can be done either by using a set number of lines or using a string of text (for example, the character “P”), to detect on the page. Be aware that this is not a Boundary setting; it detects each new page, not each new record.
For an explanation of all the options, see: "Text file Input Data settings" on page 477.

Settings for an XML file

XML is a special file format because these file types can have a theoretically unlimited number of structure types. The input data has three options that basically determine at which node level a new record is created. You can:

- Select an element type to create a new delimiter every time that element is encountered.
- Enter an XPath to create a delimiter based on the node name of elements. The Show all elements option allows you to extract information from all elements, even when the delimiter is set to a specific lower-level element.
- Use the root node. If there is only one top-level element, there will only be one record before the Boundaries are set.

See also: "XML File Input Data settings" on page 478.

Note: The DataMapper only extracts elements for which at least one value is defined in the file.

Settings for a JSON file

For JSON files there are two input data options that determine where a new source record starts.
You can either use the object or array at the root and get one output record, or select an object or array as **parent element**. Its direct child elements - objects and arrays, *not* key-value pairs - are then seen as individual source records. Any elements at the same level as the parent element or at a higher level are repeated in each source record.

See also: "JSON File Input Data settings" on page 480.

Record boundaries

Boundaries are the division between **records**: they define where one record ends and the next record begins. Using boundaries, you can organize the data the way you want.
You could use the exact same data source with different boundaries in order to extract different information. If, for instance, a PDF file contains multiple invoices, each invoice could be a record, or all invoices for one customer could go into a single record.
Keep in mind that when the data is merged with a template, each **record** generates output (print, email, web page) for a **single** recipient.

To set a boundary, a specific **trigger** must be defined.
The trigger can be a natural delimiter between blocks of data, such as a row in a CSV file or a page in a PDF file.
It can also be something in the data that is either static (for example, the text "Page 1 of" in a PDF file) or changing (a customer ID, a user name, etc).
To define a more complex trigger, you can write a script (see "Setting boundaries using JavaScript" on page 535). This option is not available with XML and JSON files.
A new record cannot start in the middle of a data field, so if the trigger is something in the data, the boundary will be set on the nearest preceding natural delimiter. If for instance in a PDF file the text "Page 1 of" is used as the trigger, the new record starts at the page break before that text.

For an explanation of all Boundaries options per file type, see "Boundaries settings" on page 480.

Data format settings

By default the data type of extracted data is a HTMLString, but each field in the Data Model can be set to contain another data type (see "Data types" on page 436). When that data

type is Date, Number or Currency, the DataMapper will expect the data in the data source to be formatted in a certain way, depending on the settings.

The default format for dates, numbers and currencies can be set in three places: in the user preferences, in the data source settings, and per field in the Data Model.

By default, the user preferences are set to the system preferences. These user preferences become the default format values for any newly created data mapping configuration. To change these preferences, select **Window > Preferences > DataMapper > DataMapper default format** (see "DataMapper preferences" on page 1039).

Data format settings defined for a data source apply to any **new** extraction made in the **current** data mapping configuration. These settings are made on the Settings pane; see "Settings pane" on page 468.

Settings for a field that contains extracted data are made via the properties of the Extract step that the field belongs to (see "Setting the data type" on page 429). Any format settings specified per field are always used, regardless of the user preferences or data source settings.

Note: Data format settings tell the DataMapper how to read and parse data **from the data source**. They don't determine how these data are formatted in the Data Model or in a template. In the Data Model, data are converted to the native data type. Dates, for example, are converted to a DateTime object. How they are displayed in the Data Model depends on the preferences (see "Default Format" on page 1039).

Input Data settings

The **Input Data** settings specify how the input data must be interpreted. These settings are different for each data type. For a CSV file, for example, it is important to specify the delimiter that separates data fields. PDF files are already delimited naturally by pages, so the input data settings for PDF files are interpretation settings for text in the file.

- "Input Data settings" above
- "Excel file Input Data settings" on the next page
- "PDF file Input Data settings" on page 376
- "Database Input Data settings" on page 377
- "Text file Input Data settings" on page 378
- "XML File Input Data settings" on page 380
- "JSON File Input Data settings" on page 381

CSV file Input Data settings

In a CSV file, data is read line by line, where each line can contain multiple fields. The input data settings specify to the DataMapper module how the fields are separated.

- **Field separator**: Defines what character separates each field in the file. Even though CSV stands for comma-separated values, CSV can actually refer to files where fields are separated using any character, including commas, tabs, semicolons, and pipes.
- **Text delimiter**: Defines what character surrounds text in the file, preventing the **Field separator** from being interpreted within those text delimiters. This ensures that, for example, the field "Smith; John" is not interpreted as two fields, even if the field delimiter is the semicolon.
- **Comment delimiter**: Defines what character starts a comment line.
- **Encoding**: Defines what encoding is used to read the Data Source (US-ASCII, ISO-8859-1, UTF-8, UTF-16, UTF-16BE or UTF-16LE).
- **Lines to skip**: Defines a number of lines in the CSV that will be skipped and not used as records.
- **Set tabs as a field separator**: Overwrites the **Field separator** option and sets the Tab character instead for tab-delimited files.
- **First row contains field names**: Uses the first line of the CSV as headers, which automatically names all extracted fields.
- **Ignore unparseable lines**: Ignores any line that does not correspond to the settings above.
- **Skip empty lines**: Ignore any line that has no content. Note that spaces are considered content.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

Excel file Input Data settings

There are no settings for field separation in an Excel file, only settings with regards to the file as a whole.

- **Lines to skip**: Defines a number of lines in the Excel file that will be skipped and not used as records.

- **First row contains field names**: Check this option to use the first line of the Excel file as headers. This option automatically names all extracted fields.
- **Sheet**: Only one sheet can be selected as the data source. Enter a sheet name, or click the {} Dynamic Sheet Name icon to compose a dynamic sheet name that will be populated at runtime. A dynamic sheet name starts with an equals sign (=), and any variable values (i.e. job info variables and runtime parameters, see "Properties and runtime parameters" on page 387) must be enclosed within curly brackets: {}.

  Example: =sheet{automation.parameters.ID}
  This expression resolves to "sheet1" if the runtime parameter ID is set to 1.

  Tip: Double-click a parameter to add it to the expression.

  If an expression resolves to a pure integer (e.g. 1, 19, 42) the DataMapper will attempt to open a sheet whose name is exactly that (e.g. "1", "19", "42"). If the sheet with that name does not exist, the DataMapper attempts to open the sheet whose **index** matches the integer (e.g. the first, 19th or 42nd sheet).
- **Skip empty lines**: Ignore any line that has no content. Note that spaces are considered content.
  - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

PDF file Input Data settings

These settings also apply to files that are converted to PDF by the DataMapper: PS, PCL and AFP files.

PDF Files have a natural, static delimiter in the form of pages, so the options here are interpretation settings for text in the PDF file.
The Input Data settings for PDF files determine how words, lines and paragraphs are detected in the PDF when creating data selections.
Each value represents a fraction of the average font size of text in a data selection, meaning "0.3" represents 30% of the height or width.

- **Word spacing**: Determines the spacing between words. As PDF text spacing is somehow done through positioning instead of actual text spaces, text position is what is used to find new words. This option determines what percentage of the average

width of a single character needs to be empty to consider a new word has started. The default value is 0.3, meaning a space is assumed if there is a blank area of 30% of the width of the average character in the font.

- **Line spacing**: Determines the spacing between lines of text. The default value is 1, meaning the space between lines must be equal to at least the average character height.
- **Paragraph spacing**: Determines the spacing between paragraphs. The default value is 1.5, meaning the space between paragraphs must be equal to at least 1.5 times the average character height to start a new paragraph.
- **Magic number**: Determines the tolerance factor for all of the above values. The tolerance is meant to avoid rounding errors. If two values are more than 70% away from each other, they are considered distinct; otherwise they are the same. For example, if two characters have a space of exactly the width of the average character, any space of between 0.7 and 1.43 of this average width is considered one space. A space of 1.44 is considered to be 2 spaces.
- **PDF file color space**: Determines if the PDF if displayed in Color or Monochrome in the Data Viewer. Monochrome display is faster in the Data Viewer. This has no influence on the actual data extraction or the data mapping performance.
- **Processing method**: This option determines which search method is used by the Goto step's **Next Occurrence of** option. The most recent method searches for text within the specified constraints. It is more precise and more reliable than the **Original** method which searches for the target text in the entire page and then determines if the text appears within the specified constraints.
  Both methods may sometimes return slightly different values. Data mapping configurations made with previous version of the software are therefore set to use the original method by default. It is however recommended to modify them to use the newer method.
- **Auto-Rotation** (PS files only): If in the original PostScript file the orientation of a page and its text don't match, this option sets the page's orientation to match the orientation of the text.

Database Input Data settings

Databases all return the same type of information. Therefore the Input Data options for a database refer to the database itself instead of to the data.
The following settings apply to any database or ODBC Data Sample.

- **Connection String**: Displays the connection string used to access the Data Source.
- **Browse** button: Opens the **Edit Database configuration** dialog, which can replace the existing database data source with a new one. This is the same as using the **Replace** feature in the Data Samples window.
- **Table**: Displays the tables and stored procedures available in the database. The selected table is the one the data is extracted from. Clicking on any of the tables shows the first line of the data in that table.
- **Custom SQL** button: Click to open the SQL Query Designer (see"SQL Query Designer" on page 485) and type in a custom SQL query. If the database supports stored procedures, including inner joins, grouping and sorting, you can use custom SQL to make a selection from the database, using whatever language the database supports.
  The query may contain variables and properties, so that the selection will be dynamically adjusted each time the data mapping configuration is actually used in a Workflow process; see"Using variables and properties in an SQL query" on page 486.
- **Encoding**: Defines what encoding is used to read the Data Source (US-ASCII, ISO-8859-1, UTF-8, UTF-16, UTF-16BE or UTF-16LE).
- **Skip empty lines**: Ignore any row that has no content, e.g. only nulls or empty strings. Note that spaces are characters and thus considered content.
- **Sort on**: Allows to select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the sorting column has numbers, it will be sorted as a text.
  With a Custom Query, this option is not available.
- **Use previous JDBC-ODBC driver (not recommended)**: The JDBC-ODBC driver that was used in OL Connect versions before 2024.2 is deprecated and has been replaced with a new JDBC-ODBC driver. The new driver is used by default with both new and existing data mapping configurations. However, this option allows the old driver to be used if necessary. The functionality of the old driver remains unchanged.

Text file Input Data settings

Because text files have many different shapes and sizes, there are many options for the input data in these files.

- **Encoding**: Defines what encoding is used to read the Data Source (US-ASCII, ISO-8859-1, UTF-8, UTF-16, UTF-16BE or UTF-16LE).

- **Selection/Text is based on bytes**: Check for text files that use double-bytes characters (resolves width issues in some text files).
- **Add/Remove characters**: Defines the number of characters to add to, or remove from, the head of the data stream. The spin buttons can also increment or decrement the value. Positive values add blank characters while negative values remove characters.
- **Add/Remove lines**: Defines the number of lines to add to, or remove from, the head of the data stream. The spin buttons can also increment or decrement the value. Positive values add blank lines while negative values remove lines.
- **Maximum line length**: Any line that is longer than the given maximum line length will be split at the maximum line length, as often as necessary. This option is used to cut (and wrap) long lines into logical blocks of data.
  The maximum value for this option is 65,535 characters. The default value is 80 characters.
- **Page delimiter type**: Defines the delimiter between each page of data. Multiples of such pages can be part of a record, as defined by the **Boundaries**.
  - **On lines**: Triggers a new page in the Data Sample after a number of lines.
    - **Cut on number of lines**: Triggers a new page after the given number of lines. With this number set to 1, and the Boundaries set to On delimiter, it is possible to create a record for each and every line in the file.
    - **Cut on FF**: Triggers a new page after a Form Feed character.
  - **On text**: Triggers a new page in the Data Sample when a specific string is found in a certain location.
    - **Word to find**: Compares the text value with the value in the data source.
    - **Match case**: Activates a case sensitive text comparison.
    - **Location**: Choose **Selected area** or **Entire width** to use the value of the current data selection as the text value.
    - **Left/Right**: Use the spin buttons to set the start and stop columns to the current data selection (**Selected area**) in the record.
    - **Lines before/after**: This option places the delimiter a certain number of lines before or after the current line. This is useful if the text that triggers the delimiter is not on the first line of each page.
- **Text from right to left**: Sets the writing direction of the data source to right-to-left.
- **Expand tabs to spaces**: Replaces tabs with the given number of spaces.

- **Ignore CR/LF/FF at end of file**: Instructs the DataMapper to ignore any CR, LF, FF or CR/LF characters when they are the last characters in a file. This prevents the addition of an unintended trailing record when the data mapping configuration is set up to cut on every line.

XML File Input Data settings

For an XML file you can either choose to use the root node, or select an element type, to create a new record every time that element is encountered.

Note that the data file is not split. The XML tree is scanned from top to bottom (node by node), and when an element matches the record definition condition, the parsing mechanism stops. All parent or sibling nodes found before that element are displayed as part of the record. Any nodes located after the record definition node are not displayed (unless the Show all elements option is enabled). Parsing then starts again, at the top, and looks for the next element that matches the record definition condition, ignoring elements for which a record has already been created.

- **Use root element**: Selects the top-level element. No other boundaries can be set. If there is only one top-level element, there will only be one record.

  This setting is not recommended for XML data files is larger than 5 - 10MB. For files this large, it would be better to implement a Pre-processor script to pre-process the XML data file, or even convert it into CSV format.

- **Use specific element**: Displays a list containing all the elements in the XML file. Selecting an element causes a new page of data to be created every time an instance of this element is encountered.
  Note that higher-level nodes above the selected element are shown, but those below the selected element are not displayed unless the **Show all elements** option is checked.

- **Use XPath**: Enter an XPath to create records based on the **node name** of elements. For example: `./*[starts-with(name(),'inv')]` will create a record for every element of which the name starts with 'inv'.
  Note that `starts-with()` is an XPath function. For an overview of XPath functions, see Mozilla: XPath Functions.
  The XPath may also contain JavaScript code. Note that since the XPath is a string, the return value of the JavaScript statement will be interpreted as a string.
  In order to use JavaScript:

- The XPath must start with **=**
- The entire JavaScript statement must be enclosed in curly brackets: **{...}**
- Any other curly brackets that are not part of the JavaScript code must be escaped with a backslash.
- Single line comments (//...) in the code are not supported.

Note: Currently, XPaths that select elements based on an attribute, attribute value, node value, node counter or node index are not supported.

- **Show all elements**: When the delimiter is set to a specific element or XPath, checking this option allows to extract information from higher-level nodes, including those that follow the element or path. This might slow down the processing, so if you don't need any information from the higher-level nodes that follow that specific element, it is recommended to leave this option unchecked.

  When this option is used in combination with a trigger element that is not repeated at the same node level (in other words, it doesn't have a sibling with the same name), the entire XML document will be shown for each record, except the trigger element, which will only be shown for the record that is currently selected in the Data Model pane. This could lead to a problem with some steps that use an XPath with absolute indexes, such as a location-based Extract step. Using a dynamic index in the XPath will fix the problem. For example, in the case of a location-based Extract step, switch to extracting the data via JavaScript (see "Expression-based field" on page 425); in the JavaScript expression, replace the index of the element's parent in the XPath with record.index.

Note: The information contained in all of the selected parent nodes will be copied for each instance of that node. For example, if a client node contains multiple invoice nodes, the information for the client node can be duplicated for each invoice.

The DataMapper only extracts elements for which at least one value or attribute value is defined in the file.

JSON File Input Data settings

For a JSON file you can either use the object or array at the root and get one output record, or select an object or array as **parent element**. Its direct child elements - objects and arrays, *not* key-value pairs - can be output as individual records.

- **Use root element**: Selects the top-level array or object. There will only be one record.
- **Use specific element**: Select an array **[ ]** or object **{ }** in the JSON data as **Parent element** to define its child elements - objects and/or arrays - as source records. Any elements outside the parent element and key-value pairs inside the parent will be repeated in each source record.

  Note: Only arrays and objects can be seen as a record. It is not possible to split the JSON between key-value pairs.

Boundaries settings

Boundaries are the division between **records**: they define where one record ends and the next record begins; for an explanation see "Record boundaries" on page 373.

- "CSV, Excel or Database file boundaries" below
- "PDF file boundaries" on the next page
- "Text file boundaries" on page 384
- "XML file boundaries" on page 386
- "JSON file boundaries" on page 386

CSV, Excel or Database file boundaries

Since database data sources are structured the same way as CSV and Excel files, the options for these file types are identical.

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Line limit**: Defines the limit of detail lines in any detail table. This is useful for files with a high number of detail lines, which in the DataMapper interface can slow down things. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
  - **Record(s) per page**: Defines a fixed number of lines in the file that go in each record.
    - **Records**: The number of records (lines, rows) to put in each record.

- **On change**: Defines a new record when a specific field (**Field name**) has a new value.
    - **Field name**: Displays the fields in the top line. The boundaries are set on the selected field name.
- **On script:** Defines the boundaries using a custom JavaScript. For more information see "Setting boundaries using JavaScript" on page 535.
- **On field value**: Sets a boundary on a specific field value.
    - **Field name**: Displays the fields in the top line. The value of the selected field is compared with the **Expression** below to create a new boundary.
    - **Expression**: Enter the value or **Regular Expression** to compare the field value to.
    - **Use Regular Expression**: Treats the **Expression** as a regular expression instead of static text. For more information on using **Regular Expressions** (regex), see theRegular-Expressions.info Tutorial.

PDF file boundaries

For a PDF file, Boundaries determine how many pages are included in each record. You can set this up in one of three ways: by giving a static number of pages; by checking a specific area on each page for text changes, specific text, or the absence of text; or by using an advanced script.

- **Record limit**: Defines how many records are displayed in the Data Viewer. To disable the limit, use the value 0 (zero).
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
    - **On page**: Defines a boundary on a static number of pages.
        - **Number of pages**: Defines how many pages go in each record.
    - **On text**: Defines a boundary on a specific text comparison.
        - **Start coordinates (x,y)**: Defines the left and top coordinates of the data selection to compare with the text value.
        - **Stop coordinates (x,y)**: Defines the right and bottom coordinates.
        - **Use Selection**: Select an area in the Data Viewer and click the **Use selection** button to set the start and stop coordinates to the current data selection.

Note: In a PDF file, all coordinates are in millimeters.

- **Times condition found**: When the boundaries are based on the presence of specific text, you can specify after how many instances of this text the boundary can be effectively defined. For example, if a string is always found on the first and on the last page of a document, you could specify a number of occurrences of 2. This way, there is no need to inspect other items for whether it is on the first page or the last page. Having found the string two times is enough to set the boundary.
- **Pages before/after**: Defines the boundary a certain number of pages before (-) or after (+) the current page. This is useful if the text triggering the boundary is not located on the first page of the record.
- **Operator**: Selects the type of comparison (for example, "contains").
- **Word to find**: Compares the text value with the value in the data source.
- **Match case**: Makes the text comparison case sensitive.

- **On script:** Defines the boundaries using a custom JavaScript. For more information see "Setting boundaries using JavaScript" on page 535.
- **On all pages**: Sets a boundary after the last page, creating one source record.

Text file boundaries

For a text file, Boundaries determine how many 'data pages' are included in each record. These don't have to be actual pages, as is the case with PDF files. The data page delimiters are set in the "Text file boundaries" above.

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Selection/Text is based on bytes**: Select this option for text records with fixed width fields whose length is based on the number of bytes and not the number of characters.
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
    - **On delimiter:**Defines a boundary on a static number of pages.
        - **Occurrences**: The number of times that the delimiter is encountered before fixing the boundary. For example, if you know that your documents always

have four pages delimited by the FF character, you can set the boundaries after every four delimiters.

- **On text**: Defines a boundary on a specific text comparison.
  - **Location**:
    - **Selected area**:
      - **Select the area**button: Uses the value of the current data selection as the text value. Making a new selection and clicking on Select the area will redefine the location.
      - **Left/Right**: Defines where to find the text value in the row.
      - **Top/Bottom**: Defines the start and end row of the data selection to compare with the text value.
    - **Entire width**: Ignores the column values and compares using the whole line.
    - **Entire height**: Ignores the row values and compares using the whole column.
    - **Entire page**: Compares the text value on the whole page. Only available with`contains`,`not contains`,`is empty`and`is not empty`operators.
  - **Times condition found**: When the boundaries are based on the presence of specific text, you can specify after how many instances of this text the boundary can be effectively defined. For example, if a string is always found on the first and on the last page of a document, you could specify a number of occurrences of 2. This way, there is no need to inspect other items for whether it is on the first page or the last page. Having found the string two times is enough to set the boundary.
  - **Delimiters before/after**: Defines the boundary a certain number of data pages before or after the current data page. This is useful if the text triggering the boundary is not located on the first data page of the record.
  - **Operator**: Selects the type of comparison (for example, "contains").
  - **Word to find**: Compares the text value with the value in the data source.
  - **Use selected text** button: copies the text in the current selection as the one to compare to it.
  - **Match case**: Makes the text comparison case sensitive.

- **On script:** Defines the boundaries using a custom JavaScript. For more information see "Setting boundaries using JavaScript" on page 535.

XML file boundaries

The delimiter for an XML file is a node. The Boundaries determine how many of those nodes go in one record. This can be a specific number, or a variable number if the boundary is to be set when the content of a specific field or attribute within a node changes (for example when the invoice_number field changes in the invoice node).

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
  - **On Element**: Defines a new record on each new instance of the XML element selected in the Input Data settings.
    - **Occurrences**: The number of times that the element is encountered, as a direct descendant of the same parent element, before the boundary is set.
  - **On Change**: Defines a new record when a specific field or attribute in the XML element has a new value.
    - **Field**: Displays the fields and (optionally) attributes in the XML element. The value of the selected field determines the new boundaries.
    - **Also extract element attributes**: Check this option to include attribute values in the list of content items that can be used to trigger a boundary.

JSON file boundaries

The delimiter for a JSON file is an object or array inside the selected parent element (see "JSON file boundaries" above). The Boundaries determine how many of them go in one record.

Note: Only arrays and objects can be seen as a record. It is not possible to split the JSON between key-value pairs.

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero). The default value is 200.

- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
    - **On element**: Creates a new record in the output for each object or array, or - if you set a higher number of occurrences - after every *n*-th object or array in the parent element.
        - **Occurrences**: The number of times that an element is encountered in the parent element before fixing the boundary.
    - **On change**: Creates a new record each time the value in a certain key-value pair changes.
        - **Field**: Displays the keys of key-value pairs that exist at the root of direct child elements of the selected parent element. The value of the selected field determines the new boundaries.

Properties and runtime parameters

A data mapping configuration's **properties** hold data that can be used throughout the data mapping workflow to compare against in conditions or to complement the existing data.

A number of properties are predefined, such as the path of the file that is currently being processed, and the current position of the pointer in the data.
It is also possible to define custom properties. For instance, a unique ID could be created to be added to each record in the output for integrity checks later on, or a time stamp could be added to create reports. A tag could be added to process certain records differently.

Ultimately data mapping configurations are meant to be used in automated processes, to extract data from a particular type of data file.
Runtime parameters can pass values from an automation tool - OL Connect Workflow, usually - to a template, data mapping configuration or Job Preset. The actual values of the parameters may be different from one time that a process runs to the next, which means the output could be different even when the same template, data mapping configuration and/or Job Presets are used.

In a data mapping configuration, certain runtime parameters are predefined, such as the ones that refer to the JobInfo variables in Workflow. For an overview see "Fixed automation properties" on page 489.

You can also define custom runtime parameters. With those, you could, for example:

- Pass a JSON structure, containing boundary information as well as the name and location of fields to be extracted dynamically from a data stream. This way, a generic

data mapping configuration could be used to process several kinds of different files.

- Define default values that are different from one data mapping instance to the next, even when using the same data mapping configuration.

Defining custom properties and runtime parameters

Defining properties

You can define **custom properties** under **Properties** in the "Preprocessor step" on page 410 (see "Preprocessor step properties" on page 489).

To add a property:

1. Select the **Preprocessor** step on the **Steps** pane.
2. On the **Step properties** pane, under **Properties**, click the **Add** button. See "Properties" on page 490 for an explanation of the settings for properties.

Defining runtime parameters

Custom **runtime parameters** in a data mapping configuration are defined via the **Parameters** pane. This can be found next to the Data Model pane at the right. If it isn't visible, use the menu: **Window > Show View > Parameters** to make it visible.

> **Note:** The **source** of a runtime parameter value must be set in the automation tool, e.g. OL Connect Workflow, in the Runtime Parameters section of the OL Connect task that uses the data mapping configuration. (See All In One and Execute Data Mapping in the Online Help of Workflow.)

To add a runtime parameter, start by opening the data mapping configuration to which the parameter should be added. Make sure the data mapping configuration is visible in the workspace; then open the Parameters pane.

> **Note:** Runtime parameters are always added to the file currently visible in the workspace.

To add a parameter:

1. Click the **Add** button. The Add Parameter dialog opens.
2. Give the runtime parameter a **name**. The name is needed to access the parameter in a script and to set its source in the automation tool.

3. Optionally, set a **debug** value. Note that the debug value is never actually used outside of the DataMapper. Its only purpose is to make it easier to design and test the data mapping configuration. The actual value of a runtime parameter comes from the automation tool, e.g. the Execute Data Mapping task or the All in One task in OL Connect Workflow.
4. Click **OK**.

Editing a runtime parameter

To modify a runtime parameter, click its name or value in the Parameters pane and enter the new name or value.

To remove a runtime parameter, select it and click the **Remove** button ().

Accessing properties and runtime parameters

There are different ways to access properties and runtime parameters in a data mapping workflow.

- **Property-based fields**. A property-based field is filled with the value of a property. See "Property-based field" on page 427.
- **Step settings**. Properties can be selected in the **Condition**, **Multiple Conditions** and **Repeat** step properties (see "Condition step" on page 414, "Multiple Conditions step" on page 416 and "Repeat step" on page 412).
- **Script**. Scripts can access properties through the "DataMapper Scripts API" on page 531. (See also: "Using scripts in the DataMapper" on page 533.)
    - The `sourceRecord` object has a `properties` array that lets you access **custom properties** that are added to each individual record, i.e. the scope of the property is set to **Each record**. (See: "sourceRecord" on page 568.)
      To change the value of such a property you could use an **Action** step (see "Action step" on page 417). Note that the property's value will be reset at the beginning of each source record.
    - The `data` object has a `properties` array that lets you access **custom properties** of the data as a whole, i.e. the scope of the property is set to **Entire data**. (See: "data" on page 546.) These are read-only.
    - To access a **runtime parameter** inside of any JavaScript code within the data mapping configuration, use `automation.parameters.runtimeparametername`. (See: "Objects" on page 540.) Runtime parameters are read-only.

Note: Runtime parameters in a data mapping configuration are always of the type String. Remember to parse the value if necessary, e.g. before using it in a calculation.

Extracting data

Data are extracted via Extraction steps into fields in the Data Model. This topic explains how to do that.

Fields can also be filled with other data: the result of a JavaScript or the value of a property. To learn how to do that, see "Fields" on page 425.

Before you start

Data source settings

Data source settings must be made beforehand, not only to make sure that the data is properly read but also to have it organized in a record structure that meets the purpose of the data mapping configuration (see "Data source settings" on page 370). It is important to set the **boundaries** before starting to extract data, especially transactional data (see "Extracting transactional data" on page 396). Boundaries determine which data blocks - lines, pages, nodes - form a record in the source data. Data that are located in different records cannot be merged into a single record inside the record set that is the result of the extraction workflow.

Preprocessor step

The Preprocessor step allows the application to perform actions on the data file itself before it is handed over to the Data Mapping workflow. In addition, properties can be defined in this step. These properties may be used throughout the extraction workflow. For more information, see "Preprocessor step" on page 410.

Adding an extraction

In an extraction workflow, Extract steps are the pieces that take care of the actual data extractions.
To add an Extract step:

1. In the Data Viewer pane, select the data that needs to be extracted. (See "Selecting data" on page 394.)
2. Choose one of two ways to extract the selected data.

- Right-click on the selected data and select **Add Extraction** from the contextual menu.

  Note: For optimization purposes, it is better to add data to an existing Extract step than to have a succession of extraction steps. To do that, select that step on the Steps pane first; then right-click on the selected data and choose **Add Extract Field**.

- Alternatively, **drag & drop** the selected fields into the Data Model pane.

  Tip: In a PDF or Text file, use the Drag icon to drag selected data into the Data Model.

  With this method, a new Extract step will only be added to the extraction workflow if another Extract step is not currently selected. Otherwise, the field is added to the currently selected Extraction step.
  Dragging data into an **existing field** in the Data Model will replace the data. The field name stays the same.
  Drop data on empty fields or on the **record** itself to add new fields.

Special conditions

The Extract step may need to be combined with another type of step to get the desired result.

- Data can be extracted **conditionally** with a Condition step or Multiple Conditions step; see "Condition step" on page 414 or "Multiple Conditions step" on page 416.
- Normally the same extraction workflow is automatically applied to all records in the source data. It is however possible to **skip records**, entirely or partially, or to **stop data mapping** using an Action step. Add an Action step in a branch under a Condition step or Multiple Conditions step (see "Action step" on page 417) and set the type of action to Stop processing record or Stop data mapping, according to what is needed (see "Action step properties" on page 499).
- To extract **transactional** data, the Extract step must be placed inside a Repeat step. See "Extracting transactional data" on page 396.

Note: Data cannot be extracted more than once in any record, unless the Extract steps are mutually exclusive. This is the case when they are located in different branches of a Condition step or Multiple Conditions step.

Inside a Detail table, multiple Extract steps may extract the same data but each of them will create a new child record in the Detail table.

If you tick the *Append values to current record* option when several steps are extracting the same field, the step will error out.

Extracting data into multiple fields

When you select multiple fields in a CSV or tabular data file and extract them simultaneously, they are put into different fields in the Data Model automatically.
In a PDF or Text file, when multiple lines are extracted at the same time, they are by default joined and put into one field in the Data Model. To split them and put the data into different fields:

1. Select the field in the Data Model that contains the extracted lines.
2. On the **Step properties** pane, under **Field Definition**, click the drop-down next to **Split** and select **Split lines**.

Adding fields to an existing Extract step

For optimization purposes, it is better to add fields to an existing Extract step than to have a succession of extraction steps.

To add fields to an existing Extract step:

1. In the Data Viewer pane, select the data that needs to be extracted. (See "Selecting data" on page 394.)
2. Select an **Extract** step on the **Steps** pane.
3. Right-click on the data and select **Add Extract Field**, or drag & drop the data on the Data Model.

When data are dropped on the Data Model, they are by default added to the currently selected Extract step.

Extracting metadata

The pages in PDF/VT and AFP (Enterprise edition only) files can be grouped on several levels. Additional information, such as TLEs - a TLE is a **T**agged **L**ogical **E**lement - in AFP files (Enterprise edition only), can be attached to each level in the structure. The structure and additional information are stored in the file's metadata.

When setting up a data mapping configuration for these type of files, a metadata level can be set to define a record.
Next, if you are using the wizard, you can select fields to be extracted from the metadata automatically.
It is also possible to extract information from the metadata without the wizard. Here's how to do that.

1. Go to the **Steps** pane.
2. Click the **Extract step** toolbar button. This adds an Extract step with one field. Alternatively you could select an existing Extract step and add a field to it.
3. On the **Extract step properties** pane, under **Field definition**, set the field's **Mode** to **Metadata**.
4. Select the **Level** of the metadata on which the information can be found.
5. Select the **Property** to extract.

Alternatively you could create a JavaScript extraction (see "Using scripts in the DataMapper" on page 533 and "extractMeta()" on page 554).

Note: NOP (No Operation) records in AFP files cannot be extracted.

Editing fields

After extracting some data, you may want to:

- Change the **names** of fields that are included in the extraction.
- Change the **order** in which fields are extracted.
- Set the **data type**, **data format** and **default value** of each field.
- Modify the extracted data through a **script**.
- Delete a field.

All this can be done via the Step properties pane (see "Extract step properties" on page 492), because the fields in the Data Model are seen as properties of an Extract step.
See also: "Fields" on page 425.

Testing the extraction workflow

The extraction workflow is always performed on the current record in the data source. When an error is encountered, the extraction workflow stops, and the data field on which the error occurred and all data fields related to subsequent steps will be greyed out. Click the **Messages** tab (next to the Step properties pane) to see any error messages.

If you want to test a data mapping workflow on **all** records that are displayed, and see how the steps perform, read Testing a data mapping workflow.

> **Note:** At design time, steps automatically error out after the timeout value has been reached that is set in the preferences. This does not apply to the Preprocessor and Postprocessor steps and when the boundaries are being set. See "Common DataMapper preferences" on page 1039.

Selecting data

In order to extract the data, it is necessary to first define the data to be extracted, by selecting it. How this is done depends on the data source type. The following paragraphs explain how to create and manipulate a data selection for each different type of data.

Data selections are used to extract promotional data ("Extracting data" on page 390), transactional data ("Extracting transactional data" on page 396) and to apply a condition to an extraction (Condition step). They can also be used to control the workflow through conditions and loops.

Right-clicking on a data selection displays a contextual menu with the actions that can be performed with that selection or the steps that can be added with it. That menu also displays the keyboard shortcuts.

Text or PDF file

To **select** data in a Text or PDF file, click on a starting point, keep the mouse button down, drag to the end of the data that needs to be selected and release the mouse button. The data selection can contain multiple lines.

To **resize** a data selection, click and hold one of the resize handles on the borders or corners, move them to the new size and release the mouse button.

To **move** the data selection, click and hold anywhere on the data selection, move it to its new desired location and release the mouse button.

> **Note:** In a Text or PDF file, when you move the selection rectangle directly after extracting data, you can use it to select data for the next extraction.
> However, moving the selection rectangle that appears after clicking on a field in the Data Model actually changes which data is extracted into that field.

CSV/XLS/XLSX file or database recordset

Tabular data is displayed in the Data Viewer in a table where multiple fields appear for each line and row in the original data.

To select data, click on a field, keep the mouse button down, drag to the last field that you want to select and release the mouse button. You cannot select multiple lines with tabular data.
Alternatively you can select fields just like files in the Windows Explorer: keep the Ctrl button pressed down while clicking on fields to select or deselect them, or keep the Shift button pressed down to select consecutive fields.

XML file

XML data is displayed as a tree view inside the Data Viewer.

In this tree view you can select nodes just like files in the Windows Explorer: keep the Ctrl key pressed down while clicking on nodes to select or deselect them, or keep the Shift key pressed down to select consecutive nodes.
You can select multiple fields even if those fields are in different nodes.

To get a better overview you can collapse any XML level.

XMLPath

The Goto step is rarely used in XML extraction workflows. The DataMapper moves through the file using **Xpath**, a standard query language to identify and navigate nodes in an XML document.

JSON file

JSON data is displayed as a tree view inside the Data Viewer. Arrays and objects are enclosed in square or curly brackets, respectively: [ ] or { }.

In this tree view you can select elements just like files in the Windows Explorer. Keep the Ctrl key pressed down while clicking on key-value pairs or brackets to select multiple elements, or keep the Shift key pressed down to select consecutive elements.
You can select multiple key-value pairs, arrays and objects even if those are in different elements.

To get a better overview you can collapse any JSON level.

JsonPath

The DataMapper moves through JSON files using **JsonPath**, a path-like syntax to identify and navigate elements in a JSON document. For an overview of the JsonPath syntax and functions supported, see https://github.com/json-path/jsonpath.
One example is given in a How-to: XML/JSON loops.

In the DataMapper the JsonPath can be absolute (start with **$** which is the root) or relative to the current position (start with **.** which is the current element).
Note, however, that with a relative JsonPath going up to a parent element is not possible.

JsonPath can be used in a Repeat step, Extract step and Condition step, often eliminating the need for a Goto step.

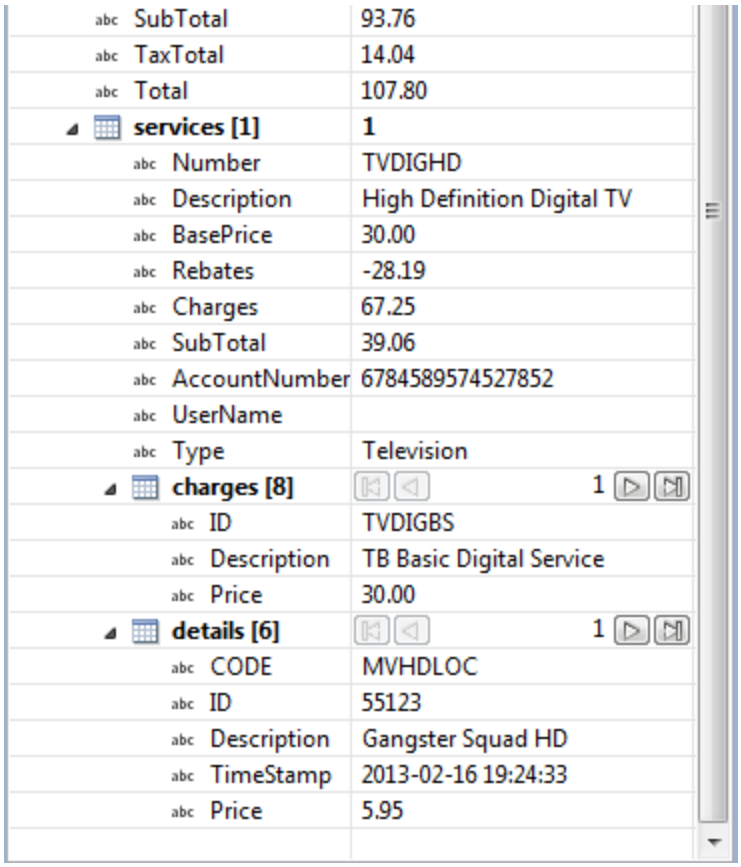
Tip: The full JsonPath to an element is displayed at the bottom left of the window when you select it. To copy the path, right-click it and select Copy.

Note: If a key in a JSON file has a name that looks like a function (e.g. "TLIST(A1)"), then the Extract step has to use a JsonPath with bracket notation instead of the default dot notation. For information about the bracket notation see https://goessner.net/articles/JsonPath/.

Extracting transactional data

Promotional data are data about customers, such as addresses, names and phone numbers. In Connect, each record in the extracted record set represents one recipient. The number of fields that contain promotional data is the same in each record. These data are stored on the root level of the extracted record.

Transactional data, on the other hand, are used in communications about transactions between a company and their customers or suppliers: invoices, statements, and purchase orders, for example. Naturally these data differ per customer. They are stored in **detail tables** in the extracted record. The number of detail lines in a detail table can vary from record to record.



(For more information about detail tables, **multiple** detail tables and **nested** detail tables, see "Detail tables" on page 453.)

Detail tables are created when an **Extract** step is added within a **Repeat** step. The Repeat step goes through a number of lines or nodes. An Extract step within that loop extracts data from each line or node.
How exactly this loop is constructed depends on the type of source data.

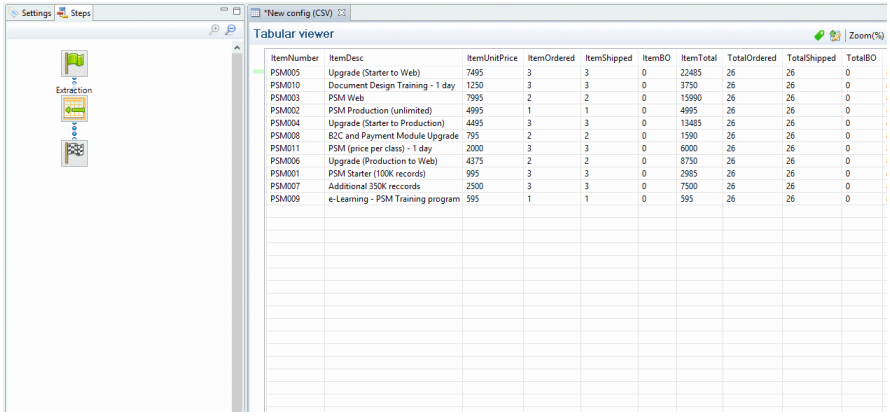
> **Tip:** To break out of a loop and immediately jump to the next task following the current loop, use an **Action** task and set its action to *Break out of repeat loop*.

Note that such loops work well when the data are structured uniformly, e.g. when all rows have the same number of columns, and when detail data always start at the same point in the line. Obviously, that is not always the case.
How to extract transactional data that is structured differently is explained in another topic: Extracting data with a script.
The topic: "Extracting data of variable length" on page 406 explains a few ways to extract data from a variable number of lines into one data field.

From a CSV file or a Database

The transactional data (also called line items) appear in multiple rows.

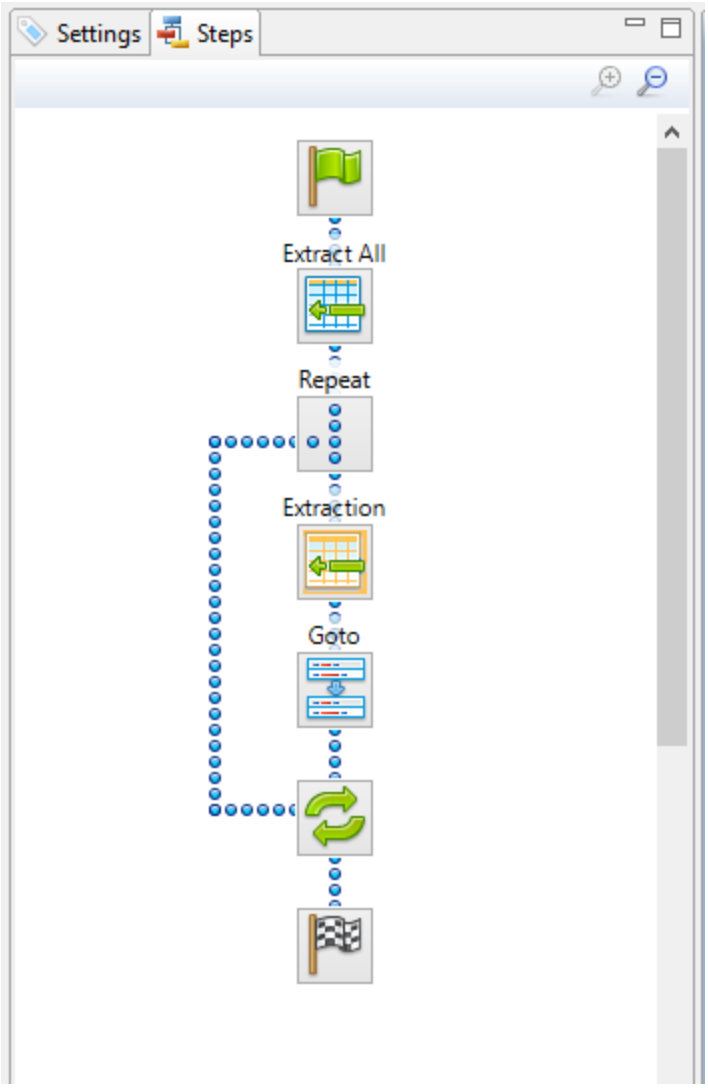
1. Select a field in the column that contains the first line item information.
2. Right-click this data selection and select **Add Repeat**.



ItemNumber	ItemDesc	ItemUnitPrice	ItemOrdered	ItemShipped	ItemBO	ItemTotal	TotalOrdered	TotalShipped	TotalBO
PSM005	Upgrade (Starter to Web)	7495	3	3	0	22485	26	26	0
PSM010	Document Design Training - 1 day	1250	3	3	0	3750	26	26	0
PSM003	PSM Web	7995	2	2	0	15990	26	26	0
PSM002	PSM Production (unlimited)	4995	1	1	0	4995	26	26	0
PSM004	Upgrade (Starter to Production)	4495	3	3	0	13485	26	26	0
PSM008	B2C and Payment Module Upgrade	795	2	2	0	1590	26	26	0
PSM011	PSM (price per class) - 1 day	2000	3	3	0	6000	26	26	0
PSM006	Upgrade (Production to Web)	4375	2	2	0	8750	26	26	0
PSM001	PSM Starter (100K records)	995	3	3	0	2985	26	26	0
PSM007	Additional 350K reccords	2500	3	3	0	7500	26	26	0
PSM009	e-Learning - PSM Training program	595	1	1	0	595	26	26	0

   This adds a Repeat step with a GoTo step inside it. The GoTo step moves the cursor down to the next line, until there are no more lines (see "Goto step" on page 413).
3. (Optional.) Add an empty detail table via the Data Model pane: right-click the Data Model and select **Add a table**. Give the detail table a name.
4. Select the Repeat step on the Steps pane.
5. Start extracting data (see "Adding an extraction" on page 390).
   When you drag & drop data on the name of a detail table in the Data Model pane, the data are added to that detail table.
   Dropping the data somewhere else on the Data Model pane creates a new detail table, with a default name that you can change later on (see "Renaming a detail table" on page 453).

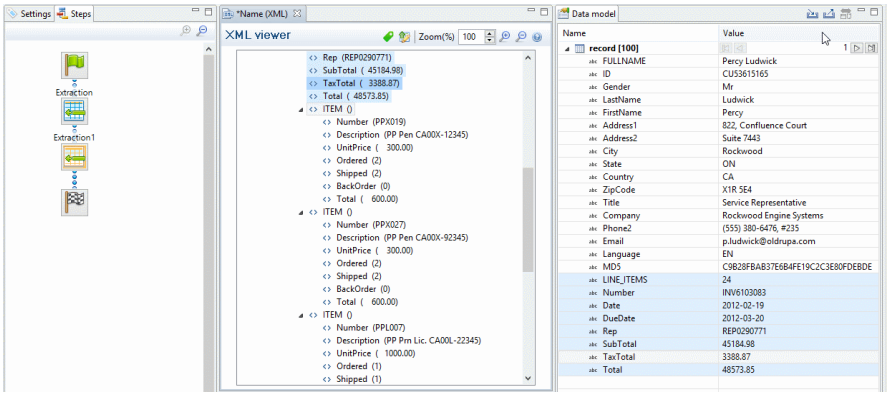
The extraction step is placed inside the Repeat step, just before the GoTo step.



From an XML file

The transactional data appears in repeated **elements**.

1. Right-click one of the repeating elements and select **Add Repeat**.



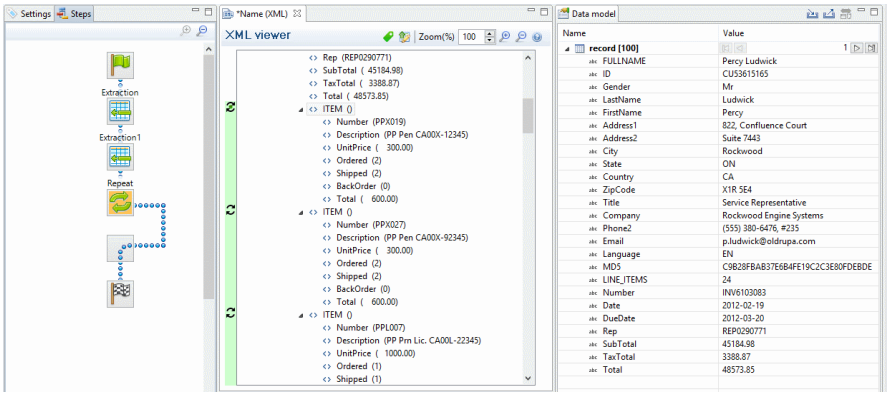
This adds a **Repeat step** to the data mapping configuration.
By default, the **Repeat type** of this step is set to **For Each**, so that each of the repeated elements is iterated over. You can see this on the **Step properties** pane, as long as the Repeat step is selected on the Steps pane. In the **Collection** field, you will find the corresponding node path.

> **Tip:** You may edit the XPath in the **Collection** field, to **include or exclude** elements from the loop.
> One example of this is given in a how-to: XML/JSON loops.
> For an overview of XPath functions, see Mozilla: XPath Functions.
> In addition, it is possible to use JavaScript statements in an XPath in the **Collection** field to dynamically select elements; see "Repeat Definition" on page 507.

The Goto step isn't used in XML extraction workflows in most cases. Instead, the cursor can be moved using **Xpath**, a path-like syntax to identify and navigate nodes in an XML document.

2. (Optional.) Add an empty detail table via the Data Model pane: right-click the Data Model and select Add a table. Give the detail table a name.

3. Select the Repeat step on the Steps pane.



4. Extract the data: inside a repeating element, select the data that you want to extract. Then right-click the selected nodes and select **Add Extraction**, or drag & drop them in the Data Model.
   When you drag & drop data on the name of a detail table in the Data Model pane, the data are added to that detail table.
   Dropping the data somewhere else on the Data Model pane creates a new detail table. By default, the table is named after the repeating element. You can change it later on (see "Renaming a detail table" on page 453).
   The new **Extract step** will be located in the Repeat step.

From a JSON file

The transactional data appears in repeated **elements**.

1. Move the cursor to the **parent element** of the repeating elements. By default the cursor is located at the top of the page, but previous steps may have moved it. Note that an Extract step does not move the cursor.

a. Select the parent element of the repeating elements.

b. Right-click and select **Add Goto**. The Goto step will move the cursor to the start of the first line item.

   Instead of using a Goto step you could define the path to the collection of elements directly in the JsonPath Collection field of the Repeat step. Note that this cannot be an absolute path. An absolute path points to one element only.

2. Right-click the opening bracket of the first of the repeating elements and select **Add Repeat**. This adds a **Repeat step** to the data mapping configuration.
   By default, the **Repeat type** of this step is set to **For Each**. With this setting the Extract step goes through the defined collection of elements, without the need for a Goto step

within the loop. (With the other Repeat type settings the loop must contain a Goto step.)
You can find the corresponding JsonPath in the **JsonPath Collection** field on the **Step properties** pane, as long as the Repeat step is selected on the Steps pane.

> **Tip:** You may edit the JsonPath in the **JsonPath Collection** field to **include or exclude** elements from the loop. For an overview of the JsonPath syntax, see https://github.com/json-path/jsonpath.

3. (Optional.) Add an empty detail table via the Data Model pane: right-click the Data Model and select Add a table. Give the detail table a name.
4. Select the Repeat step on the Steps pane.
5. Extract the data: inside the first of the repeating elements, select the data that you want to extract. Then right-click the selected nodes and select **Add Extraction**, or drag & drop in the Data Model.
   When you drag & drop data on the name of a detail table in the Data Model pane, the data are added to that detail table.
   Dropping the data somewhere else on the Data Model pane creates a new detail table. By default, the table is named after the repeating node. You can change it later on (see "Renaming a detail table" on page 453).
   The new **Extract step** will be located in the Repeat step.

About JsonPath

The DataMapper moves through JSON files using **JsonPath**, a path-like syntax to identify and navigate elements in a JSON document. For an overview of the JsonPath syntax and functions supported, see https://github.com/json-path/jsonpath.
One example is given in a How-to: XML/JSON loops.

In the DataMapper the JsonPath can be absolute (start with **$** which is the root) or relative to the current position (start with **.** which is the current element).
Note, however, that with a relative JsonPath going up to a parent element is not possible.

> **Tip:** The full JsonPath to an element is displayed at the bottom left of the window when you select it. To copy the path, right-click it and select Copy.

> **Note:** If a key in a JSON file has a name that looks like a function (e.g. "TLIST(A1)"), then the Extract step has to use a JsonPath with bracket notation instead of the default dot notation. For information about the bracket notation see https://goessner.net/articles/JsonPath/.

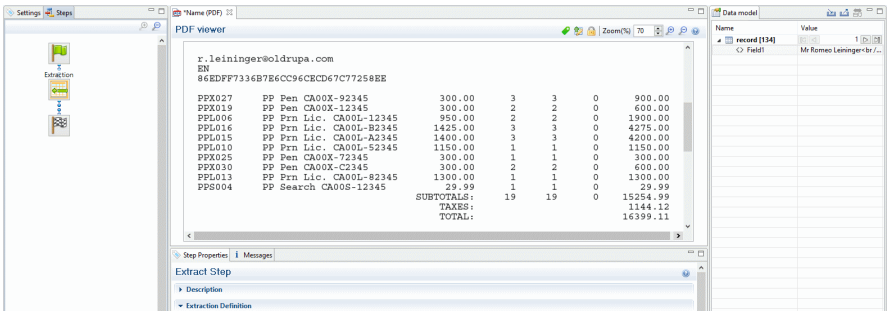
From a Text or a PDF file

In a PDF or Text file, transactional data appears on multiple lines and can be spread over multiple pages.

1. **Add a Goto step if necessary.** Make sure that the cursor is located where the extraction loop must start. By default the cursor is located at the top of the page, but previous steps may have moved it. Note that an Extract step does not move the cursor.

a. Select an element in the first line item.

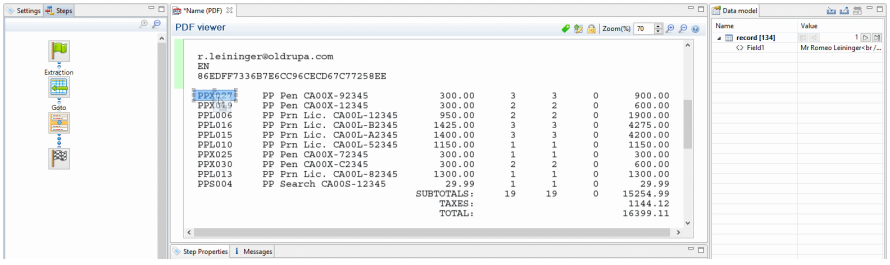
b. Right-click on the selection and select **Add Goto**. The Goto step will move the cursor to the start of the first line item.



2. Add a **Repeat** step where the loop must stop.

a. In the line under the last line item, look for a text that can be used as a condition to stop the loop, for example "Subtotals", Total" or "Amount".

b. Select that text, right-click on it and select **Add Repeat**. The Repeat step loops over all lines **until** the selected text is found.

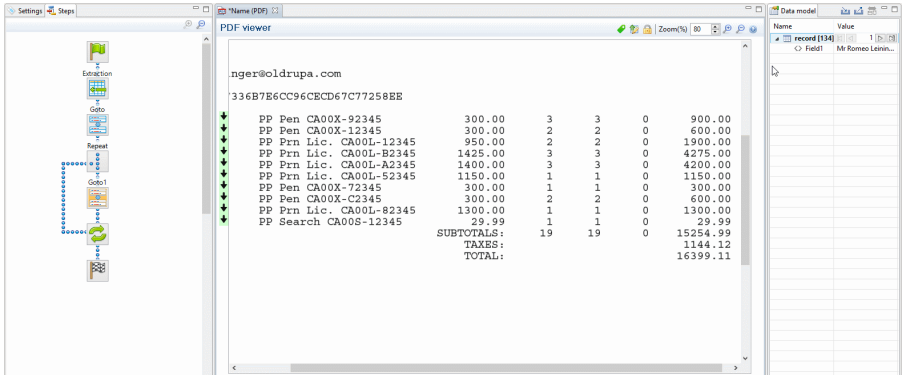


3. **Include/exclude lines.** Lines between the start and end of the loop that don't contain a line item must be excluded from the extraction. Or rather, all lines that contain a line item have to be included. This is done by adding a **Condition** step within the Repeat step.

a. Select the start of the Repeat step on the Steps pane.

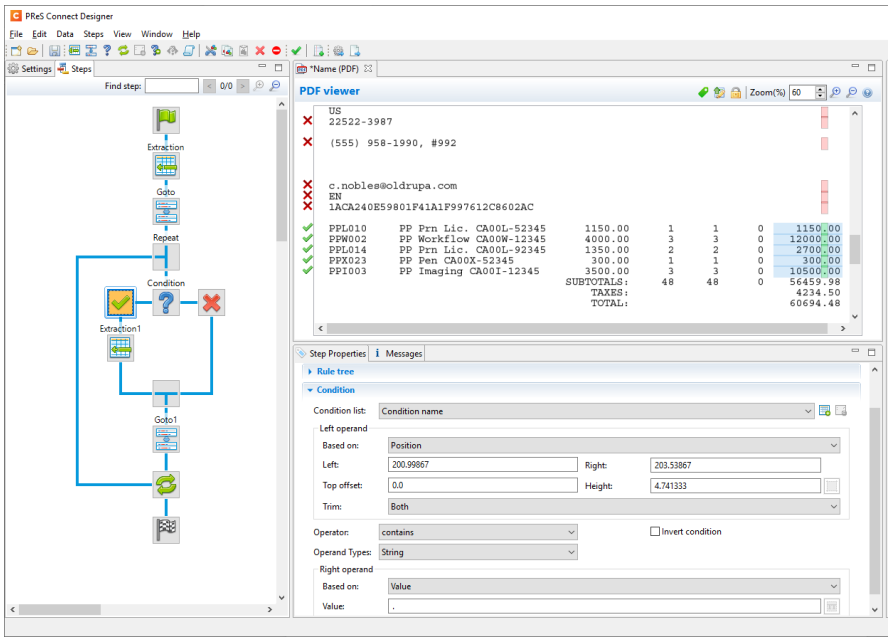
b. Look for something in the data that distinguishes lines with a line item from other lines (or the other way around). Often, a "." or "," appears in prices or totals at the same place in every line item, but not on other lines.

c. Select that data, right-click on it and select **Add Conditional**.



Selecting data - especially something as small as a dot - can be difficult in a PDF file. To make sure that a Condition step checks for certain data: Set the **Right operand** to **Value** (in the Step properties pane). Make a selection in the Data Viewer and click the **Use selected text** button in the **Right Operand** section. You will now be able to see whether or not the proper text is extracted by the current selection. Repeat this until you are satisfied that the proper data is being extracted. Click on the **Use selection** button in the **Left Operand** section to fill out the coordinates. The point of origin of each character is at the bottom left of each of them and extends up and to the right.

In the Data Viewer, you will see a green check mark in the left margin next to each included line and an X for other lines.



4. (Optional.) **Add an empty detail table** to the Data Model: right-click the Data Model and select **Add a table**. Give the detail table a name.

5. **Extract the data** (see "Adding an extraction" on page 390).
   When you drag & drop data on the name of a detail table in the Data Model pane, the data are added to that detail table.
   Dropping the data somewhere else on the Data Model pane, or using the contextual menu in the Data Viewer, creates a new detail table, with a default name that you can change later on (see "Renaming a detail table" on page 453).

   Note: In a PDF or Text file, pieces of data often have a variable size: a product description, for example, may be short and fit on one line, or be long and cover two lines. To learn how to handle this, see "Extracting data of variable length" on the facing page.

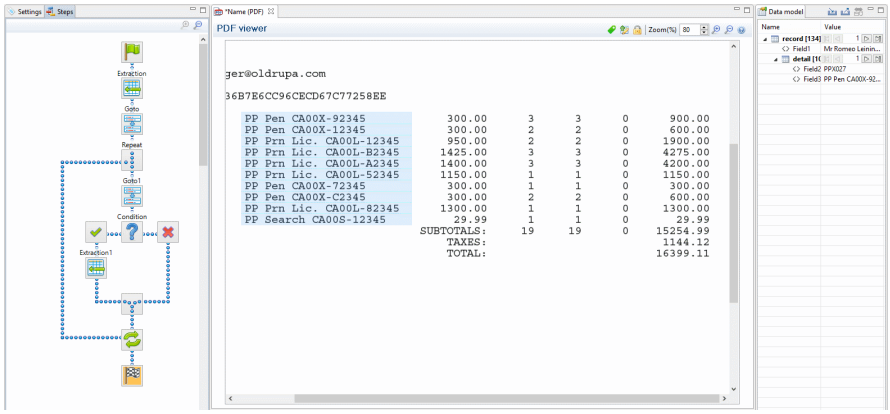
a6.Select the amount or amounts.

b. Click on the **end** of the Repeat step ( ) in the Steps panel.

c. Right-click on the selected data and select **Add Extraction**.

   Extract the sum or totals. If the record contains sums or totals at the end of the line items list, the end of the Repeat step is a good place to add an Extract step for these data. After the loop step, the cursor position is at the end of line items.Alternatively, right-click on the end of the Repeat step in the Steps panel and select **Add a Step >**

Add Extraction.



Extracting data of variable length

In PDF and Text files, transactional data isn't structured uniformly, as in a CSV, database or XML file. Data can be located **anywhere** on a page. Therefore, data are extracted from a certain **region** on the page. However, the data can be spread over **multiple** lines and multiple pages:

- Line items may continue on the next page, separated from the line items on the first page by a page break, a number of empty lines and a letterhead.
- Data may vary in length: a product description for example may or may not fit on one line.

How to exclude lines from an extraction is explained in another topic: "Extracting transactional data" on page 396 (see From a PDF or Text file).
This topic explains a few ways to extract a variable number of lines.

Text file: setting the height to 0

If the variable part in a TXT file is at the end of the record (for example, the body of an email) the height of the region to extract can be set to 0. This instructs the DataMapper to extract all lines starting from the current position in a record until the end of the record, and store them in a single field.

This also works with the data.extract() method in a script; see "extract()" on page 547.

Finding a condition

Where it isn't possible to use a setting to extract data of variable length, the key is to find one or more differences between lines that make clear how big the region is from where data needs to be extracted.
Whilst, for example, a product description may extend over two lines, other data - such as the unit price - will never be longer than one line. Either the area above or the one below the

unit price will be empty when the product description covers two lines.
Such a difference can then be used as a condition in a Condition step or a Case in a Multiple Conditions step.
A Condition step, as well as each Case in a Multiple Conditions step, can only check for one condition. To combine conditions, you would need a script.

Using a Condition step or Multiple Conditions step

Using a Condition step ("Condition step" on page 414) or a Multiple Conditions step ("Multiple Conditions step" on page 416) one could determine how big the region is that contains the data that needs to be extracted.
In each of the branches under the Condition or Multiple Conditions step, an Extract step could be added to extract the data from a particular region. The Extract steps could write their data to the same field.

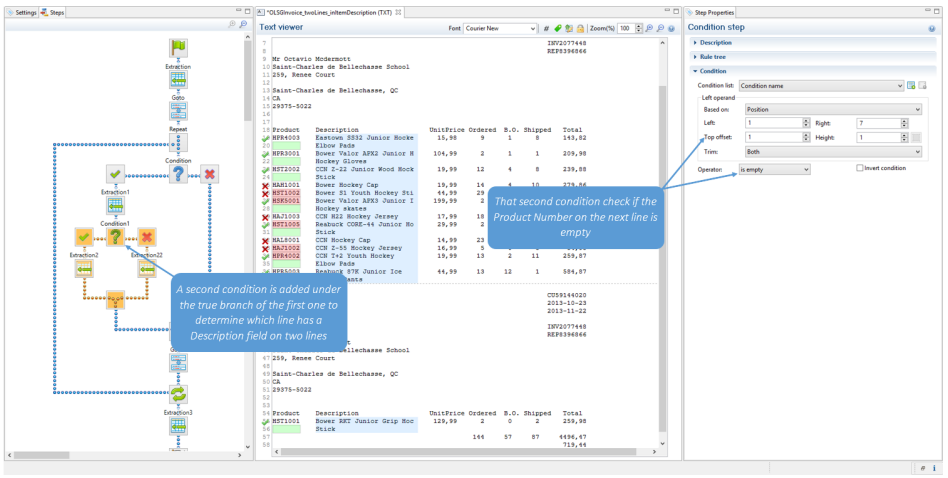
Note: Data cannot be extracted more than once in any record, unless the Extract steps are mutually exclusive. This is the case when they are located in different branches of a Condition step or Multiple Conditions step.

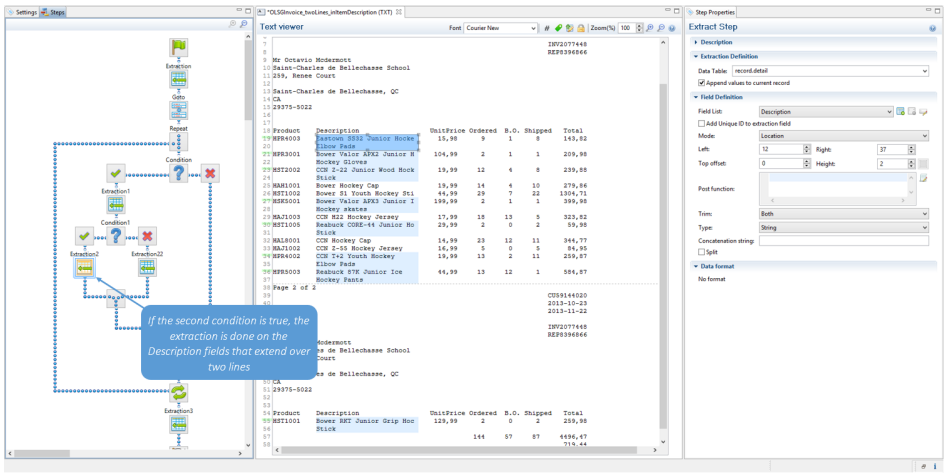
Inside a Detail table, multiple Extract steps may extract the same data but each of them will create a new child record in the Detail table.

If you tick the *Append values to current record* option when several steps are extracting the same field, the step will error out.

Tip: Create and edit the Extract step in the 'true' branch, then right-click the step on the Steps pane, select Copy Step, and paste the step in the 'false' branch. Now you only have to adjust the region from which this Extract step extracts data.

To learn how to configure a Condition step or a Case in a Multiple Conditions step, see "Configuring a Condition step" on page 415.





Using a script

A script could also provide a solution when data needs to be extracted from a variable region. This requires using a Javascript-based field.

1. Add a field to an Extract step, preferably by extracting data from one of the possible regions; see "Extracting data" on page 390. To add a field without extracting data, see "Expression-based field" on page 425.
2. On the Step properties pane, under Field Definition, select the field and change its **Mode** to **Javascript**.
   If the field was created with its Mode set to Location, you will see that the script already contains one line of code to extract data from the original location.
3. Expand the script. Start by doing the check(s) to determine where the data that needs to be extracted is located. Use the `data.extract()` function to extract the data. The parameters that this function expects depend on the data source, see "extract()" on page 547.

Example: The following script extracts data from a certain region in a Text file; let's assume that this region contains the unit price. If the unit price is empty (after trimming any spaces), the product description has to be extracted from two lines; else the product description should be extracted from one line.

```
var s = data.extract(1,7,1,2,"");
if (s.substring(1,3).trim().length == 0)
{ data.extract(12,37,0,2,""); } /* extract two lines */
else { data.extract(12,37,0,1,""); } /* extract one line */
```

The fourth parameter of the `extract()` function contains the height of the region. When working with a Text file, this equals a number of lines.

Tip: With a Text file, the `data.extract()` method accepts **0** as its height parameter. With the height set to 0 it extracts all lines starting from the given position until the end of the record.

Note that this script replicates exactly what can be done in a Condition step. In cases like this, it is recommended to use a Condition step. Only use a script when no steps are sufficient to give the expected result, or when the extraction can be better optimized in a script.

Steps

In the DataMapper, **steps** are part of an extraction workflow (see "Data mapping workflow" on page 368). They contain a specific instruction for the DataMapper, for example to extract data, create a loop, or apply a condition. Some types of steps are containers for other steps.

The steps, displayed on the "Steps pane" on page 486, are executed sequentially, from top to bottom in an extraction workflow.
Inside a Condition step, some steps may be skipped altogether when they are on a particular branch, whereas in a Repeat step - a loop - several steps may be repeated a number of times.

The Preprocessor and Postprocessor steps are special in that the former can be used to modify the incoming data prior to executing the rest of the extraction workflow while the latter can be used to further process the resulting record set after the entire extraction workflow has been executed.

Step types

These are the types of steps that can be added to a data mapping workflow:

- "Preprocessor step" below
- "Extract step" on the next page
- "Repeat step" on page 412
- "Goto step" on page 413
- "Condition step" on page 414
- "Multiple Conditions step" on page 416
- "Action step" on page 417
- "Postprocessor step" on page 418

Preprocessor step

The Preprocessor step allows to modify the incoming data through a number of **preprocessor actions** prior to executing the rest of the extraction workflow (see "Preprocessor actions" below). It also lets you define **properties** and **runtime parameters** that can be used throughout the data mapping workflow.

For a complete overview of the settings for a Preprocessor step, see: "Preprocessor step properties" on page 489.

Properties and runtime parameters

A data mapping configuration's **properties** hold data that can be used throughout the data mapping workflow to compare against in conditions or to complement the existing data. **Runtime parameters** are properties that will pass information from OL Connect Workflow to the data mapping configuration.

For more information see "Properties and runtime parameters" on page 387.

Preprocessor actions

The Preprocessor step can contain any number of preprocessor actions. They will be performed in sequence before the data is sent to the data mapping workflow.
A preprocessor action could remove certain records altogether.

One example of how a Preprocessor step could be used is given in a How-to: Using Preprocessors in DataMapper.

Note that preprocessor actions are not executed automatically while designing the data mapping workflow; you must therefore execute them manually. The reason for this is that preprocessor actions can potentially be quite lengthy operations that would hinder the automatic refresh of the display whenever anything is changed in the data mapping workflow.

To add a preprocessor action:

1. Select the **Preprocessor** step on the **Steps** pane.
2. On the **Step properties** pane, under **Actions**, click the **Add** button.
3. Under **Preprocessor definition**, add the script. Preprocessing tasks must be written in JavaScript (see "Using scripts in the DataMapper" on page 533 and "DataMapper Scripts API" on page 531).

Extract step

The **Extract** step is essential in each and every data mapping configuration. It extracts data from the data source, based on their location (a row and column in CSV or tabular data, an XPath in XML, a JsonPath in JSON, or a region of the page in PDF and Text) or on JavaScript code. The data is stored in the record set that is the result of the extraction workflow.

Fields always belong to an Extract step, but they don't necessarily all contain extracted data. To learn how to add fields without extracted data to an Extract step, see "Fields" on page 425.

Adding an Extract step

To add an Extract step, first select the step on the Steps pane after which to insert the Extract step. Then:

- In the Data Viewer, select some data, right-click that data and choose **Add Extraction**, or drag & drop the data in the Data Model. For more detailed information and instructions, see: "Extracting data" on page 390.
- Alternatively, right-click the Steps pane and select **Add a Step > Add Extraction**. Make the required settings on the Step properties pane.

If an **Extract** step is added within a **Repeat** step, the extracted data are added to a detail table by default; see "Extracting transactional data" on page 396 and "Detail tables" on page 453.

Configuring an Extract step

The names, order, data type and default value of the fields extracted in an Extract step are properties of that Extract step. These and other properties can be edited via the Step properties pane. For an explanation of all the options, see "Extract step properties" on page 492.

Note: Data cannot be extracted more than once in any record, unless the Extract steps are mutually exclusive. This is the case when they are located in different branches of a Condition step or Multiple Conditions step.

Inside a Detail table, multiple Extract steps may extract the same data but each of them will create a new child record in the Detail table.

If you tick the *Append values to current record* option when several steps are extracting the same field, the step will error out.

Repeat step

The **Repeat** step is a loop that may run 0 or more times, depending on the condition specified. It is generally used for the extraction of transactional data; see "Extracting transactional data" on page 396.

Repeat steps do not automatically move the pointer in the source file. Therefore, when adding a Repeat step, a **Goto** step that moves the cursor is inserted automatically inside the loop to avoid an infinite loop, except with XML and JSON data.
When you select a node in an XML file and add a Repeat step on it, the Repeat step will automatically loop over all nodes of the same type on the same level in the XML file.
The same applies to JSON files. When you select an element in a JSON file and add a Repeat step on it, the Repeat step will automatically loop over all elements on the same level in the JSON file.

Tip: To break out of a loop and immediately jump to the next task following the current loop, use an **Action** task and set its action to *Break out of repeat loop*.

Adding a Repeat step

To add a Repeat step:

1. On the Steps pane, select the step after which to insert the Condition step.
2. Make sure that the cursor is located where the extraction loop must start. By default the cursor is located at the top of the page or record, but previous steps in the extraction workflow may have moved it down. If necessary, add a **Goto** step (see "Goto step" on the next page).
   Generally this step can be skipped when the data source is an XML file or JSON file.

3. Add the Repeat step: select data in the line or row where the loop must end, or select the node/element. Then either:
    - Right-click on it and select **Add Repeat**.
    - Right-click the Steps pane and select **Add a Step > Add Repeat**.
    - Select **Steps > Add Repeat Step** from the menu.
4. Make the required settings on the Step properties pane. See: "Repeat step properties" on page 507.

For more detailed instructions per file type see "Extracting transactional data" on page 396.

Configuring a Repeat step

For information about how to configure the Repeat step, see "Repeat step properties" on page 507.

How to use it in an extraction workflow is explained in the topic: "Extracting transactional data" on page 396.

Goto step

Although invisible, there is a **cursor** in the Data Viewer that represents the location that the extraction workflow will be processing next in the current record. In an extraction workflow the cursor starts off at the top-left corner of each record in the source data.
The **Goto** step can move the cursor to a certain location in the current record. The target location can be relative to the top of the record or to the current position.

When the **Goto** step is used within a **Repeat** step, it moves the cursor in each loop of the Repeat step. In this case the new location has to be relative to the current position.

The Goto step isn't used in XML extraction workflows in most cases. Instead, the cursor can be moved using **Xpath**, a path-like syntax to identify and navigate nodes in an XML document.

The DataMapper moves through JSON files using **JsonPath**, a path-like syntax to identify and navigate elements in a JSON document. For an overview of the JsonPath syntax and functions supported, see https://github.com/json-path/jsonpath.
One example is given in a How-to: XML/JSON loops.

In the DataMapper the JsonPath can be absolute (start with **$** which is the root) or relative to the current position (start with **.** which is the current element).
Note, however, that with a relative JsonPath going up to a parent element is not possible.

In cases where an absolute JsonPath is necessary a Goto step might be needed.

Adding a Goto step

To add a Goto step:

- On the Steps pane, select the step after which to insert the Goto step. In the Data Viewer, select some data, right-click that data and choose **Add Goto**, to add a Goto step that moves the cursor to that data.
- Alternatively, right-click the Steps pane and select **Add a Step > Add Goto**. Make the required settings on the Step properties pane.

Configuring a Goto step

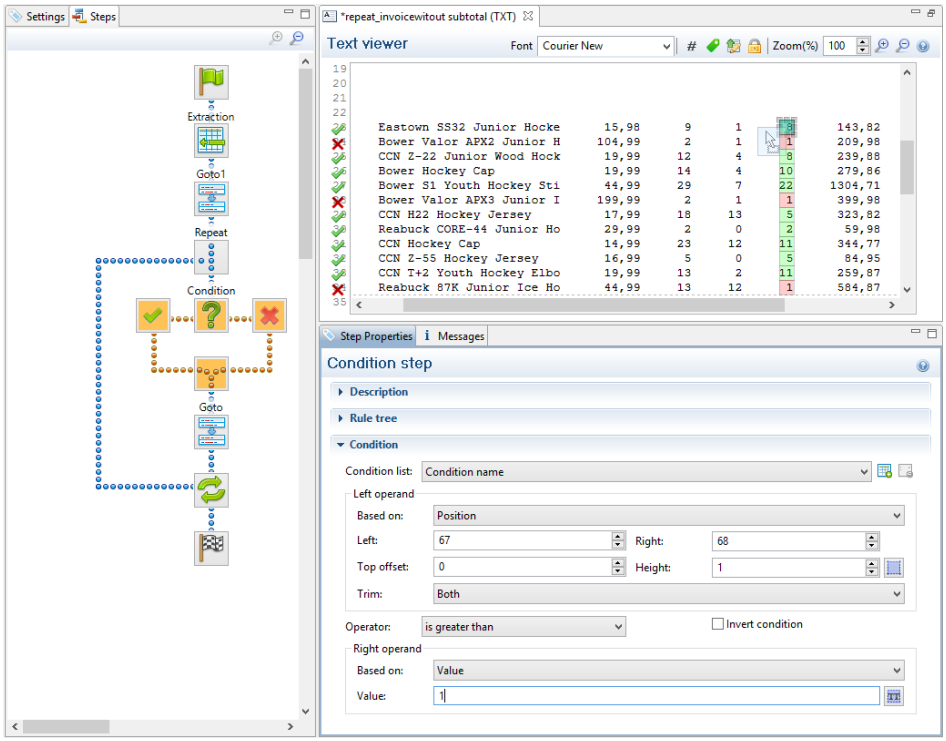
For information about how to configure the Goto step, see "Goto step properties" on page 521.

Condition step

A **Condition** step is used when the data extraction must be based on specific criteria. The Condition step splits the extraction workflow into two separate branches, one that is executed when the condition is **true**, the other when it is **false**.

Extract steps can be added to both the 'true' and the 'false' branch (see "Extracting data" on page 390 and "Extracting transactional data" on page 396).

In the Data Viewer pane, icons on the left indicate the result of the evaluation in the Condition step for each line: ✔ (check) when true and ✘ (cross) when false.



Adding a Condition step

To add a Condition step:

- On the Steps pane, select the step after which to insert the Condition step; then, in the Data Viewer, select some data, right-click that data and choose **Add Conditional**.
  In the **Step properties** pane, you will see that the newly added Condition step checks if the selected **position** (the left operand) **contains** the selected **value** (the right operand). Both operands and the operator can be adjusted.
  Note that the left operand is by default **trimmed**, meaning that spaces are cut off.
  Selecting data - especially something as small as a dot - can be difficult in a PDF file. To make sure that a Condition step checks for certain data: Set the **Right operand** to **Value** (in the Step properties pane). Make a selection in the Data Viewer and click the **Use selected text** button in the **Right Operand** section. You will now be able to see whether or not the proper text is extracted by the current selection. Repeat this until you are satisfied that the proper data is being extracted. Click on the **Use selection** button in the **Left Operand** section to fill out the coordinates. The point of origin of each character is at the bottom left of each of them and extends up and to the right.
- Alternatively, right-click the Steps pane and select **Add a Step > Add Conditional**.
  Enter the settings for the condition on the Step properties pane.

Configuring a Condition step

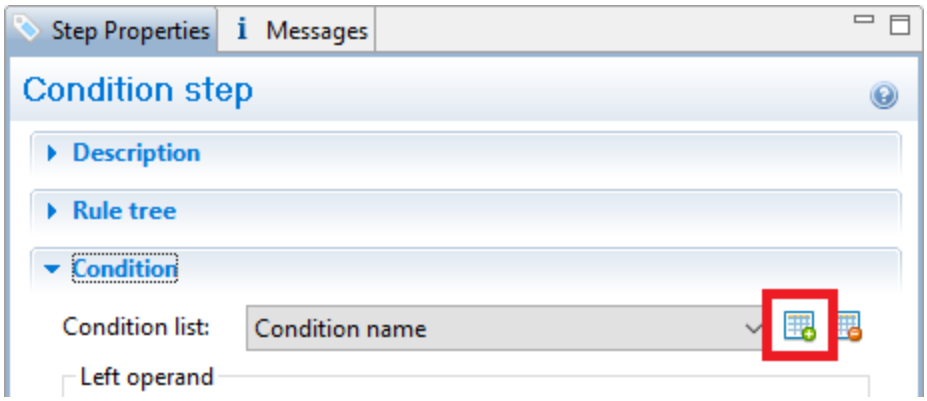
The condition in a Condition step is expressed in a rule or combination of rules. Rules have a left operand, an operand type (for example: contains, is empty) and a right operand.
For an overview of all options on the Step properties pane, see "Condition step properties" on page 515.

Inverting a rule

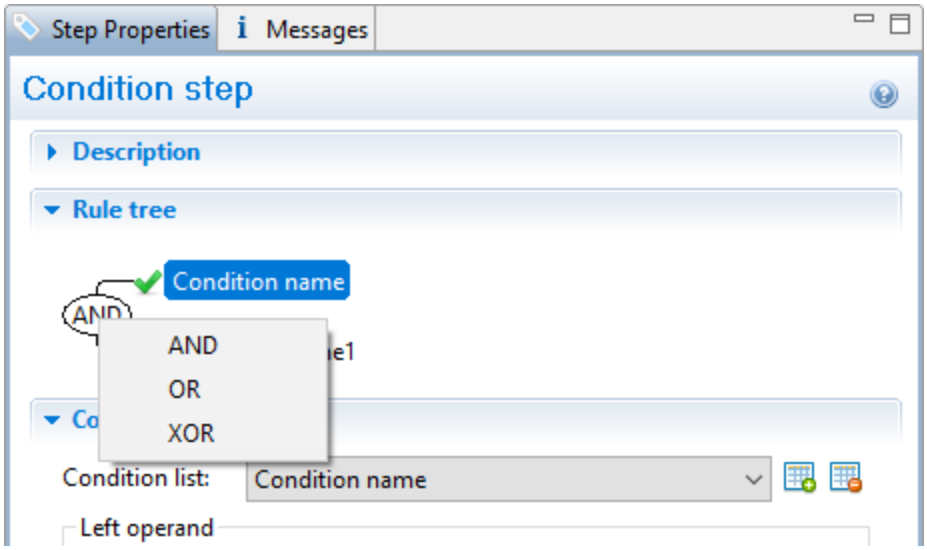
Inverting a rule adds **not** to the operand type. For instance, **is empty** becomes **is not empty**.
To invert a rule, check the **Invert condition** option next to Operator under Condition on the Step properties pane.

Combining rules

One rule is already present in a newly added Condition step. To add another rule, click the **Add condition** button under Condition, next to Condition List, on the Step properties pane.



Rules are by default combined with AND. To change the way rules are combined, right-click "AND" in the Rule Tree, on the Step properties pane, and select OR or XOR instead. (XOR means one or the other, but not both.)



Renaming a rule

To rename a rule, double-click its name in the Rule Tree and type a new name.

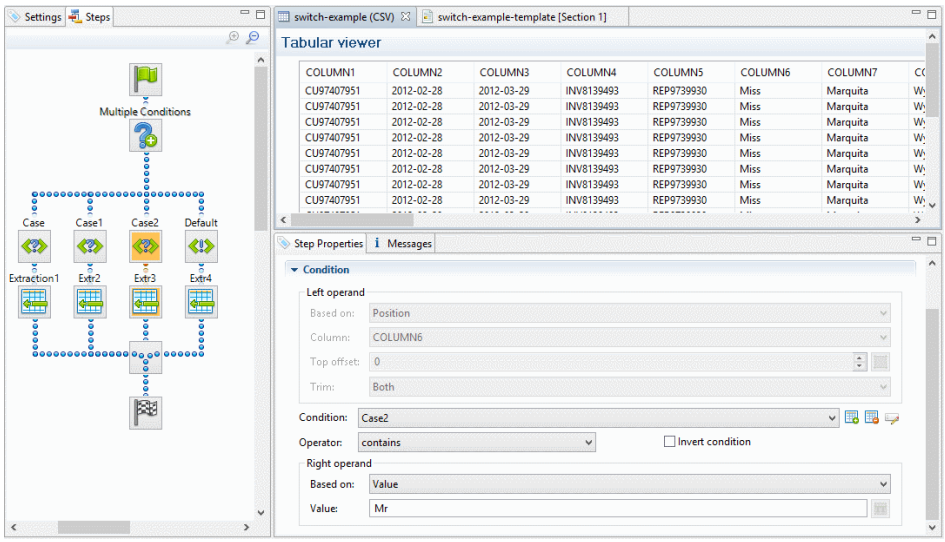
Multiple Conditions step

The **Multiple Conditions** step is useful to avoid the use of nested Condition steps (Condition steps inside other Condition steps).

In a **Multiple Conditions** step, conditions or rather **Cases** are positioned side by side.

Each **Case** condition can lead to an extraction.

Cases are executed from left to right.



Adding a Multiple Conditions step

To add a Multiple Conditions step, right-click the Steps pane and select **Add a Step > Add Multiple Conditions**.

To add a **case**, click the **Add case** button to the right of the **Condition** field in the Step properties pane.

Configuring a Multiple Conditions step

For information about how to configure the Multiple Conditions step, see "Multiple Conditions step properties" on page 518. The settings for a Case are the same as for a Condition step; see "Condition step properties" on page 515.

Action step

The **Action** step can:

- Execute JavaScript code.
- Break out of the loop of a Repeat step.
- Set the value for a record property. Record properties are defined in the Preprocessor step; see "Preprocessor step" on page 410.
- Stop the processing of the current record and move on to the next one. Normally an extraction workflow is automatically executed on all records in the source data. By stopping the processing of the current record, you can filter out some records or skip records partially. Note that if the extraction workflow has already extracted some data before it gets stopped, the record containing that partial data will still be stored in the Connect Database; to skip a record entirely, the Action step must occur before

any data is extracted for that record.

- Stop the data mapping. If fields of the current record were already extracted, then those fields are stored as usual, but the rest of the record and any following records are skipped.

The **Action** step can run multiple specific actions one after the other in order.

Adding an Action step

To add an Action step, right-click on the Steps pane and select **Add a Step > Add Action**.

Configuring an Action step

For information about how to configure the Action step, see "Action step properties" on page 499.

Postprocessor step

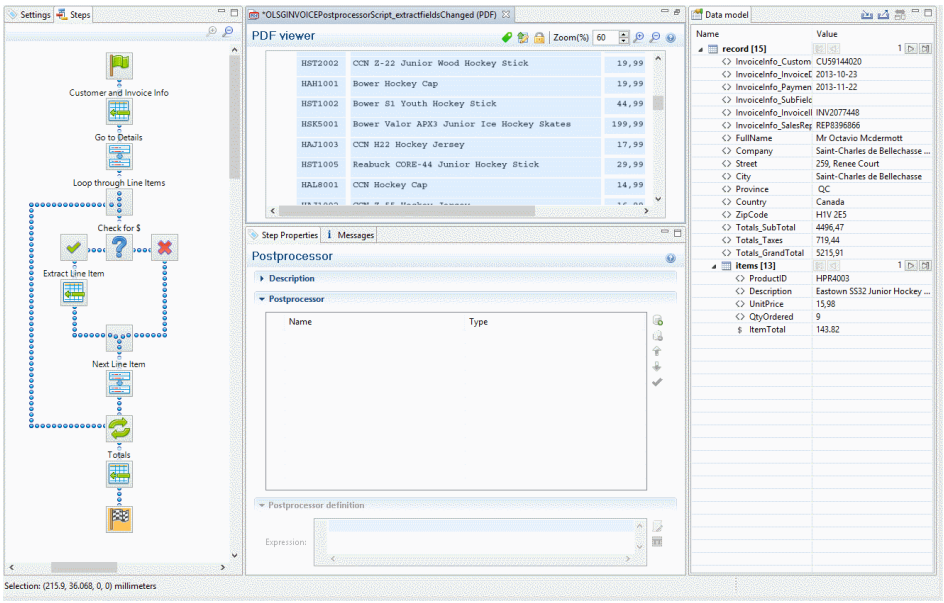
The Postprocessor step allows the application to further process the resulting record set after the entire extraction workflow has been executed, using JavaScript.
For example, a post-processor can export all or parts of the data to a CSV file which can then be used to generate daily reports of the Connect Workflow processes that use this data mapping configuration (see "Data mapping configurations" on page 337).
A post-processor could also write the results of the extraction process to a file and immediately upload that file to a Workflow process.

The Postprocessor step can contain any number of post-processors.

Adding a post-processor

To add a post-processor:

- Select the **Postprocessor** step on the **Steps** pane.
- On the **Step properties** pane, under **Postprocessor**, click the **Add** button.
- Under **Postprocessor definition**, add the script. Postprocessor tasks must be written in JavaScript (see "Using scripts in the DataMapper" on page 533 and "DataMapper Scripts API" on page 531).



Configuring the Postprocessor step

For an explanation of the settings for post-processors, see "Postprocessor step properties" on page 525.

Testing postprocessors

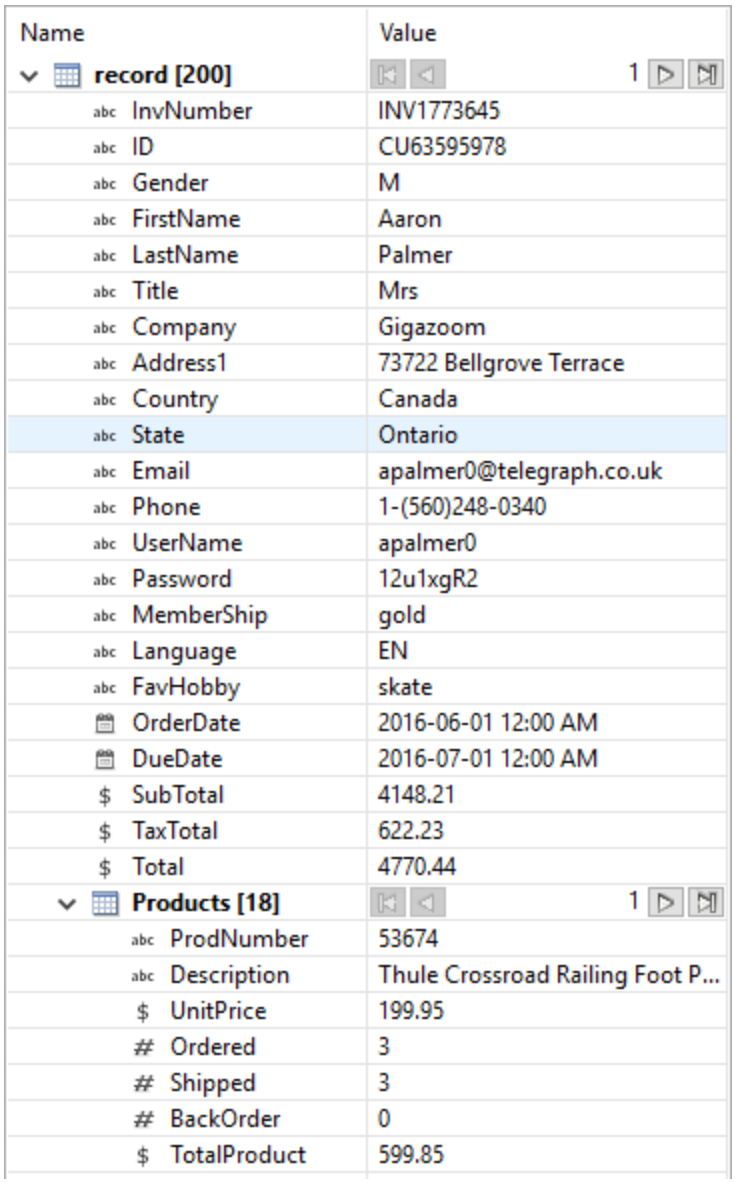
Post-processors are not executed automatically while designing the data mapping workflow. The reason for this is that post-processors can potentially be quite lengthy operations that would hinder the automatic refresh of the display whenever anything is changed in the data mapping workflow.
In order to test post-processors you must execute them manually by clicking the **Apply** button in the Post-processor step properties (see "Postprocessor step properties" on page 525).

> **Note:** In the DataMapper and Designer, only one data record is active at any given time. Therefore, the changes made by the post-processes are only visible on the current data record (i.e. the one currently displayed). If you change data records, you must press the **Apply** button on the Postprocessor step once more.
> In order to test a case with multiple records you must either use the data mapping configuration in a print job through the Designer or in the Execute Data Mapping task in Workflow so that all records will be processed by the Postprocessor step.

The Data Model

The Data Model is the structure of records into which extracted data are stored. It contains the names and types of the fields in a record and in its detail tables. A detail table is a field

that contains a record set instead of a single value. The Data Model is shown in the Data Model pane, filled with data from the current record.



Name	Value
record [200]	1
InvNumber	INV1773645
ID	CU63595978
Gender	M
FirstName	Aaron
LastName	Palmer
Title	Mrs
Company	Gigazoom
Address1	73722 Bellgrove Terrace
Country	Canada
State	Ontario
Email	apalmer0@telegraph.co.uk
Phone	1-(560)248-0340
UserName	apalmer0
Password	12u1xgR2
MemberShip	gold
Language	EN
FavHobby	skate
OrderDate	2016-06-01 12:00 AM
DueDate	2016-07-01 12:00 AM
SubTotal	4148.21
TaxTotal	622.23
Total	4770.44
Products [18]	1
ProdNumber	53674
Description	Thule Crossroad Railing Foot P...
UnitPrice	199.95
Ordered	3
Shipped	3
BackOrder	0
TotalProduct	599.85

The Data Model is not related to the type of data source: whether it is XML, JSON, CSV, PDF, Text, or a database does not matter. The Data Model is a new structure, designed to contain only the required data.

About records

A **record** is a block of information that may be merged with a template to generate a single *document* (invoice, email, web page...) for a single *recipient*. It is part of the *record set* that is generated by a data mapping configuration.

In each record, data from the data source can be combined with data coming from other sources.

Records can be duplicated by setting the number of **copies** in a script (see "record" on page 565). Duplicates are not shown in the Data Model.

Creating a Data Model

A Data Model is created automatically within each data mapping configuration, but it is empty at the start. To fill it you could use another Data Model (see "Importing/exporting a Data Model" below) or start creating a data mapping workflow (see "Data mapping workflow" on page 368).

To learn how to add and edit fields, see "Fields" on page 425.

Importing/exporting a Data Model

To use a Data Model in another data mapping configuration, or to use it in a Designer template without a data mapping configuration, you have to export that Data Model and import it into a data mapping configuration or template.

Importing and exporting Data Models is done from within the Data Model pane, using Import and Export icons in the View menu at the top right of the Data Model.

For information about the structure of the **exported** Data Model file, see "Data Model file structure" on page 446.

When you **import** a Data Model, it appears in the Data Model pane where you can see all the fields and their types.
Instead of a Data Model file, you can import a JSON file, or a Connect file that contains a Data Model. The file's data model structure will be displayed in the Data Model pane.

You can delete or add fields, or change their type. Once the Data Model is imported and all the fields are properly set, all you need to do is extract the information from the active data sample (see "Extracting data" on page 390).

Note:

- Imported Data Model fields always overwrite existing field properties when the field name is the same (although they will still be part of the same Extract step). Non-existent fields are created automatically with the appropriate field settings. The import is case-insensitive.

- All imported Data Model fields are marked as required in the Data Model (indicated with an asterisk (*) next to their names). This is to prevent them from being removed, as the DataMapper immediately discards non-required fields that are not referenced by any Extract step.

Editing the Data Model

The Data Model is generally constructed by extracting data; see "Extracting data" on page 390.

Empty fields and data tables can be added via the contextual menu; see "Adding empty fields via the Data Model pane" on page 427.

Editing fields

You can modify the fields in the Data Model via the contextual menu that opens when you right-click on something in the Data Model pane. For an explanation of the options in this menu, see "Data Model contextual menu" on page 450.

As long as a field or data table, added via the Data Model pane, remains **empty**, it can be edited (renamed, deleted etc.) via the contextual menu.
Fields in a Data Model that are actually used in the extraction workflow cannot be edited via the contextual menu of the Data Model pane. They are related to a step in the extraction workflow and are edited via the Step properties pane instead (see "Editing fields" on page 428).

Ordering and grouping fields in the Data Model

To make the Data Model clearer and more manageable, you can change the order of the fields in it and create groups of fields. All grouping and ordering information is saved in the data mapping configuration, so that when you reopen it, the Data Model will look the same.

Note that these changes have no impact on the order in which data are extracted, or on the final records in the OL Connect database. They only help you organize the Data Model **visually**.
This also means group names are not used in scripts; fields and detail tables are accessed directly through the record object (see the "Designer Script API" on page 1479 and "DataMapper Scripts API" on page 531).

To change the order in which data are extracted, see "Renaming and reordering fields in an extraction step" on page 428.

Moving fields

To move a field, a group of fields or a detail table, you can simply **drag and drop** it to some other place in the Data Model.

Alternatively, you can **right-click** the field, group or detail table and select one of the options in the **Move** menu, to move it up or down within the list or group.

Fields cannot be moved into another table. However, inside a table they can be moved up or down.

Grouping fields

To group one or more fields, select the field(s), right-click and select **Add group**. It is also possible to create groups within groups; this is done in the same way.
To move a field into an existing group you can simply **drag and drop** it into the group or onto the name of that group.

To delete a group, right-click it and select **Ungroup**. The fields will be moved up one level in the structure. To move a field out of an existing group you can also simply **drag and drop** it out of that group.
To delete an empty group, right-click it and select Delete.

Using the Data Model in templates

The Data Model is what enables you to create personalized templates in the **Designer** module.
You can drag & drop fields from the Data Model into the template that you are creating (see "Variable data in the text" on page 924). For this, you need to either have a template and a data mapping configuration open at the same time, or import a Data Model (see "Importing/exporting a Data Model" on page 421).

The Data Model is reusable, meaning that it can be shared amongst different template layouts and output types.
Different data mapping configurations could use the same Data Model, allowing a template to be populated with data from different sources and formats, without the need to modify the template (see "Importing/exporting a Data Model" on page 421).

In **Workflow**, when a data mapping configuration is used to extract data from a data source (see "Data mapping configurations" on page 337), the extracted data is stored in a record set in the OL Connect database.

Adding fields and data via Workflow

The Data Model is not extensible outside of the DataMapper. When it is used in Workflow - as part of a data mapping configuration - the contents of its fields can be updated but not its structure. There are a number of instances however, where data may need to be added to the data model after the initial data mapping operation has been performed. For instance, you might need to add a cleansed postal address next to the original address, or retrieve a value from a database and add it to the record.

Runtime parameters

Runtime parameters can pass values from an automation tool - OL Connect Workflow, usually - to a template, data mapping configuration or Job Preset. The actual values of the parameters may be different from one time that a process runs to the next, which means the output could be different even when the same template, data mapping configuration and/or Job Presets are used.

See "Properties and runtime parameters" on page 387 to learn how to use them in a data mapping configuration.

ExtraData fields

Before the introduction of runtime parameters, empty fields needed to be added in advance to provide space in the Data Model for Workflow to store data. For convenience, one field called **ExtraData** was (and is) automatically created at every level of each data record. That means the record itself gets an **ExtraData** field, and each detail table also gets one. By default the field is not visible in the DataMapper's Data Model, because it is not meant to be filled via an extraction. It can be made visible using the **Show ExtraData Fields** icon in the View menu at the top of the Data Model. This icon is only visible if the View ExtraData option has been checked in the preferences: **Window** > **Preferences** > **Data Model**.

Workflow process

ExtraData fields can be filled in a OL Connect **Workflow** process as follows:

1. Use an **Execute Data Mapping** task or **Retrieve Items** task to create a record set. On the General tab select **Outputs records in Metadata**.
2. Add a value to a field **in the Metadata** using the **Metadata Fields Management** task. Data added to the **_vger_fld_ExtraData** field on the Document level will appear in the record's ExtraData field, once the records are updated from the Metadata (in the next step).
   Other fields have the same prefix: **_vger_fld_**.

3. Update the **record/s** from the Metadata. There are several ways to do this. You could, for example:
    - Use the **Update Data Records** plugin.
    - Add an Output task and check the option **Update records with Metadata**.
    - Select Metadata as the data source in the **Create Preview PDF** plugin.

Note: Many of these actions can also be performed using REST calls.

Please refer to OL Connect Workflow documentation for more information about the plugins involved: Workflow's Online Help.

Fields

Extracted data are stored in fields in the Data Model (see "The Data Model" on page 419).
Fields can be present on different levels: on the record level or in a detail table (see "Detail tables" on page 453).
They can be moved up or down and put in groups (see below).

Fields with data belong to an Extract step and are accessible via the Step Properties pane (see "Extract step properties" on page 492).
Location-based fields always contain extracted data: they read data from a certain location in the data source.
Other fields may contain the result of a JavaScript (expression-based fields, or fields filled by an Action step) or the value of a property (property-based fields).

Adding fields

Location-based field

Generally, location-based fields are added to a Data Model by extracting data; see "Extracting data" on page 390. Location-based fields in **detail tables** are created by extracting transactional data; see "Extracting transactional data" on page 396.

Alternatively, you can add fields and detail tables directly in the Data Model pane (see "Adding empty fields via the Data Model pane" on page 427).
After adding a field or detail table this way, you can drag & drop data into it to convert it into a location-based field. Without data it is not accessible via the Step properties pane.

Expression-based field

Expression-based fields are filled with the result of a (JavaScript) expression: the script provides a value. Note that the last value attribution to a variable is the one used as the

result of the expression.

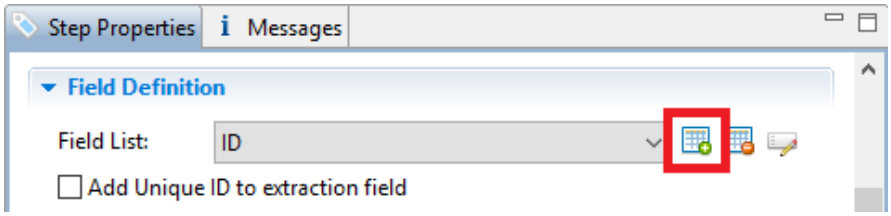
There is a number of ways to add an expression-based field.

Via the Steps pane

1. Make sure there is **no data selection** in the Data Viewer.
2. Right-click on an Extract step on the Steps pane and select **Add a Step > Add Extract Field**. (To add a new Extract step, select **Add a Step > Add Extraction** first.)
3. On the Step properties pane, under Field Definition, enter the script in the **Expression** field.

Via the Step properties pane

1. Select an Extract step on the Steps pane. (To add a new, empty Extract step, right-click the Steps pane and select **Add a Step > Add Extraction**.)
2. On the Step properties pane, under Field Definition, click the **Add JavaScript Field** button next to the Field List.



3. On the Step properties pane, under Field Definition, enter the script in the **Expression** field.

By changing a field's mode

Alternatively you can change a location-based field into a JavaScript-based field.

1. Select the field in the Data Model.
2. On the Step properties pane, under Field Definition, change its **Mode** to JavaScript.
3. Enter the script in the **Expression** field.

Tip: The default extraction method for fields in a CSV or XLS(X) file is `data.extract(columnName, rowOffset)`. Changing the expression to use the `data.extractByIndex(index, rowOffset)` method will allow the data mapping configuration to extract data from files that have the same structure, but different column names.

Property-based field

A property-based field is filled with the value of a **property** (see "Properties and runtime parameters" on page 387).

Custom properties can be added via the Preprocessor step; see "Preprocessor step" on page 410.

A property-based field cannot be added directly. To fill a field with the value of a property, you have to change an existing field's Mode to Properties.

1. Select the field in the Data Model.
2. On the Step properties pane, under Field Definition, change its **Mode** to **Properties**.
3. Select the property from the **Property** drop-down list, or click the button to the right, to open a filter dialog that lets you find a property based on the first few letters that you type.

Another way to add the value of a property to a field is by setting the field's Mode to **JavaScript** and entering the corresponding property in the Expression field, e.g. `data.properties.myProperty;`.

Adding empty fields via the Data Model pane

You can add fields and detail tables directly in the Data Model pane. Right-click anywhere on the Data Model and a contextual menu will appear. Which menu items are available depends on where you've clicked. If you right-click inside the record itself, you can add a field, detail table or group.

For a field, you can add a default value (see "Setting a default value" on page 429).

A field will be added with no extraction, while a detail table will be added with no fields inside.

Here's how you can fill the field with data.

- Drag & drop data into it. The field will be converted into a location-based field and is then accessible via the Step properties pane.
- Fill the field with an Action step (see Extracting data with a script). Fields that are filled this way are not accessible via the Step properties pane.

Adding fields dynamically

Outside of the DataMapper the Data Model cannot be changed. It isn't possible to add fields to it when using the data mapping configuration in an automated process. It is

however possible to add data to existing fields via Workflow; see "Adding fields and data via Workflow" on page 424.

Editing fields

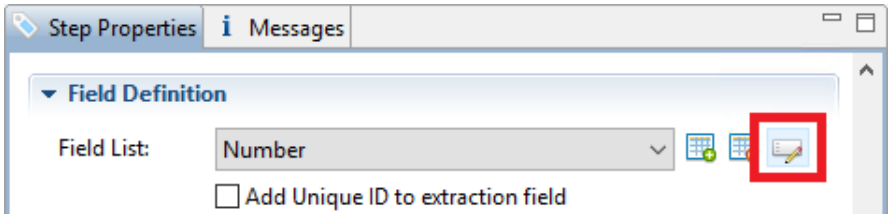
The list of fields that are included in the extraction, the order in which fields are extracted and the data format of each field, are all part of the Extract step's properties. These can be edited via the Step properties pane (see "Extract step properties" on page 492).

Tip: To change the name of a field quickly, right-click it in the Data Model and select **Rename**.

Renaming and reordering fields in an extraction step

The names of fields, as well as the order of fields in an extraction step, can be changed via the properties of the Extract step that they belong to.

1. Select the Extract step that contains the fields that you want to rename. To do this you could click on one of those fields in the Data Model, or on the step in the Steps pane.
2. On the Step properties pane, under Field Definition, click the **Order and rename fields** button.



3. Now you can rename and order the fields; see "Order and rename fields dialog" on page 499.

Note: About field **names**:

- Fields cannot have the same name, unless they are located on a different level in the record (e.g. root level, detail table).
- It is recommended that you **do not use spaces** in field names.
  When a data field is dragged into a template, the data field name is used in the selector of the script that is created, but any spaces will be removed.
  Also, data fields may be used as Metadata in a Workflow process, and spaces are not permitted in Metadata field names.

- It is recommended that you do not use special characters (" \ / ' £ $ & @ ~ #) in data field names. Special characters are not supported in a placeholder (e.g. @fieldname@) that is used as a selector for a script in a Designer template.

Note: The **order** of fields in an extraction step may be different from their order in the Data Model. To make the Data Model clearer and more manageable, you can change the order of the fields in it and create groups of fields. See also: "Ordering and grouping fields in the Data Model" on page 422.

Setting the data type

Fields store extracted data as a String by default. The data type of a field can be changed via the properties of the Extract step that the field belongs to.

1. Select the Extract step that contains the field. You can do this by clicking on the field in the Data Model, or on the step in the Steps pane that contains the field.
2. On the Step properties pane, under Field Definition, set the **Type** to the desired data type. See "Data types" on page 436 for a list of available types.

Changing the type does not only set the data type inside the record. In the case of dates, numbers and currencies, it also means that the DataMapper will expect the data in the data source to be formatted in a certain way. If the actual data doesn't match the format that the DataMapper expects, it cannot interpret the date, number or currency as such. If for example a date in the data source is formatted as "yyyy-mm-dd" but the default format adds the time, the date cannot be read and the DataMapper will stop with an error.

The default format for dates, numbers and currencies can be set in the user preferences ("DataMapper preferences" on page 1039), in the data source settings () and per data field (in the Extract step properties, see "Data Format" on page 498).

Setting a default value

You may want to set a default value for a field, in case no extraction can be made. Make sure to set the data type of the field via the step properties (see above). Then right-click the field and select **Default Value**.
The default value must match the selected data type. If the data type of the field is set to Integer, for example, you cannot enter a value of 2,3. A default date must be formatted as a DateTime object ("year-month-day" plus the time stamp, for example: 2012-04-11 12.00 AM);

see "Date" on page 439. The default value of a JSON field can be an object or an array; see "JSON" on page 444.

Modifying extracted data

To modify extracted data - the contents of a field - you have to write a script. The script can be entered as a Post function in a location-based field or as an Expression in a JavaScript-based field.

Post function

On the Step properties pane, under Field Definition, you can enter a script in the **Post function** field to be run after the extraction. (Click the **Use JavaScript Editor** button to open the "boundaries" on page 541 dialog if you need more space.)
A Post function script operates directly on the extracted data. Its results replace the extracted data. For example, the Post function script replace("-", ""); replaces the first dash character that occurs inside the extracted string. The code toLowerCase(); converts the string to lowercase letters.
Note that the function must be appropriate for the field's data type.

> **Tip:** Click and drag the Resize icon at the bottom right of the Post function area to make it larger.

JavaScript Expression

Alternatively you can change a field's **Mode** from Location to Javascript:

1. Select the field in the Data Model.
2. On the Step properties pane, under Field Definition, change its Mode to JavaScript.

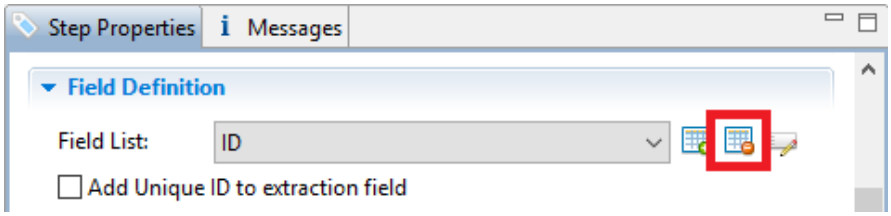
You will see that the JavaScript Expression field is not empty; it contains the code that was used to extract data from the location. This code can be used or deleted.

> **Note:** The last value attribution to a variable is the one used as the result of the expression.

Deleting a field

The list of fields that are included in an extraction is one of the properties of an Extract step. To delete a field:

1. Select the field: click on the field in the Data Model, or select the Extract step that contains the field that you want to delete, and in the Step properties pane, under Field Definition, select the field from the Field List.
2. In the Step properties pane, under Field Definition, click the **Remove Extract Field** button next to the Field List drop-down.



Detail tables

A detail table is a field in the Data Model that contains a record set instead of a single value. Detail tables contain transactional data. They are created when an **Extract** step is added within a **Repeat** step; see "Extracting transactional data" on page 396.

In the most basic of transactional communications, a single detail table is sufficient. However, it is possible to create multiple detail tables, as well as nested tables. Detail tables and nested tables are displayed as separate levels in the Data Model (see "The Data Model" on page 419).

Detail tables can be included in a template via a **Dynamic Table** ; see "Dynamic Table" on page 974.

Renaming a detail table

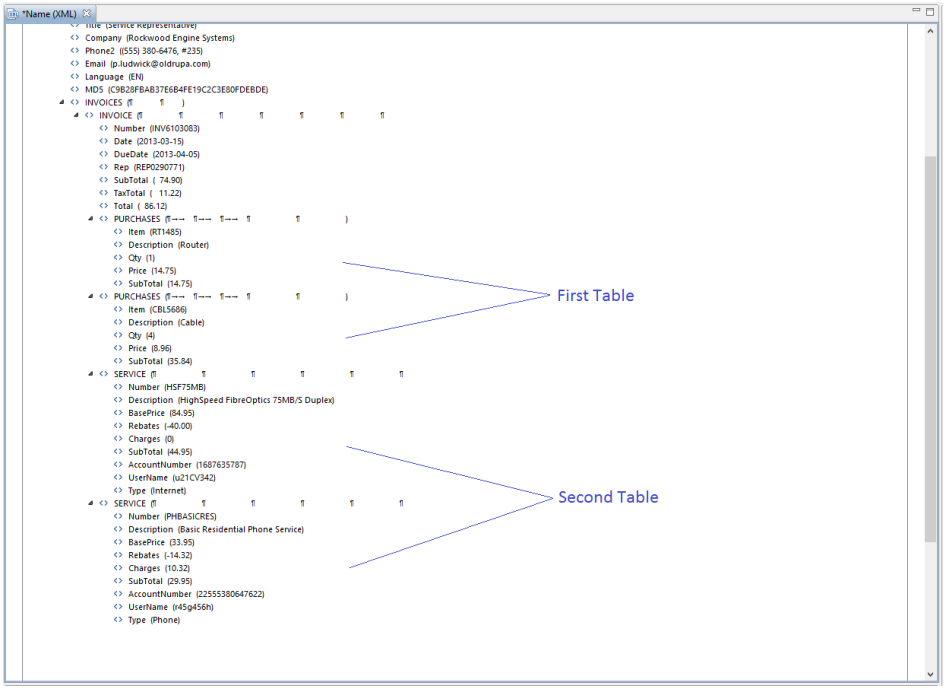
Renaming detail tables is especially useful when there are more detail tables in one record, or when a detail table contains another detail table. For this detail table, 'products' would be a better name.

1. On the Data Model pane, click one of the fields in the detail table.
2. On the Step Properties pane, under **Extraction Definition**, in the **Data Table** field, you can find the name of the detail table: **record.detail** by default. Change the **detail** part in that name into something else.

   Note: A detail table's name should always begin with '**record.**'.
3. Click somewhere else on the Step Properties pane to update the Data Model. You will see the new name appear.

Creating multiple detail tables

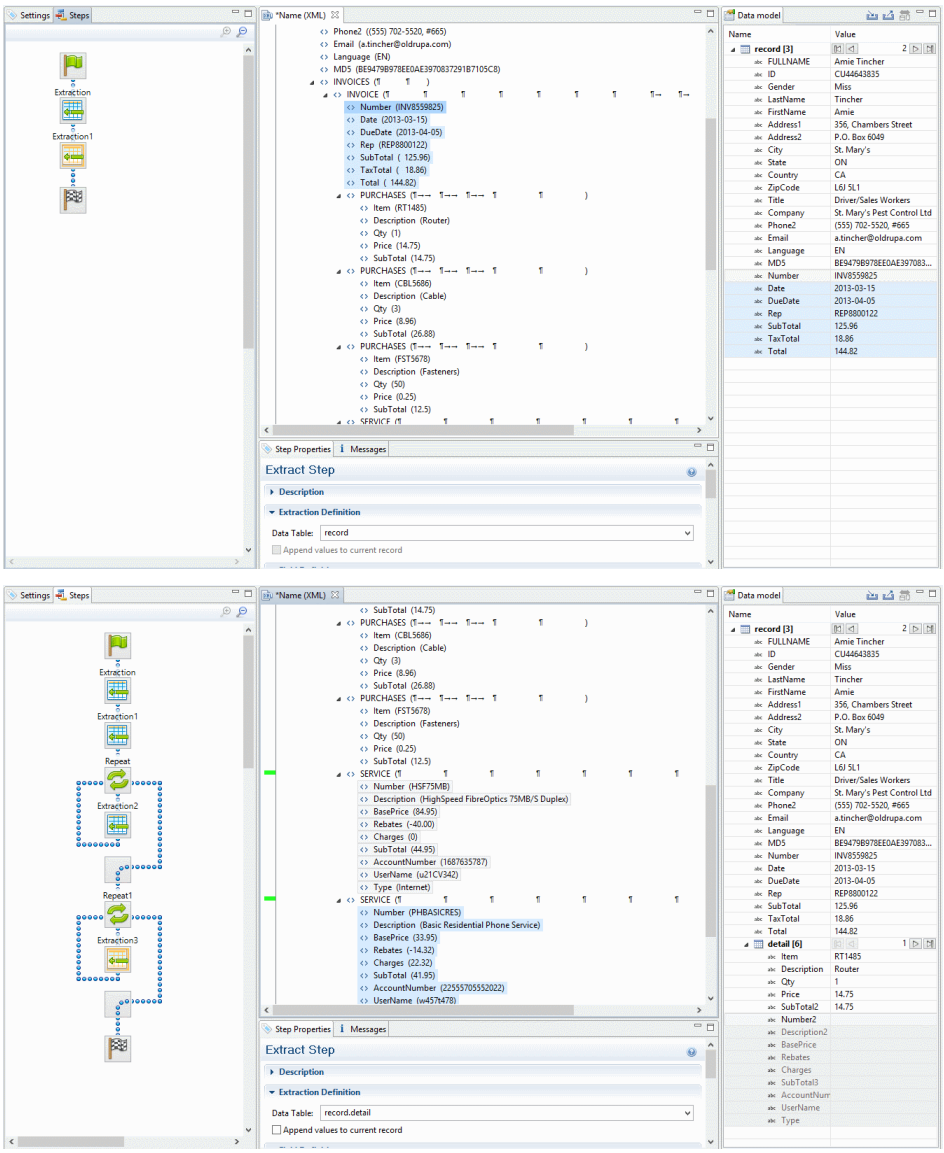
Multiple detail tables are useful when more than one type of transactional data is present in the source data, for example purchases (items with a set price, quantity, item number) and services (with a price, frequency, contract end date, etc).



To create more than one detail table, simply extract transactional data in different Repeat steps (see "Extracting transactional data" on page 396).

The best way to do this is to add an empty detail table (right-click the Data Model, select **Add a table** and give the detail table a name) and drop the data on the name of that detail table.

Else the extracted fields will all be added to one new detail table with a default name at first, and you will have to rename the detail table created in each Extract step to pull the detail tables apart (see "Renaming a detail table" on the previous page).



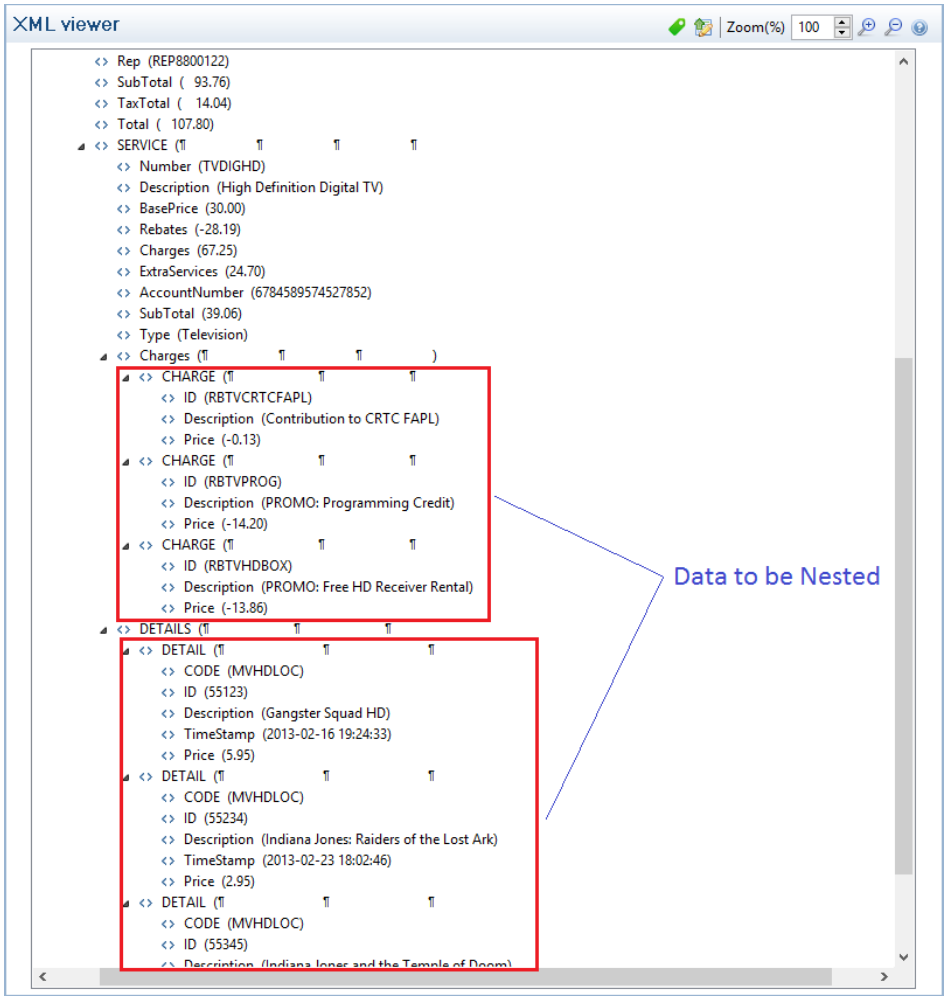
Nested detail tables

Nested detail tables are used to extract transactional data that are relative to other data. They are created just like multiple detail tables, with two differences:

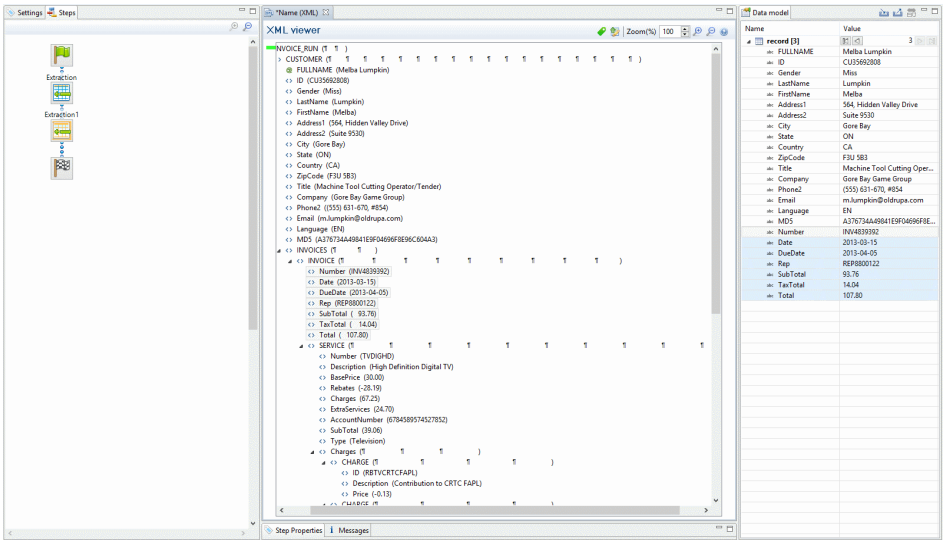
- For the tables to be actually nested, the **Repeat** step and its **Extract** step that extract the nested transactional data must be located **within** the **Repeat** step that extracts data to a detail table.
- In their name, the dot notation (record.services) must contain one extra level (record.services.charges).

Here is an example.

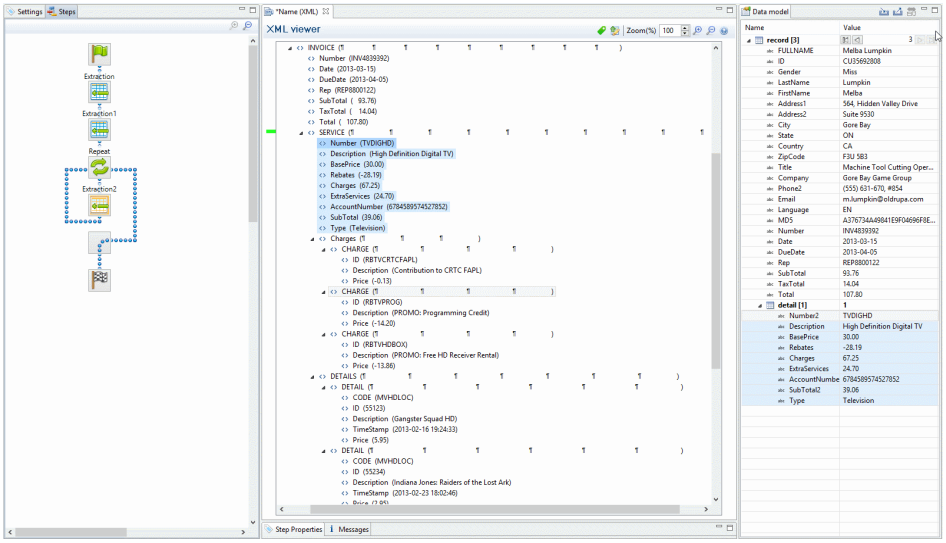
An XML source file lists the services of a multi-service provider: Internet, Cable, Home Phone, Mobile. Each service in turn lists a number of "charges", being service prices and rebates, and a number of "details" such as movie rentals or long distance calls.



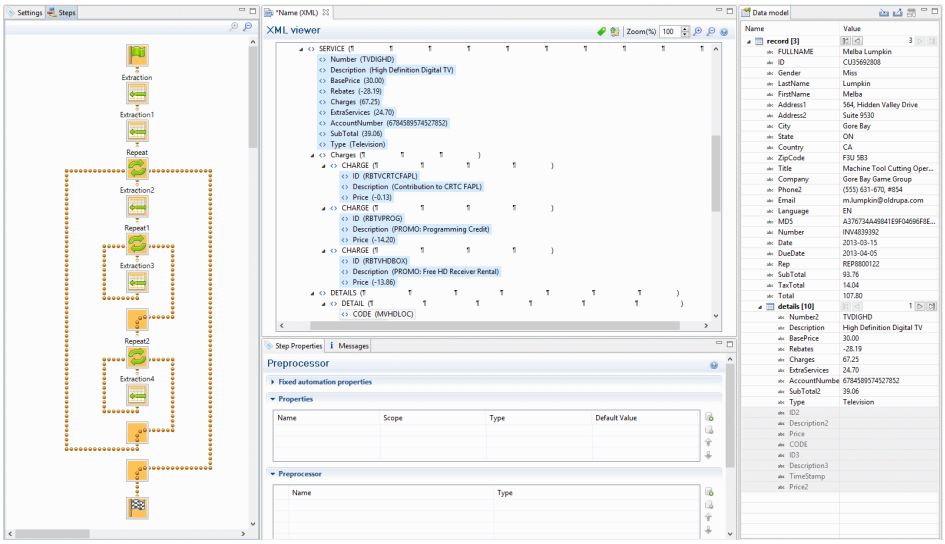
The services can be extracted to a detail table called **record.services**.



The "charges" and "details" can be extracted to two nested detail tables.



The nested tables can be called **record.services.charges** and **record.services.details**.



Now one "charges" table and one "details" table are created for each row in the "services" table.

Data types

By default the data type of extracted data is a String, but each field in the Data Model can be set to contain another data type.

To do this:

1. In the **Data Model**, select a field.
2. On the **Step properties** pane, under **Field Definition** choose a data type from the **Type** drop-down.

Changing the type does not only set the data type inside the record. In the case of dates, numbers and currencies, it also means that the DataMapper will expect the data in the data source to be formatted in a certain way. If the actual data doesn't match the format that the DataMapper expects, it cannot interpret the date, number or currency as such. If for example a date in the data source is formatted as "yyyy-mm-dd" but the default format adds the time, the date cannot be read and the DataMapper will stop with an error.

The default format for dates, numbers and currencies can be set in the user preferences ("DataMapper preferences" on page 1039), in the data source settings () and per data field (in the Extract step properties, see "Data Format" on page 498).

> **Note:** Data format settings tell the DataMapper how to read and parse data **from the data source**. They don't determine how these data are formatted in the Data Model or

in a template. In the Data Model, data are converted to the native data type. Dates, for example, are converted to a DateTime object. How they are displayed in the Data Model depends on the preferences (see "Default Format" on page 1039).

Tip: Read more about data types in the Data Model in the following how-to: DataMapper tidbits.

The following data types are available in OL Connect.

- "Boolean" below
- "String" on page 445
- "HTMLString" on page 442
- "Integer" on page 443
- "Float" on page 442
- "Currency" on the facing page
- "Date" on page 439
- "JSON" on page 444
- "Object" on page 445

  Note: The Object data type is only available in the DataMapper module. It can be used for properties in the Preprocessor step, but not for fields in the Data Model.

Boolean

Booleans are a simple true/false data type often used in conditions and comparisons.

Defining Boolean values

- **Preprocessor**:
  - In the **Step properties** pane, under **Properties**, add or select a field.
  - Specify the **Type** as **Boolean** and set a default value of either `true` or `false`, followed by a semicolon.
- **Extraction**:
  - In the **Data Model**, select a field.
  - On the **Step properties** pane, under **Field Definition** set the **Type** to **Boolean**.

  The field value must be `true` or `false`.

- **JavaScript Expression**: Set the desired value to either `true` or `false`.
  Example: `record.fields["isCanadian"] = true;`

Note: The value must be all in lowercase: `true`, `false`. Any variation in case (True, TRUE) will not work.

Boolean expressions

Boolean values can be set using an expression of which the result is true or false. This is done using operators and comparisons.

Example: `record.fields["isCanadian"] = (extract("Country") == "CA");`

For more information on JavaScript comparison and logical operators, please see MDN Web Docs: Expressions and Operators.

Currency

The Currency data type is a signed, numeric, fixed-point 64-bit number with 4 decimals. Values range from -922 337 203 685 477.5808 to 922 337 203 685 477.5808. This data type is routinely used for financial calculations: it is as precise as integers.

Defining Currency values

- **Preprocessor**:
  - In the **Step properties** pane, under **Properties**, add or select a field.
  - Specify the **Type** as **Currency** and set a default value as a number with up to 4 decimal points, followed by a semicolon; such as `546513.8798;`
- **Extraction**: The field value will be extracted and treated as a Float.
  - In the **Data Model**, select a field.
  - On the **Step properties** pane, under **Field Definition** set the **Type** to **Currency**.
  - Under **Data Format**, specify how the value is formatted in the data source (see "Extract step properties" on page 492; for the default format settings, see "Data source settings" on page 370).
- **JavaScript Expression**: Set the desired value to any Float value.
  Example: `record.fields["PreciseTaxSubtotal"] = 27.13465;`

Note: While Currency values can be set to up to 4 significant digits, only 2 are displayed on screen.

Building Currency values

Currency values can be the result of direct attribution or mathematical operations just like Integer values (see "Integer" on page 443).

Date

Dates are values that represent a specific point in time, precise up to the second. They can also be referred to as datetime values. While dates are displayed as UTC (Coordinated Universal Time) or using the system's regional settings (see "Default Format" on page 1039), in reality they are stored unformatted.

> **Note:** The **Date** property is stored in the **Connect** database with zero time zone offset, which makes it possible to convert the time correctly in any location. OL Connect Workflow, however, shows the date/time as it is stored in the database (with 0 time zone offset). This is expected behavior and the zone offset must be calculated manually in OL Connect Workflow.

Extracting dates

To extract data and have that data interpreted as a Date, set the type of the respective field to Date:

1. Select the field in the data model.
2. On the **Step properties** pane, under **Field Definition**, specify the **Type** as **Date**.
3. Make sure that the date in the data source is formatted in a way that matches the expectations of the DataMapper. If the date doesn't match the format that the DataMapper expects, it cannot be interpreted as a date. For example, if a date in the data source is formatted as "yyyy-mm-dd" but the DataMapper expects a time as well, the date cannot be read and the DataMapper will stop with an error.
   The expected date format can be set in three places:
   - In the user preferences ("DataMapper preferences" on page 1039).
   - In the data source settings ("Data source settings" on page 370).
   - In the field properties: on the **Step properties** pane, under **Data Format**, specify the **Date/Time Format**.
4. For the letters and patterns that you can use in a date format, see "Defining a date/time format" on the facing page.

   Data format settings tell the DataMapper how to read and parse data **from the data source**. They don't determine how these data are formatted in the Data Model or in a

template. In the Data Model, data are converted to the native data type. Dates, for example, are converted to a DateTime object. How they are displayed in the Data Model depends on the preferences (see "Default Format" on page 1039)..

Defining a date/time format

A date format is a mask representing the order and meaning of each digit in the raw data, as well as the date/time separators. The mask uses several predefined markers to parse the contents of the raw data. Here is a list of markers that are available in the DataMapper:

- **yy**: Numeric representation of the Year when it is written out with only 2 digits (i.e. 13)
- **yyyy**: Numeric representation of the Year when it is written out with 4 digits (i.e. 2013)
- **M**: Short version of the month name (i.e. Jan, Aug). These values are based on the current regional settings.
- **MM**: Long version of the month name (i.e. January, August). These values are based on the current regional settings.
- **mm**: Numeric representation of the month (i.e. 1, 09, 12)
- **D**: Short version of the weekday name (i.e. Mon, Wed). These values are based on the current regional settings.
- **DD**: Long version of the weekday name (i.e. Monday, Wednesday). These values are based on the current regional settings.
- **dd**: Numeric representation of the day of the month (i.e. 1, 09, 22)
- **hh**: Numeric representation of the hours
- **nn**: Numeric representation of the minutes
- **ss**: Numeric representation of the seconds
- **ms**: Numeric representation of the milliseconds.
- **ap**: AM/PM string.
- **x**: Indicates a time zone in the form of +05
- **xx**: Indicates a time zone in the form of +0500
- **xxx**: Indicates a time zone in the form of +05:00
- In addition, any constant character can be included in the mask, usually to indicate date/time separators (i.e. **/ - :**) . If one of those characters happens to be one of the reserved characters listed above, it must be escaped using the **** symbol.

Note: The markers that can be used when **extracting** dates are different from those that are used to **display** dates in a template (see the Designer's "Date and time patterns" on page 1491).

Examples of masks

Value in raw data	Mask to use
June 25, 2013	MM dd, YYYY
06/25/13	mm/dd/yy
2013.06.25	yyyy.mm.dd
2013-06-25 07:31 PM	yyyy-mm-dd hh:nn ap
2013-06-25 19:31:14.1206	yyyy-mm-dd hh:nn:ss.ms
Tuesday, June 25, 2013 @ 7h31PM	DD, MM dd, yyyy @ hh\hnnap

Entering a date using JavaScript

In several places in the DataMapper, Date values can be set through a JavaScript. For example:

- In a **field** in the Data Model. To do this, go to the Steps pane and select an Extract step. Then, on the Step properties pane, under Field Definition click the **Add JavaScript Field** button (next to the Field List drop-down). Type the JavaScript in the **Expression** field. (To rename the field, click the Order and rename fields button.)
- In a **Preprocessor property**. To do this, go to the Steps pane and select the Preprocessor step. Then, on the Step properties pane, under Properties add a property, specify its Type as Date and put the JavaScript in the Default Value field.

The use of the JavaScript `Date()` object is necessary when creating dates through a JavaScript expression. For more information, see MDN Web Docs: JavaScript Dates.

Example: The following script creates a date that is the current date + 30 days.

```
function addDays(date, days) {
    var result = new Date(date);
    result.setDate(result.getDate() + days);
    return result;
}
```

```
addDays(new Date(), 30);
```

Float

Floats are signed, numeric, floating-point numbers whose value has 15-16 significant digits. Floats are routinely used for calculations. Note that Float values can only have up to 3 decimals. They are inherently imprecise: their accuracy varies according to the number of significant digits being requested.
The Currency data type can have up to 4 decimals; see "Currency" on page 438.

Defining Float values

- **Preprocessor**:
  - In the **Step properties** pane, under **Properties**, add or select a field.
  - Specify the **Type** as **Float** and set a default value as a number with decimal points, followed by a semicolon; for example `546513.879;`.
- **Extraction**: The field value will be extracted and treated as a Float.
  - In the **Data Model**, select a field.
  - On the **Step properties** pane, under **Field Definition** set the **Type** to **Float**.
- **JavaScript Expression**: Set the desired value to any Float value.
  Example: `record.fields["PreciseTaxSubtotal"] = 27.134;`

Building Float values

Float values can be the result of direct attribution or mathematical operations just like Integer values (see "Integer" on the next page).

HTMLString

HTMLStrings contain textual data that includes HTML markup. They are essentially the same as String values except in cases where the HTML markup can be interpreted.

Example: Assume that a field has the value `He said <b>WOW!</b>`.
If the data type is String and the value is placed in a template, it will display exactly as "He said WOW!" (without the quotes).
If the data type is HTMLString, it will display as "He said **WOW!**" (again, without the quotes).

Reserved characters (such as < and >) that should not be interpreted as HTML should be represented in the data as HTML entities (such as < and >) or replaced with HTML entities after extraction using a Post function (in the "Extract step properties" on page 492).

Considering this is the only difference, for more information on how to create and use HTML String values, see "String" on page 445.

Integer

Integers are signed, numeric, whole 64bit numbers whose values range from -(2^63) to (2^63). Integers are the numerals with the highest precision (and the fastest processing speed) of all, since they are never rounded.

Defining Integer values

- **Preprocessor**:
  - In the **Step properties** pane, under **Properties**, add or select a field.
  - Specify the **Type** as **Integer** and set a default value as a number, such as `42`.
- **Extraction**: The field value will be extracted and treated as an integer.
  - In the **Data Model**, select a field.
  - On the **Step properties** pane, under **Field Definition** set the **Type** to **Integer**.
- **JavaScript Expression**: Set the desired value to any Integer value.
  Example: `record.fields["AnswerToEverything"] = 42;`

Building Integer Values

Integers can be set through a few methods, all of which result into an actual integer result.

- **Direct attribution**: Assign an integer value directly, such as `42`, `99593463712`, `record.fields.Total;` or `data.extract("TotalOrdered");`.
- **Mathematical operations**: Assign the result of any mathematical operation. For example: `22+51`, `3*6`, `10/5`, `record.fields["InvTotal"]`, or `sourceRecord.property.SubTotal`.

  For more information on mathematics in JavaScript , see MDN Web Docs - Arithmetic Operators. For more advanced mathematical functions, see MDN Web Docs - Math Object.

Note: When adding numbers that are not integers, for instance `4.5 + 1.2`, a `round towards zero` rounding is applied after the operation was made.
In the previous example, the result, `5.7`, is rounded to `5`.
In another example, `-1.5 - 1` results in `-2`

JSON

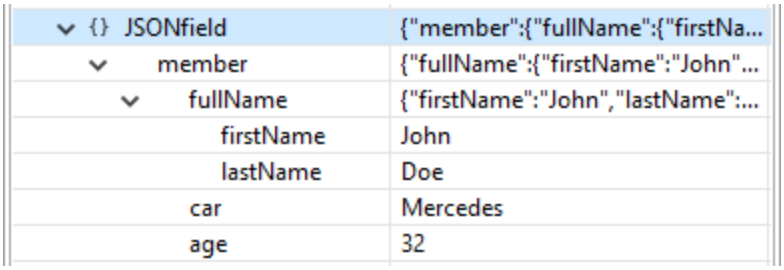
JSON is a text format for storing and transporting data.

This is an example of a JSON object.

Example:

```
{"member" : {
    "fullName":    {
        "firstName": "John",
        "lastName": "Doe"
        },
    "car":"Mercedes",
    "age":32
    }
}
```

In the Data Model, JSON objects are displayed in the JSON field on a single line. The individual keys and values appear in subsequent lines.



For a further explanation of the syntax see: MDN Web Docs.

Defining JSON values

- **Extraction**: Extracting (part of) a JSON structure by selecting it automatically creates a JSON field. However, extracting a single key-value pair results in a String field.

  You can also set the type of an existing field to JSON.

  1. In the **Data Model**, select the field.
  2. On the **Step properties** pane, under **Field Definition** set the **Type** to **JSON**.

- **JavaScript Expression**: Set the desired value as a JSON string, using the JavaScript function `JSON.stringify()`.

  Example: `record.fields["myJSON"] = JSON.stringify({"member":{"fullName":{"firstName":"John", "lastName":"Doe"},"car":"Mercedes","age":32}});`

Using JSON values in scripts

In scripts in the DataMapper and Designer, you can access properties inside a JSON field using the dot notation.

For instance, let's say that the JSON object at the top is extracted to a JSON field named `myJSON`. To access the value "Mercedes" in a standard Designer script, you would write `record.fields.myJSON.member.car`

The same notation can be used in expressions (see "Variable data in text: expressions" on page 999).

Object

Objects hold addresses that refer to objects. You can assign any reference type (string, array, class, or interface) to an Object variable. An Object variable can also refer to data of any value type (numeric, Boolean, Char, Date, structure, or enumeration).

Defining Object values

- **Preprocessor**:
  - In the **Step properties** pane, under **Properties**, add or select a field.
  - Specify the **Type** as **Object** and set a default value as a semi-colon.

String

Strings contain textual data. Strings do not have any specific meaning, which is to say that their contents are never interpreted in any way.

Defining String values

- **Preprocessor**:
  - In the **Step properties** pane, under **Properties**, add or select a field.
  - Specify the **Type** as **String** and set a default value as any text between quotes, followed by a semicolon.
    Example `"This is my text";`
- **Extraction**: The field value will be extracted and treated as a string.
  - In the **Data Model**, select a field.
  - On the **Step properties** pane, under **Field Definition** set the **Type** to **String**.
- **JavaScript Expression**: Set the desired value to any string between quotes.
  Example: `record.fields["countryOfOrigin"] = "Canada";`

Building String values

String values can be made up of more than just a series of characters between quotes. Here are a few tips and tricks to build strings:

- Both single and double quotes can be used to surround strings and they will act in precisely the same manner. So, "this is a string" and 'this is a string' mean the same thing. However, it's useful to have both in order to remove the need for escaping characters. For instance, "I'm fine!" works, but 'I'm fine!' does not since only 'I' is properly interpreted. 'I\'m fine!' works (escaping the ' with a \).
- It is possible to put more than one string, as well as variables containing strings, by concatenating them with the **+** operator. For example, `"Hello " + sourceRecord.property.FirstName + ", nice to meet you!"`.
- Adding more data to an existing string variable or field is possible using a combination of concatenation and assignment. For example, after the statements `var myVar = "Is this the real life";`, and `myVar += " or is this just fantasy?";`, the value of `myVar` will be, obviously `"Is this the real life or is this just fantasy?"`.

For more information on string variables, see quirksmode.org.

Data Model file structure

The Data Model file is an XML file that contains the structure of the Data M model, including each field's name, data type, and any number of detail tables and nested tables.

Example: promotional data

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<datamodel schemaVersion="1.0.0.3" name="Generic Address Block" version="1" xmlns="http://www.objectiflune.com/connectschemas/DataModelConfig" xsi:s-
chemaLocation="http://www.objectiflune.com/connectschemas/DataModelConfig http://www.objectiflune.com/connectschemas/DataModelConfig/1_0_0_3.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <field type="string" name="Name" required="true"/>
    <field type="string" name="Organization" required="true"/>
    <field type="string" name="Address1" required="true"/>
    <field type="string" name="Address2" required="true"/>
    <field type="string" name="Address3" required="true"/>
    <field type="string" name="City" required="true"/>
    <field type="string" name="StateOrProvince" required="true"/>
    <field type="string" name="Country" required="true"/>
    <field type="string" name="ZipOrPostalCode" required="true"/>
    <field type="string" name="Extra1" required="true"/>
    <field type="string" name="Extra2" required="true"/>
</datamodel>
```

Example: transactional details, in a simple invoice format

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<datamodel schemaVersion="1.0.0.3" name="Transactional Invoice" version="1"
xmlns="http://www.objectiflune.com/connectschemas/DataModelConfig"
xsi:schemaLocation="http://www.objectiflune.com/connectschemas/DataModelConfig http://www.objectiflune.com/connectschemas/DataModelConfig/1_0_0_
3.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <field type="string" name="ID" required="true"/>
    <field type="string" name="Gender" required="true"/>
```

```
    <field type="string" name="LastName" required="true"/>
    <field type="string" name="FirstName" required="true"/>
    <field type="string" name="Address1" required="true"/>
    <field type="string" name="Address2" required="true"/>
    <field type="string" name="City" required="true"/>
    <field type="string" name="State" required="true"/>
    <field type="string" name="Country" required="true"/>
    <field type="string" name="ZipCode" required="true"/>
    <field type="string" name="Title" required="true"/>
    <field type="string" name="Company" required="true"/>
    <field type="string" name="Phone2" required="true"/>
    <field type="string" name="Email" required="true"/>
    <field type="string" name="Language" required="true"/>
    <field type="string" name="Number" required="true"/>
    <field type="datetime" name="Date" required="true"/>
    <field type="datetime" name="DueDate" required="true"/>
    <field type="string" name="Rep" required="true"/>
    <field type="currency" name="SubTotal" required="true"/>
    <field type="currency" name="TaxTotal" required="true"/>
    <field type="currency" name="Total" required="true"/>
    <table name="detail">
        <field type="string" name="Number2" required="true"/>
        <field type="string" name="Description" required="true"/>
        <field type="currency" name="UnitPrice" required="true"/>
        <field type="integer" name="Ordered" required="true"/>
        <field type="integer" name="Shipped" required="true"/>
        <field type="integer" name="BackOrder" required="true"/>
        <field type="currency" name="Total2" required="true"/>
    </table>
</datamodel>
```

Example: nested tables (one table into another)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<datamodel schemaVersion="1.0.0.3" name="Nested Table Example" version="1"
xmlns="http://www.objectiflune.com/connectschemas/DataModelConfig"
xsi:schemaLocation="http://www.objectiflune.com/connectschemas/DataModelConfig http://www.objectiflune.com/connectschemas/DataModelConfig/1_0_0_
3.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <field type="string" name="RecordField" required="true"/>
    <table name="details">
        <field type="string" name="DetailsTableField" required="true"/>
        <table name="nestedtable">
            <field type="string" name="NestedTableField" required="true"/>
        </table>
    </table>
</datamodel>
```

Example: default values

Default values can be added to any field with the defaultValue attribute:

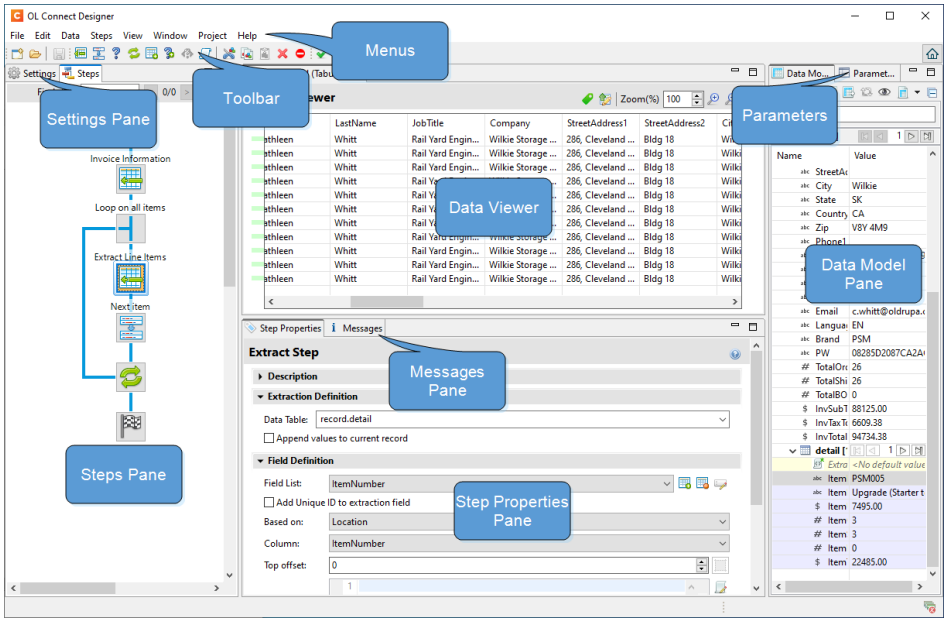
```
<field type="string" name="RecordField" required="true"  defaultValue="My Default Value">
```

DataMapper User Interface

The main ingredients in the DataMapper's user interface are the following:

- "Menus" on page 465
- "Toolbar" on page 527
- "Settings pane" on page 468

- "Steps pane" on page 486
- "The Data Viewer" on page 458
- "Step properties pane" on page 488
- "Messages pane" on page 464
- "Data Model pane" below
- Parameters pane (see "Properties and runtime parameters" on page 387)



Data Model pane

The Data Model pane displays the result of all the preparations and extractions of the extraction workflow. The pane displays the content of a single record within the record set at a time.
Data is displayed as a tree view, with the root level being the record table. On the level below that are detail tables, and a detail table inside a detail table is called a nested table. The Data Model is also used as a navigation tool between records and in all detail tables.

For more information see "The Data Model" on page 419.

Filter

To find a certain table, group or field in a large Data Model, start typing characters in the Filter box. This immediately narrows down the list of displayed items to all tables, groups and fields whose **name** or **value** contains the characters that were typed. Note that the filtering is case-insensitive.

Data Model toolbar buttons

- **Synchronize Fields and Structure**: Click to synchronize the Data Model fields and structure in the currently loaded template and data mapping configuration. If you click this button when working on the data mapping configuration, the Data Model gets updated to the one in the template. If you click it when working on the template, the Data Model gets updated to the one in the data mapping configuration.
- **Sample data file**: Hover over the button to see the name of the currently active sample data file. Click the down arrow and select a sample data file to quickly switch between available sample data files.
- **Collapse All**: Collapse all fields on the record level, in all detail tables and in all groups.
    - **Import Data Model**: Click to browse to a file that contains a Data Model. This may be:
        - A Data Model file (*.OL-datamodel).
        - A JSON file (*.json; see the note below).
        - A Connect template (*.OL-template), data mapping configuration (*.OL-datamapper) or 'typed' JSON file (*.json; see the note below).

      The file's data model structure will be displayed in the Data Model pane without data.
    - **Export Data Model**: Click to browse to a location to save the Data Model file. The available file types are:
        - A Data Model file (*.OL-datamodel).
        - A JSON file (*.json): A JSON object or an array of JSON objects representing records.
        - A 'typed' JSON file (*.json).

      Note: Typed JSON follows the structure of a JSON Record Data List (see the REST API Cookbook). Data field types are determined by the `schema` object in the JSON.
      When a Data Model is exported to a JSON file or typed JSON file, detail tables are exported, but groups are not.
    - **Show/Hide ExtraData fields**: This icon is only visible if the View ExtraData option has been checked in the Preferences: **Window** > **Preferences** > **Data Model**.

ExtraData fields are not meant to be filled via an extraction. They were originally intended to be used in a Workflow configuration to add data to the Data Model (see "Adding fields and data via Workflow" on page 424).

Nowadays, **runtime parameters** are the preferred way to pass data at runtime to a data mapping configuration (see "Properties and runtime parameters" on page 387) or template (see "Runtime parameters" on page 608).

- **Minimize/Maximize**: Click to minimize or maximize the pane. See Moving and Merging Panes.

Data Model contextual menu

The Data Model is generally constructed by extracting data; see "Extracting data" on page 390.
It is however possible to modify the Data Model, even if no data is present in the pane. To do this, open the contextual menu within the pane itself by right-clicking on something in the Data Model pane. Depending on where you've clicked, it can contain the following options.

Note: Moving and **grouping** fields in the Data Model has no impact on the order in which data are extracted, or on the final records in the OL Connect database. They only help you organize the Data Model visually.
To learn how to change the order in which data are extracted, see "Renaming and reordering fields in an extraction step" on page 428.

- **Add Field**: Click to add a new field at the current level (record or detail table). Enter the field name in the dialog; if you want you can change the field's data type and set a default value. Then click OK to add it to the Data Model.
- **Add Table**: Click to add a new detail table at the current level (record or existing detail table). Enter the table name in the dialog and click OK to add it.
- **Add Group**: Click to create a new group at the current level (record or existing group). Enter the group name in the dialog and click OK to add it. Any selected fields will be added to the group. (See also: "Ordering and grouping fields in the Data Model" on page 422.)

Note: Rename, Delete and Set Type are only available for Data Model fields or detail tables that are **not** filled via an extraction.
Fields and detail tables that are filled via an Extract step are to be changed (renamed, deleted etc.) via the properties of that Extract step; see: "Editing fields" on page 428 and "Renaming a detail table" on page 453.

- **Rename**: Click to rename the selected table, field or group. Enter the new name and click OK to rename.
- **Required**: Click to indicate that the field should be retained, even if there is no Extract step that references it. The DataMapper immediately discards non-required fields that are not referenced by any Extract step.
- **Delete**: Click to delete the selected table or field.
- **Set Type**: Use the list to select the field type (see "Data types" on page 436).
- **Ungroup**: Delete the selected group. The fields in the group will be moved up one level in the Data Model.
- **Default Value**: Click to set the default value for a field. This value is used if no extraction is present, or if an extraction attached to this field returns no value.
- **Move**: Click to move the selected field within the current level or group in the Data Model. (See: "Ordering and grouping fields in the Data Model" on page 422.)
  - **Top**: Move the field to the first place.
  - **Up**: Move the field one step up.
  - **Down**: Move the field one step down.
  - **Bottom**: Move the field to the last place.
- **Collapse All**: Collapse the fields on the record level, in all detail tables and in all groups.
- **Expand All**: Expand the fields on the record level, in all detail tables and in all groups.
- **Copy**: Copy the field's contents to the clipboard.
- **Copy Data Model**: Copy the simplified structure using key value pairs. This prepares a JSON representation of the Data Model, and copies it to the clipboard.
  The copied JSON is similar to the JSON created in the *Export Model → save as JSON files* option.
- **Copy Typed Data Model**: Copies the Typed (verbose) structure. This prepares a JSON representation of the Data Model, and copies it to the clipboard.
  The copied JSON is similar to the JSON created in the *Export Model → save as JSON files* option.

Field display

Fields in the Data Model pane are displayed in specific ways to simplify comprehension of the display data:

- The column on the left displays the **name** of the field.
- The column on the right displays the current **value** of the extracted field based on the record shown in the Data Viewer, if an Extract step has an extraction for this field (see "Extracting data" on page 390).
- The **icon** to the left of the name indicates the **data type** of the field (see "Data types" on page 436).
- A field name with an **asterisk** to the right indicates that this field is required. All imported data model fields are initially marked as required to prevent them from being removed, since the DataMapper immediately discards non-required fields that are not referenced by any Extract step.
- A field with a **grey background** indicates this Data Model field does not have any attached extracted data.
- A field with a **white background** indicates that the field has attached extracted data but the step extracting the data is not currently selected.
- A field with a **blue background** indicates that the field has attached extracted data and the step extracting the data is currently selected.
- A field or table with **red text** indicates a difference between the data model in the data mapping configuration and template.

Record navigation

Records can be navigated via the Data Model pane. The default record level navigates between records both in the Data Model pane and the Data Viewer, while each detail table has a similar navigation that influences that table and each detail table under it.

- **Expand/Contract**: Click to hide or show any fields or tables under the current table level or in the current group (see "Grouping fields" on page 423).
- **Table Name**: Displays the name of the table as well as the number of records at that level (in [brackets]). At the record level this is the number of records. In other levels it represents the number of entries in a detail table.
- **Number of Records**: The number of available records in the active data sample. This is affected by the Boundary settings (see "Record boundaries" on page 373 and "Settings pane" on page 468) and the Preprocessor step ("Preprocessor step" on page 410).
- **First Record**: Go to the first record in the data sample. This button is disabled if the first record is already shown.

- **Previous Record**: Go to the previous record in the data sample. This button is disabled if the first record is shown.
- **Current Record**: Displays the current record or table entry. Type a record number and press the **Enter** key to display that record. The number has to be within the number of available records in the data sample.
- **Next Record**: Go to the next record in the data sample. This button is disabled if the last record is shown.
- **Last Record**: Go to the last record in the data sample. This button is disabled if the last record is already shown. If a record limit is set in the Settings pane ("Settings pane" on page 468) the last record will be within that limit.

Detail tables

A detail table is a field in the Data Model that contains a record set instead of a single value. Detail tables contain transactional data. They are created when an **Extract** step is added within a **Repeat** step; see "Extracting transactional data" on page 396.

In the most basic of transactional communications, a single detail table is sufficient. However, it is possible to create multiple detail tables, as well as nested tables. Detail tables and nested tables are displayed as separate levels in the Data Model (see "The Data Model" on page 419).

Detail tables can be included in a template via a **Dynamic Table** ; see "Dynamic Table" on page 974.

Renaming a detail table

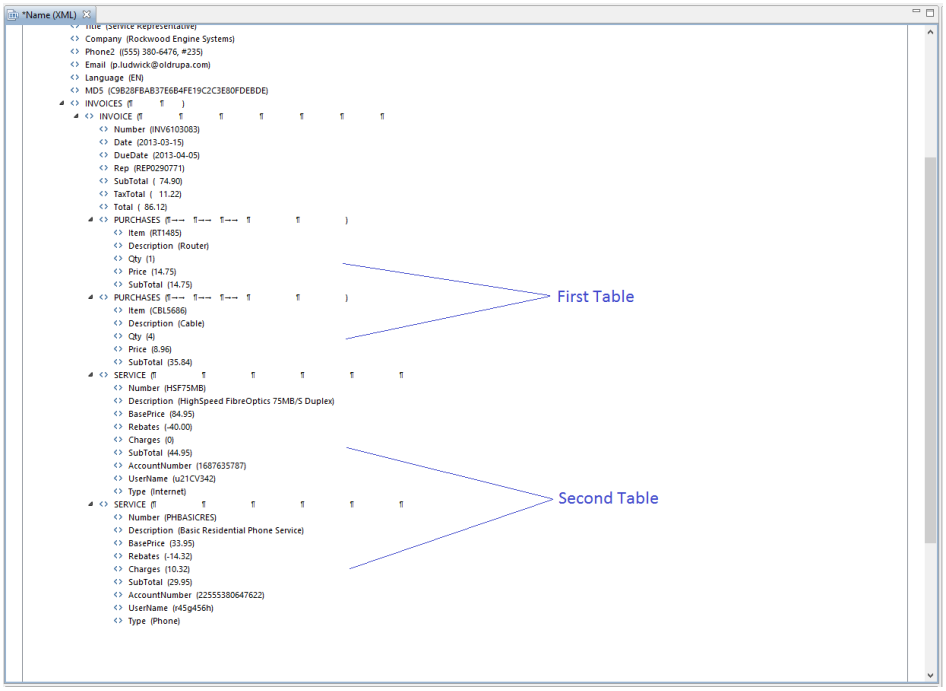
Renaming detail tables is especially useful when there are more detail tables in one record, or when a detail table contains another detail table. For this detail table, 'products' would be a better name.

1. On the Data Model pane, click one of the fields in the detail table.
2. On the Step Properties pane, under **Extraction Definition**, in the **Data Table** field, you can find the name of the detail table: **record.detail** by default. Change the **detail** part in that name into something else.

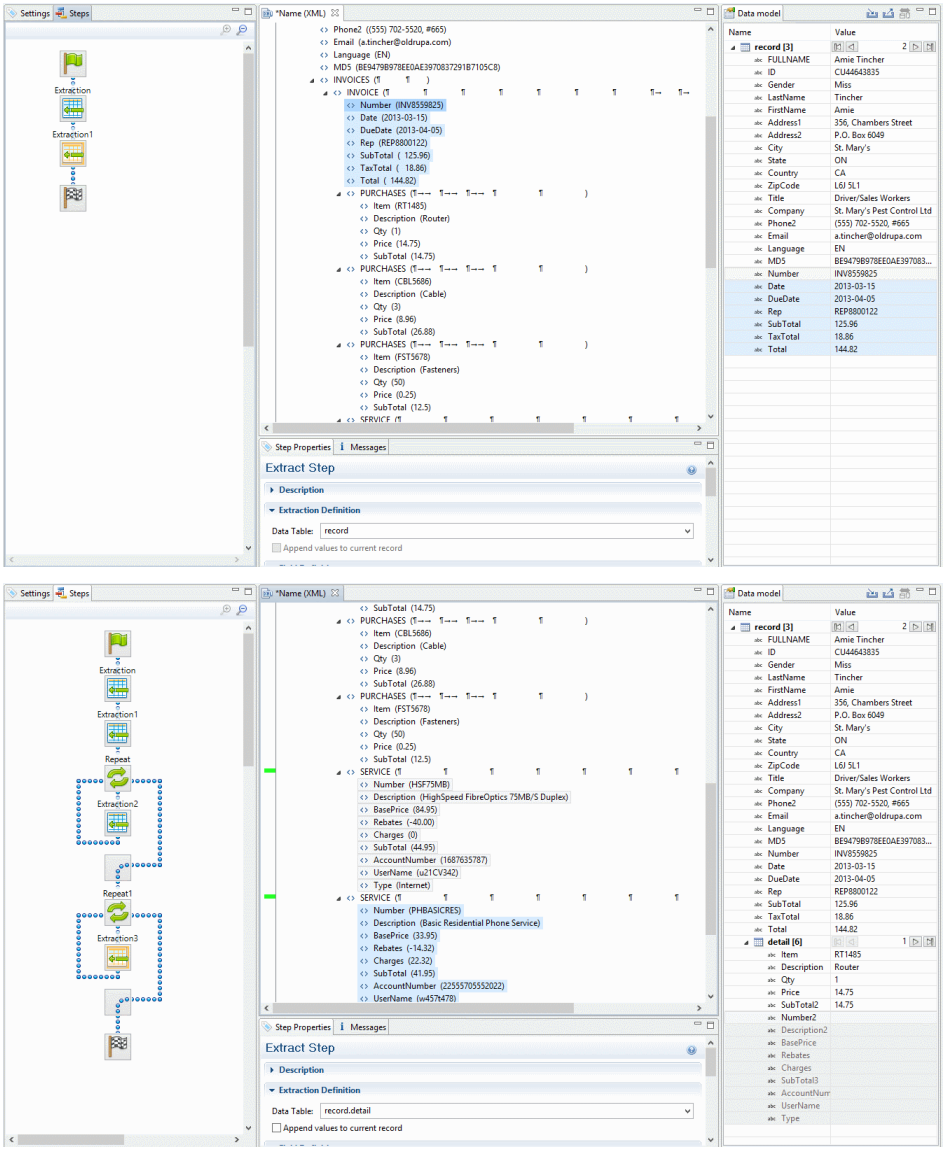
   Note: A detail table's name should always begin with '**record.**'.
3. Click somewhere else on the Step Properties pane to update the Data Model. You will see the new name appear.

Creating multiple detail tables

Multiple detail tables are useful when more than one type of transactional data is present in the source data, for example purchases (items with a set price, quantity, item number) and services (with a price, frequency, contract end date, etc).



To create more than one detail table, simply extract transactional data in different Repeat steps (see "Extracting transactional data" on page 396).
The best way to do this is to add an empty detail table (right-click the Data Model, select **Add a table** and give the detail table a name) and drop the data on the name of that detail table.
Else the extracted fields will all be added to one new detail table with a default name at first, and you will have to rename the detail table created in each Extract step to pull the detail tables apart (see "Renaming a detail table" on the previous page).



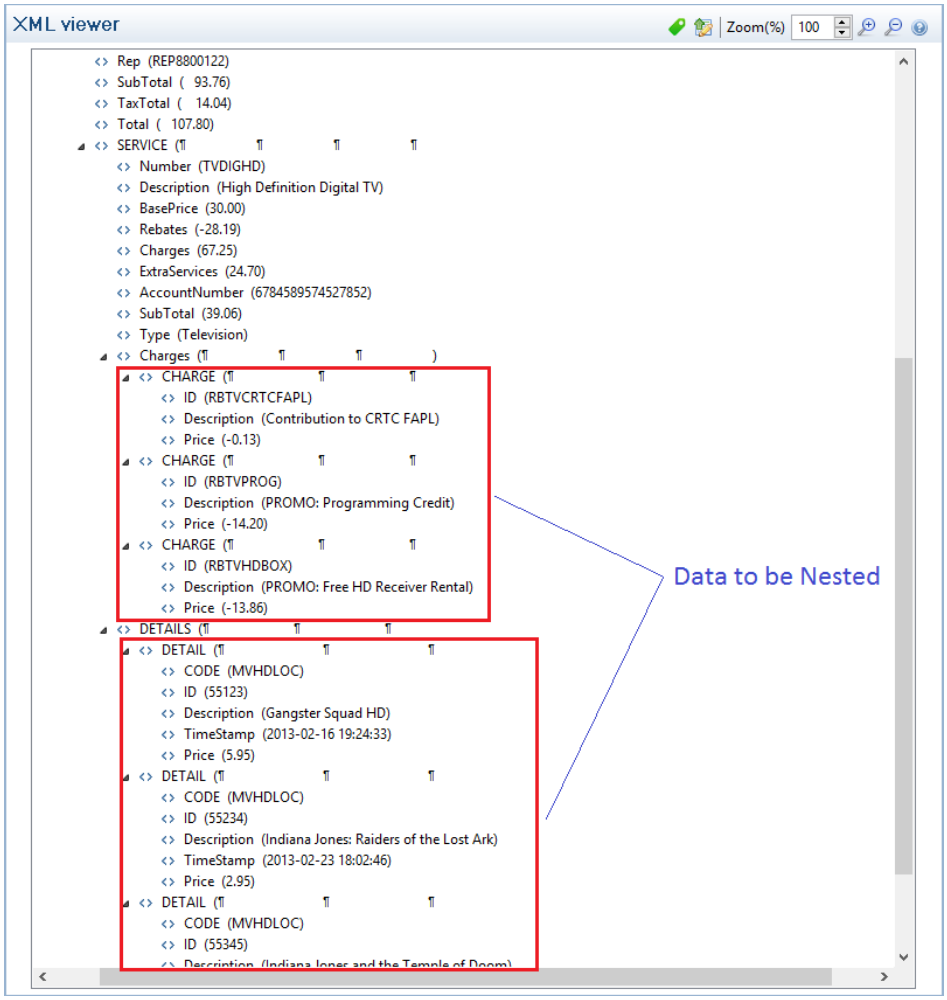
Nested detail tables

Nested detail tables are used to extract transactional data that are relative to other data. They are created just like multiple detail tables, with two differences:

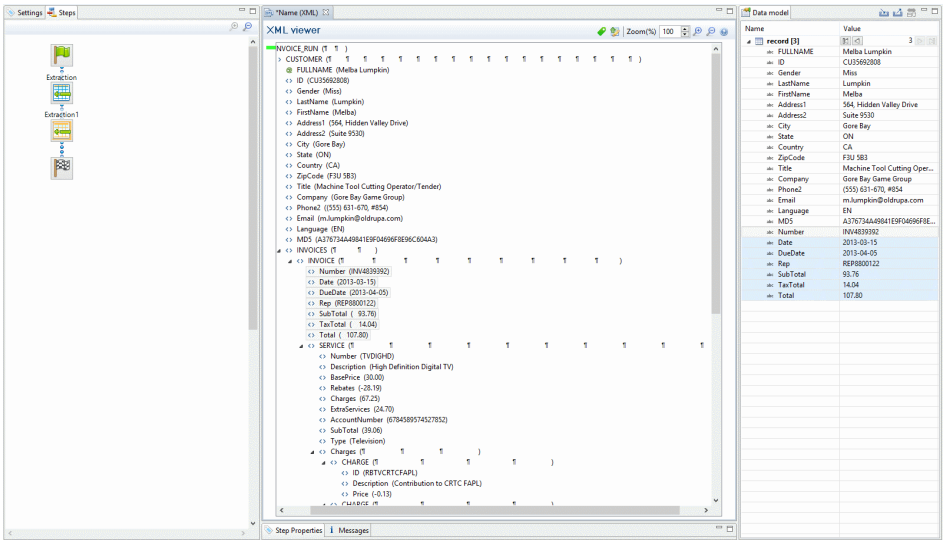
- For the tables to be actually nested, the **Repeat** step and its **Extract** step that extract the nested transactional data must be located **within** the **Repeat** step that extracts data to a detail table.
- In their name, the dot notation (record.services) must contain one extra level (record.services.charges).

Here is an example.

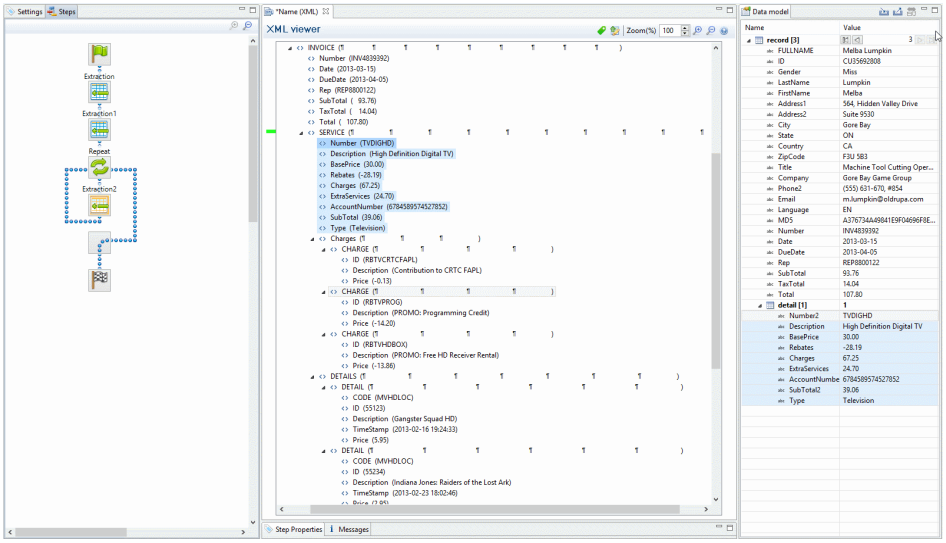
An XML source file lists the services of a multi-service provider: Internet, Cable, Home Phone, Mobile. Each service in turn lists a number of "charges", being service prices and rebates, and a number of "details" such as movie rentals or long distance calls.



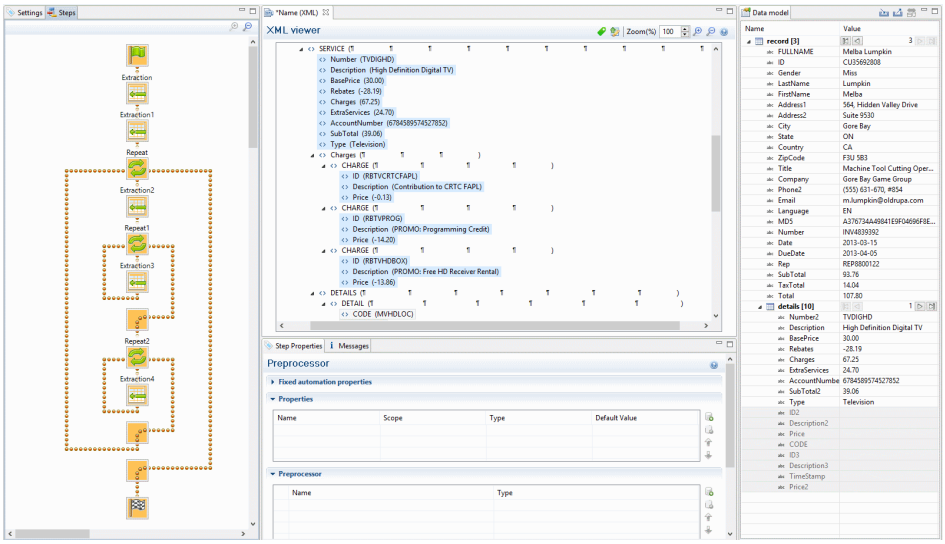
The services can be extracted to a detail table called **record.services**.



The "charges" and "details" can be extracted to two nested detail tables.



The nested tables can be called **record.services.charges** and **record.services.details**.



Now one "charges" table and one "details" table are created for each row in the "services" table.

The Data Viewer

The Data Viewer is located in the middle on the upper half of the DataMapper screen. It displays the data source that is currently loaded in the DataMapper, specifically one record in that data. Where one record ends and the next starts is determined in the Data Source settings (see "Record boundaries" on page 373). One record may contain more than one unit: PDF or Text pages, XML nodes, CSV lines, etc.
When the Delimiter or Boundary options are set in the Settings pane, the Data Viewer reflects those changes.

Any modification of the source data by a Preprocessor takes place before the data is displayed in the Data Viewer (see "Preprocessor step" on page 410).

The Data Viewer lets you select data, extract them ("Extracting data" on page 390), and apply a condition where necessary.
How data can be selected depends on the type of source file (see "Selecting data" on page 394).

Once data is extracted, clicking on any Extract step on the **Steps** pane highlights any area from which it extracts data in the Data Viewer. You can click on the Preprocessor step to select all the steps in the extraction workflow and highlight all extracted data.
Clicking on other step types also has a visible effect in the Data Viewer:Clicking on a Repeat step shows where the loop takes place.Clicking on a Goto step shows where the cursor is moved.Clicking on a Condition step shows which data fulfil the condition.For more

information about the different steps that can be added to a data mapping workflow, see "Steps" on page 409.

Data Viewer toolbar

The Data Viewer has a **toolbar** at the top to control options in the viewer. Which toolbar features are available depends on the data source type.

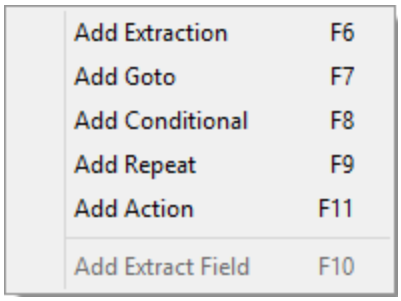
- **Font** (Text file only): Use the drop-down to change the font used to display text. Useful for double-byte data. It is recommended that monospace fonts be used.
- **Hide/Show line numbers** (Text file only): Click to show or hide the line numbers on the left of the Data Viewer.
- **Hide/Show datamap**: Click to show or hide the icons to the left of the Data Viewer which displays how the steps affect the line.
- **Hide/Show extracted data**: Click to show or hide the extraction selections indicating that data is extracted. This simplifies making data selections in the same areas and is useful to display the original data.
- **Lock/Unlock extracted data**: Click to lock existing extraction selections so they cannot be moved or resized. This simplifies making data selections in the same area.
- **Zoom Level**: Use the arrows to adjust the zoom level, or type in the zoom percentage.
- **Zoom In (CTRL +)**: Click to zoom in by increments of 10%
- **Zoom Out (CTRL -)**: Click to zoom out by increments of 10%

Additional Keyboard Shortcuts for XML and JSON files:

- **+** (while on an XML node or JSON element with children): Expand the XML node/JSON element.
- **-** (while on an XML node or JSON element with children): Collapse the XML node/JSON element, hiding all its child nodes/elements.

Contextual Menu

You can access the contextual menu using a right-click anywhere inside the Viewer window.



Note: The **Add Extract Field** item is available only after an **Extract** step has been added to the workflow.

Keyboard shortcuts

This topic gives an overview of keyboard shortcuts that can be used in the DataMapper. Keyboard shortcuts available in the Designer for menu items, script editors and the data model pane can also be used in the DataMapper; see "Keyboard shortcuts" on page 1223. Although some of the keyboard shortcuts are the same, this isn't a complete list of Windows keyboard shortcuts. Please refer to Windows documentation for a complete list of Windows keyboard shortcuts.

Menu items

The following key combinations activate a function in the menu.

Key combination	Function
Alt	Put the focus on the menu. (Alt + the underlined letter in a menu name displays the corresponding menu.) The menu can then be browsed using the Enter key, arrow up and arrow down buttons.
Alt + F4	Exit
Ctrl + C or Ctrl + Insert	Copy
Ctrl + N	New
Ctrl + O	Open file
Ctrl + Shift + O	Open configuration file
Ctrl + S	Save file

Key combination	Function
Ctrl + V or Shift + Insert	Paste
Ctrl + X	Cut
Ctrl + W or Ctrl + F4	Close file
Ctrl + Y or Ctrl + Shift + Y	Redo
Ctrl + Z or Ctrl + Shift + Z	Undo
Ctrl + Shift + S	Save all
Ctrl + Shift + W or Ctrl + Shift + F4	Close all
Ctrl + F5	Revert
Ctrl + F7	Next view
Ctrl + Shift + F7	Previous view
Ctrl + F8	Next perspective
Ctrl + Shift + F8	Previous perspective
Ctrl + F10	Save as
Ctrl + F12	Send to Workflow / Package files
F4	Ignore step/Reactivate step
F6	Add an Extract step
F7	Add a Goto step
F8	Add a Condition step
F9	Add a Repeat step
F10	Add an Extract field

Key combination	Function
F11	Add an Action step
F12	Add a Multiple Conditions step
Alt + F12	Add a Case step (under a Multiple Conditions step)
Home	Go to the first step in the workflow
End	Go to the last step in the workflow
Alt + V	Validate records
Shift + F10 or Ctrl + Shift + F10	Open context menu

Viewer pane

The following key combinations activate a function in the **Viewer**.

Key combination	Function
Alt + **-**	Open system menu
Ctrl + **-**	Zoom out **Tip:** You can also use the mouse's scroll wheel in combination with the Ctrl button to gradually zoom in or out.
Ctrl + **+**	Zoom in
Ctrl + Shift + E	Switch to Editor
Ctrl + F6	Next editor (when there is more than one file open in the Workspace)
Ctrl + Shift + F6	Previous editor (when there is more than one file open in the Workspace)

Data Model pane

Key combination	Function
PageUp	Go to previous record
PageDown	Go to next record
Alt + CR	Property page
Alt + PageDown	Scroll down to the last field
Alt + PageUp	Scroll up to the first field

Steps tab

Key combination	Function
Ctrl + **-**	Zoom out
Ctrl + **+**	Zoom in

Edit Script and Expression windows

The following key combinations have a special function in the **Expression** and in the **Edit Script** windows (expanded view).

Key combination	Function
Ctrl + space	Content assist (auto-complete)
Ctrl + A	Select all
Ctrl + D	Duplicate line
Ctrl + I	Indent (Tab)
Ctrl + J	Line break
Ctrl + L	Go to line; a prompt opens to enter a line number.
Ctrl + Shift + D	Delete line
Shift + Tab	Shift selected lines left

Key combination	Function
Tab	Shift selected lines right
Ctrl + /	Comment out / uncomment a line in code
Ctrl + Shift + /	Comment out / uncomment a code block

Messages pane

The Messages pane is shared between the DataMapper and Designer modules and displays any warnings and errors from the data mapping configuration or template.

At the top of the Message pane are control buttons:

- **Export Log**: Click to open a Save As dialog where the log file (.log) can be saved on disk.
- **Clear Log Viewer**: Click to remove all entries in the log viewer.
- **Filters**: Displays the Log filter (see "Log filter" below).
- **Activate on new events**: Click to disable or enable the automatic display of this dialog when a new event is added to the pane.
- **Time**: The date and time when the error occurred.
- **Type**: Whether the entry is a warning or an error.
- **Source**: The source of the error. This indicates the name of the step as defined in its step properties.
- **Message**: The contents of the message, indicating the actual error.

Log filter

The log filter determines what kind of events are show in the Messages pane (see "Messages pane" above).

- **Event Types** group:
  - **OK**: Uncheck to hide OK-level entries.
  - **Information**: Uncheck to hide information-level entries.

    - **Warning**: Uncheck to hide any warnings.
    - **Error**: Uncheck to hide any critical errors.
- **Limit visible events to**: Enter the maximum number of events to show in the Messages Pane. Default is 50.

Menus

The following menu items are shown in the DataMapper Module's menu:

File Menu

- **New...**: Opens the dialog to create a new data mapping configuration; see "Creating a new data mapping configuration" on page 338.
- **Open**: Opens a standard File Open dialog. This dialog can be used to open Templates and data mapping configurations.
- **Open Recent**: List the most recently opened Templates and configurations. Clicking on a template will open it in the Designer module, clicking on a data mapping configuration will open it in the DataMapper module.
- **Close**: Close the currently open data mapping configuration or Template. If the file needs to be saved, the appropriate Save dialog will open.
- **Close All**: Close any open data mapping configuration or Template. If any of the files need to be saved, the Save Resources dialog opens.
- **Save**: Saves the current data mapping configuration or Template to its current location on disk. If the file is a data mapping configuration and has never been saved, the Save As dialog appears instead.
- **Save As...**: Saves the current data mapping configuration or Template to a new location on disk. In the case of Templates, it is saved to a location that can be different than the local repository.
- **Save All**: Saves all open files. If any of the open files have never been saved, the Save As dialog opens for each new unsaved file.
- **Save a Copy**: Save a copy of the current data mapping configuration in the selected Connect version's format. See "Down-saving a data mapping configuration" on page 343.
- **Revert**: Appears only in the Designer module. Reverts all changes to the state in which the file was opened or created.
- **Data**:

    - **Add Data Sample...**: Opens the dialog to add a new data file to the currently loaded data mapping configuration. Not available if the currently loaded data mapping configuration connects to a database source.
- **Send to Workflow**: Opens the "Send to Workflow dialog" on page 1206 dialog to send files to a local OL Connect Workflow software installation.
- **Exit**: Closes the software. If any of the files need to be saved, the Save Resources dialog opens.

Edit Menu

- **Undo**: Undoes the previous action.
- **Redo**: Redoes the last action that was undone.
- **Cut Step**: Removes the currently selected step and places it in the clipboard. If the step is a Repeat or a Condition, all steps under it are also placed in the clipboard. If there is already a step in the clipboard, it will be overwritten.
- **Copy Step**: Places a copy of the currently selected step in the clipboard. The same details as the Cut step applies.
- **Paste Step**: Takes the step or steps in the clipboard and places them in the Steps after the currently selected step.
- **Delete Step**: Deletes the currently selected step. If the step is a Repeat or Condition, all steps under it are also deleted.
- **Cut**: Click to remove the currently selected step, or steps, and place them in the clipboard.
- **Copy**: Click to place a copy of the currently selected step, or steps, in the clipboard.
- **Paste**: Click to place any step, or steps, from the clipboard before the currently selected step in the "Steps pane" on page 486.

Data Menu

- **Hide/Show datamap**: Click to show or hide the icons to the left of the Data Viewer that displays how the steps affect the line.
- **Hide/Show extracted data**: Click to show or hide the extraction selections indicating that data is extracted. This simplifies making data selections in the same areas and is useful to display the original data.
- **Validate All Records**: Runs the Steps on all records and verifies that no errors are present in any of the records. Errors are displayed in the "Messages pane" on page 464.

Steps

- **Ignore Step/Reactivate Step**: Click to set the step to be ignored (aka disabled) or to reactivate it. Disabled steps do not run when in DataMapper and do not execute when the data mapping configuration is executed in Workflow. However, they can still be modified normally.
- **Add Extract Step**: Adds an Extract Step with one or more extract fields. If more than one line or field is selected in the Data Viewer, each line or field will have an extract field.
- **Add Goto Step**: Adds a Goto step that moves the selection pointer to the beginning of the data selection. For instance if an XML node is selected, the pointer moves to where that node is located.
- **Add Condition Step**: Adds a condition based on the current data selection. The "True" branch gets run when the text is found on the page. Other conditions are available in the step properties once it has been added.
- **Add Repeat Step**: Adds a loop that is based on the current data selection, and depending on the type of data. XML data will loop on the currently selected node, CSV loops for all rows in the record. In Text and PDF data, if the data selection is on the same line as the cursor position, the loop will be for each line until the end of the record. If the data selection is on a lower line, the loop will be for each line until the text in the data selection is found at the specified position on the line (e.g. until "TOTAL" is found).
- **Add Extract Field**: Adds the data selection to the selected Extract step, if an extract step is currently selected. If multiple lines, nodes or fields are selected, multiple extract fields are added simultaneously.
- **Add Multiple Conditions**: Adds a condition that splits into multiple case conditions.
- **Add Action Step**: Adds a step to run one or more specific actions such as running a JavaScript expression or setting the value of a Source Record Property.

View Menu

- **Zoom In**: Click to zoom in the "Steps pane" on page 486.
- **Zoom Out**: Click to zoom out the "Steps pane" on page 486.

Window Menu

- **Show View**
  - **Messages**: Shows the "Messages pane" on page 464
  - **Data Model**: Shows the "Data Model pane" on page 448.
  - **Steps**: Shows the "Steps pane" on page 486.
  - **Parameters**: Shows the Parameters pane. See "Properties and runtime parameters" on page 387.
  - **Settings**: Shows the "Settings pane" below.
  - **Step Properties**: Shows the "Step properties pane" on page 488
- **Reset Perspective**: Resets all toolbars and panes to the initial configuration of the module.
- **Preferences**: Click to open the "Preferences" on page 1035 dialog.

Help Menu

- **Software Activation**: Displays the Software Activation dialog. See "Activating a License" on page 64.
- **Help Topics**: Click to open this documentation.
- **Contact Support**: Click to open the Objectif Lune Contact Page in the default system Web browser.
- **About OL Connect Designer**: Displays the software's About dialog.
- **Welcome Screen**: Click to re-open the Welcome Screen.

Settings pane

Settings for the data source and a list of Data Samples and JavaScript files used in the current data mapping configuration, can be found on the Settings tab at the left. The available options depend on the type of data sample that is loaded.
The Input Data settings (especially Delimiters) and Boundaries are essential to obtain the data and eventually, the output that you need. For more explanation, see "Data source settings" on page 370.

Input Data

The **Input Data** settings specify how the input data must be interpreted. These settings are different for each data type. For a CSV file, for example, it is important to specify the

delimiter that separates data fields. PDF files are already delimited naturally by pages, so the input data settings for PDF files are interpretation settings for text in the file.

- "Input Data settings" on page 472
- "Excel file Input Data settings" on page 473
- "PDF file Input Data settings" on page 474
- "Database Input Data settings" on page 476
- "Text file Input Data settings" on page 477
- "XML File Input Data settings" on page 478
- "JSON File Input Data settings" on page 480

Boundaries

Boundaries are the division between **records**: they define where one record ends and the next record begins; for an explanation see "Record boundaries" on page 373.

- "CSV, Excel or Database file boundaries" on page 480
- "PDF file boundaries" on page 481
- "Text file boundaries" on page 482
- "XML file boundaries" on page 484
- "JSON file boundaries" on page 484

Data samples

The Data Sample area displays a list of all the imported Data Samples that are available in the current data mapping configuration. As many Data Samples as necessary can be imported to properly test the configuration.
Only one of the data samples - the active data sample - is shown in the Data Viewer.

A number of buttons let you manage the Data Samples.
In addition to using the buttons listed below, you can right-click a file to bring up the context menu, which offers the same options plus the **Copy** and **Paste** options.

To select or deselect multiple Data Samples, keep the **Ctrl** key pressed down while clicking on them, or keep the **Shift** key pressed down to select consecutive Data Samples.

> **Tip:** Data samples can be copied and pasted **to** and **from** the Settings pane using Windows File Explorer.

- **Add**: Add a new Data Sample from an external data source. The new Data Sample will need to be of the same data type as the current one. For example, you can only add PDF files to a PDF data mapping configuration. Multiple files can be added simultaneously.
- **Delete**: Remove the current Data Sample from the data mapping configuration.
- **Move up**: Move the selected Data Sample(s) up the list.
- **Move down**: Move the selected Data Sample(s) down the list.
- **Replace**: Open a Data Sample and replace it with the contents of a different data source.
- **Reload**: Reload the currently selected Data Sample and any changes that have been made to it.
- **Set as Active**: Activates the selected Data Sample. The active data sample is shown in the Data Viewer after it has gone through the **Preprocessor** step as well as the **Input Data** and **Boundary** settings.

  Tip: You can also switch between sample data files by clicking the down arrow in the Data Model toolbar and selecting the desired sample data file.

Editor Data Format

The Editor Data Format setting is only available for Excel files.

- **Date Display Format**: This setting specifies how dates must be displayed in the Data Viewer. Note that extracting a Date value will only be successful if the expected date format matches the actual format of a date in the Data Viewer (see:"Data format settings" on page 373.)
  - **Excel Default Format**: Displays dates and times the way they would be displayed in Excel, using the specified locale. For date formats without a locale, the US English locale is used. (In the format selection dialog in Excel, these date formats are marked with an asterisk.) Values can show a date, or a time, or both. Connect always uses this setting when opening an Excel file.
  - **Current Locale Settings**: Shows dates and times as formatted by Windows using the current Locale of the system on which Connect runs. All values are shown as a date including a time.

- **ISO 8601 (UTC)**: Uses the ISO 8601 (UTC) format to display dates and times. All values are shown as a date including a time, taking the time zone into account. Note that when no time was specified with a date in the original file, the default time (12.00 AM) is used and converted; this may influence the displayed date.

Note: Some Korean and Chinese date formats can't be parsed yet, and won't display correctly with any of these settings.

External JS Libraries

Right-clicking in the box brings up a control menu, with the same options as are available through the buttons on the right.

- **Add**: Add a new external library. Use the standard **Open** dialog to browse and open the .js file.
  - **Encoding** dialog: Appears after file selection, to allow selecting the file's encoding. Defaults to UTF-8 unless invalid characters are found.
- **Delete**: Remove the currently selected library from the data mapping configuration.
- **Replace**: Open a library and replace it with the contents of a different js file.
- **Reload**: Reload the currently selected library and any changes that have been made to it.

Default Data Format

The Default Data Format settings defined here apply to any new extraction in made in the current data mapping configuration. Any format already defined for an existing field remains untouched.
It is also possible to set a default format for dates and currencies in the user preferences ("DataMapper preferences" on page 1039).
Specific settings for a field that contains extracted data are made via the properties of the Extract step that the field belongs to (see "Editing fields" on page 428).

- **Negative Sign Before**: Any value in a numeric field that has a "-" sign is interpreted as a negative value.
- **Decimal Separator**: Set the decimal separator for a numerical value.
- **Thousand Separator**: Set the thousand separator for a numerical value.
- **Currency Sign**: Set the currency sign for a currency value.

- **Treat empty as 0**: A numerical empty value is treated as a 0 value.
- **Date/Time Format**: Set the date format for a date value.
    - **Automatic**: Select this option to parse dates automatically, without specifying a format. This is the default setting for new Date fields.
    - **ISO8601**: This setting allows for dates with different timestamp formats, or belonging to different time zones, to be parsed inside a single job. Dates that do not include a specific time are automatically considered to use the current locale's time zone.
      Select the ISO template to be used when parsing the timestamp. Other ISO8601 formats can be handled via the Custom option.
    - **Custom**: Set a custom date format. For the markers available in the DataMapper see "Date" on page 439.
- **Date Language**: Set the language for a date value (ex: If English is selected, the term May will be identified as the month of May).
- **Use offset from UTC**: Select the default time zone, which is to be used to extract any timestamp that does not already include time zone information with the time.

Note: Default data formats tell the DataMapper how certain types of data are formatted **in the data source**. They don't determine how these data are formatted in the Data Model or in a template. In the Data Model, data are converted to the native data type. Dates, for example, are converted to a DateTime object in the Data Model, and will always be shown as "year-month-day" plus the time stamp, for example: 2012-04-11 12.00 AM.

Input Data settings

The **Input Data** settings specify how the input data must be interpreted. These settings are different for each data type. For a CSV file, for example, it is important to specify the delimiter that separates data fields. PDF files are already delimited naturally by pages, so the input data settings for PDF files are interpretation settings for text in the file.

- "Input Data settings" above
- "Excel file Input Data settings" on the next page
- "PDF file Input Data settings" on page 474
- "Database Input Data settings" on page 476
- "Text file Input Data settings" on page 477

- "XML File Input Data settings" on page 478
- "JSON File Input Data settings" on page 480

CSV file Input Data settings

In a CSV file, data is read line by line, where each line can contain multiple fields. The input data settings specify to the DataMapper module how the fields are separated.

- **Field separator**: Defines what character separates each field in the file. Even though CSV stands for comma-separated values, CSV can actually refer to files where fields are separated using any character, including commas, tabs, semicolons, and pipes.
- **Text delimiter**: Defines what character surrounds text in the file, preventing the **Field separator** from being interpreted within those text delimiters. This ensures that, for example, the field “Smith; John” is not interpreted as two fields, even if the field delimiter is the semicolon.
- **Comment delimiter**: Defines what character starts a comment line.
- **Encoding**: Defines what encoding is used to read the Data Source (US-ASCII, ISO-8859-1, UTF-8, UTF-16, UTF-16BE or UTF-16LE).
- **Lines to skip**: Defines a number of lines in the CSV that will be skipped and not used as records.
- **Set tabs as a field separator**: Overwrites the **Field separator** option and sets the Tab character instead for tab-delimited files.
- **First row contains field names**: Uses the first line of the CSV as headers, which automatically names all extracted fields.
- **Ignore unparseable lines**: Ignores any line that does not correspond to the settings above.
- **Skip empty lines**: Ignore any line that has no content. Note that spaces are considered content.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

Excel file Input Data settings

There are no settings for field separation in an Excel file, only settings with regards to the file as a whole.

- **Lines to skip**: Defines a number of lines in the Excel file that will be skipped and not used as records.
- **First row contains field names**: Check this option to use the first line of the Excel file as headers. This option automatically names all extracted fields.
- **Sheet**: Only one sheet can be selected as the data source. Enter a sheet name, or click the Dynamic Sheet Name icon to compose a dynamic sheet name that will be populated at runtime. A dynamic sheet name starts with an equals sign (=), and any variable values (i.e. job info variables and runtime parameters, see "Properties and runtime parameters" on page 387) must be enclosed within curly brackets: {}.

  Example: =sheet{automation.parameters.ID}
  This expression resolves to "sheet1" if the runtime parameter ID is set to 1.

  Tip: Double-click a parameter to add it to the expression.

  If an expression resolves to a pure integer (e.g. 1, 19, 42) the DataMapper will attempt to open a sheet whose name is exactly that (e.g. "1", "19", "42"). If the sheet with that name does not exist, the DataMapper attempts to open the sheet whose **index** matches the integer (e.g. the first, 19th or 42nd sheet).
- **Skip empty lines**: Ignore any line that has no content. Note that spaces are considered content.
  - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

PDF file Input Data settings

These settings also apply to files that are converted to PDF by the DataMapper: PS, PCL and AFP files.

PDF Files have a natural, static delimiter in the form of pages, so the options here are interpretation settings for text in the PDF file.
The Input Data settings for PDF files determine how words, lines and paragraphs are detected in the PDF when creating data selections.
Each value represents a fraction of the average font size of text in a data selection, meaning "0.3" represents 30% of the height or width.

- **Word spacing**: Determines the spacing between words. As PDF text spacing is somehow done through positioning instead of actual text spaces, text position is what is used to find new words. This option determines what percentage of the average width of a single character needs to be empty to consider a new word has started. The default value is 0.3, meaning a space is assumed if there is a blank area of 30% of the width of the average character in the font.
- **Line spacing**: Determines the spacing between lines of text. The default value is 1, meaning the space between lines must be equal to at least the average character height.
- **Paragraph spacing**: Determines the spacing between paragraphs. The default value is 1.5, meaning the space between paragraphs must be equal to at least 1.5 times the average character height to start a new paragraph.
- **Magic number**: Determines the tolerance factor for all of the above values. The tolerance is meant to avoid rounding errors. If two values are more than 70% away from each other, they are considered distinct; otherwise they are the same. For example, if two characters have a space of exactly the width of the average character, any space of between 0.7 and 1.43 of this average width is considered one space. A space of 1.44 is considered to be 2 spaces.
- **PDF file color space**: Determines if the PDF if displayed in Color or Monochrome in the Data Viewer. Monochrome display is faster in the Data Viewer. This has no influence on the actual data extraction or the data mapping performance.
- **Processing method**: This option determines which search method is used by the Goto step's **Next Occurrence of** option. The most recent method searches for text within the specified constraints. It is more precise and more reliable than the **Original** method which searches for the target text in the entire page and then determines if the text appears within the specified constraints.
  Both methods may sometimes return slightly different values. Data mapping configurations made with previous version of the software are therefore set to use the original method by default. It is however recommended to modify them to use the newer method.
- **Auto-Rotation** (PS files only): If in the original PostScript file the orientation of a page and its text don't match, this option sets the page's orientation to match the orientation of the text.

Database Input Data settings

Databases all return the same type of information. Therefore the Input Data options for a database refer to the database itself instead of to the data.
The following settings apply to any database or ODBC Data Sample.

- **Connection String**: Displays the connection string used to access the Data Source.
- **Browse** button: Opens the **Edit Database configuration** dialog, which can replace the existing database data source with a new one. This is the same as using the **Replace** feature in the Data Samples window.
- **Table**: Displays the tables and stored procedures available in the database. The selected table is the one the data is extracted from. Clicking on any of the tables shows the first line of the data in that table.
- **Custom SQL** button: Click to open the SQL Query Designer (see"SQL Query Designer" on page 485) and type in a custom SQL query. If the database supports stored procedures, including inner joins, grouping and sorting, you can use custom SQL to make a selection from the database, using whatever language the database supports.
  The query may contain variables and properties, so that the selection will be dynamically adjusted each time the data mapping configuration is actually used in a Workflow process; see"Using variables and properties in an SQL query" on page 486.
- **Encoding**: Defines what encoding is used to read the Data Source (US-ASCII, ISO-8859-1, UTF-8, UTF-16, UTF-16BE or UTF-16LE).
- **Skip empty lines**: Ignore any row that has no content, e.g. only nulls or empty strings. Note that spaces are characters and thus considered content.
- **Sort on**: Allows to select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the sorting column has numbers, it will be sorted as a text.
  With a Custom Query, this option is not available.
- **Use previous JDBC-ODBC driver (not recommended)**: The JDBC-ODBC driver that was used in OL Connect versions before 2024.2 is deprecated and has been replaced with a new JDBC-ODBC driver. The new driver is used by default with both new and existing data mapping configurations. However, this option allows the old driver to be used if necessary. The functionality of the old driver remains unchanged.

Text file Input Data settings

Because text files have many different shapes and sizes, there are many options for the input data in these files.

- **Encoding**: Defines what encoding is used to read the Data Source (US-ASCII, ISO-8859-1, UTF-8, UTF-16, UTF-16BE or UTF-16LE).
- **Selection/Text is based on bytes**: Check for text files that use double-bytes characters (resolves width issues in some text files).
- **Add/Remove characters**: Defines the number of characters to add to, or remove from, the head of the data stream. The spin buttons can also increment or decrement the value. Positive values add blank characters while negative values remove characters.
- **Add/Remove lines**: Defines the number of lines to add to, or remove from, the head of the data stream. The spin buttons can also increment or decrement the value. Positive values add blank lines while negative values remove lines.
- **Maximum line length**: Any line that is longer than the given maximum line length will be split at the maximum line length, as often as necessary. This option is used to cut (and wrap) long lines into logical blocks of data.
  The maximum value for this option is 65,535 characters. The default value is 80 characters.
- **Page delimiter type**: Defines the delimiter between each page of data. Multiples of such pages can be part of a record, as defined by the **Boundaries**.
  - **On lines**: Triggers a new page in the Data Sample after a number of lines.
    - **Cut on number of lines**: Triggers a new page after the given number of lines. With this number set to 1, and the Boundaries set to On delimiter, it is possible to create a record for each and every line in the file.
    - **Cut on FF**: Triggers a new page after a Form Feed character.
  - **On text**: Triggers a new page in the Data Sample when a specific string is found in a certain location.
    - **Word to find**: Compares the text value with the value in the data source.
    - **Match case**: Activates a case sensitive text comparison.
    - **Location**: Choose **Selected area** or **Entire width** to use the value of the current data selection as the text value.

- **Left/Right**: Use the spin buttons to set the start and stop columns to the current data selection (**Selected area**) in the record.
- **Lines before/after**: This option places the delimiter a certain number of lines before or after the current line. This is useful if the text that triggers the delimiter is not on the first line of each page.

- **Text from right to left**: Sets the writing direction of the data source to right-to-left.
- **Expand tabs to spaces**: Replaces tabs with the given number of spaces.
- **Ignore CR/LF/FF at end of file**: Instructs the DataMapper to ignore any CR, LF, FF or CR/LF characters when they are the last characters in a file. This prevents the addition of an unintended trailing record when the data mapping configuration is set up to cut on every line.

XML File Input Data settings

For an XML file you can either choose to use the root node, or select an element type, to create a new record every time that element is encountered.

Note that the data file is not split. The XML tree is scanned from top to bottom (node by node), and when an element matches the record definition condition, the parsing mechanism stops. All parent or sibling nodes found before that element are displayed as part of the record. Any nodes located after the record definition node are not displayed (unless the Show all elements option is enabled). Parsing then starts again, at the top, and looks for the next element that matches the record definition condition, ignoring elements for which a record has already been created.

- **Use root element**: Selects the top-level element. No other boundaries can be set. If there is only one top-level element, there will only be one record.

  This setting is not recommended for XML data files is larger than 5 - 10MB. For files this large, it would be better to implement a Pre-processor script to pre-process the XML data file, or even convert it into CSV format.

- **Use specific element**: Displays a list containing all the elements in the XML file. Selecting an element causes a new page of data to be created every time an instance of this element is encountered.
  Note that higher-level nodes above the selected element are shown, but those below the selected element are not displayed unless the **Show all elements** option is checked.

- **Use XPath**: Enter an XPath to create records based on the **node name** of elements. For example: `./*[starts-with(name(),'inv')]` will create a record for every element of which the name starts with 'inv'.
  Note that `starts-with()` is an XPath function. For an overview of XPath functions, see Mozilla: XPath Functions.
  The XPath may also contain JavaScript code. Note that since the XPath is a string, the return value of the JavaScript statement will be interpreted as a string.
  In order to use JavaScript:
  - The XPath must start with **=**
  - The entire JavaScript statement must be enclosed in curly brackets: **{**...**}**
  - Any other curly brackets that are not part of the JavaScript code must be escaped with a backslash.
  - Single line comments (//...) in the code are not supported.

  Note: Currently, XPaths that select elements based on an attribute, attribute value, node value, node counter or node index are not supported.
- **Show all elements**: When the delimiter is set to a specific element or XPath, checking this option allows to extract information from higher-level nodes, including those that follow the element or path. This might slow down the processing, so if you don't need any information from the higher-level nodes that follow that specific element, it is recommended to leave this option unchecked.

  When this option is used in combination with a trigger element that is not repeated at the same node level (in other words, it doesn't have a sibling with the same name), the entire XML document will be shown for each record, except the trigger element, which will only be shown for the record that is currently selected in the Data Model pane. This could lead to a problem with some steps that use an XPath with absolute indexes, such as a location-based Extract step. Using a dynamic index in the XPath will fix the problem. For example, in the case of a location-based Extract step, switch to extracting the data via JavaScript (see "Expression-based field" on page 425); in the JavaScript expression, replace the index of the element's parent in the XPath with record.index.

Note: The information contained in all of the selected parent nodes will be copied for each instance of that node. For example, if a client node contains multiple invoice nodes, the information for the client node can be duplicated for each invoice.

The DataMapper only extracts elements for which at least one value or attribute value is defined in the file.

JSON File Input Data settings

For a JSON file you can either use the object or array at the root and get one output record, or select an object or array as **parent element**. Its direct child elements - objects and arrays, *not* key-value pairs - can be output as individual records.

- **Use root element**: Selects the top-level array or object. There will only be one record.
- **Use specific element**: Select an array **[ ]** or object **{ }** in the JSON data as **Parent element** to define its child elements - objects and/or arrays - as source records. Any elements outside the parent element and key-value pairs inside the parent will be repeated in each source record.

  Note: Only arrays and objects can be seen as a record. It is not possible to split the JSON between key-value pairs.

Boundaries settings

Boundaries are the division between **records**: they define where one record ends and the next record begins; for an explanation see "Record boundaries" on page 373.

- "CSV, Excel or Database file boundaries" below
- "PDF file boundaries" on the next page
- "Text file boundaries" on page 482
- "XML file boundaries" on page 484
- "JSON file boundaries" on page 484

CSV, Excel or Database file boundaries

Since database data sources are structured the same way as CSV and Excel files, the options for these file types are identical.

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Line limit**: Defines the limit of detail lines in any detail table. This is useful for files with a high number of detail lines, which in the DataMapper interface can slow down

things. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).

- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
  - **Record(s) per page**: Defines a fixed number of lines in the file that go in each record.
    - **Records**: The number of records (lines, rows) to put in each record.
  - **On change**: Defines a new record when a specific field (**Field name**) has a new value.
    - **Field name**: Displays the fields in the top line. The boundaries are set on the selected field name.
  - **On script:** Defines the boundaries using a custom JavaScript. For more information see "Setting boundaries using JavaScript" on page 535.
  - **On field value**: Sets a boundary on a specific field value.
    - **Field name**: Displays the fields in the top line. The value of the selected field is compared with the **Expression** below to create a new boundary.
    - **Expression**: Enter the value or **Regular Expression** to compare the field value to.
    - **Use Regular Expression**: Treats the **Expression** as a regular expression instead of static text. For more information on using **Regular Expressions** (regex), see the Regular-Expressions.info Tutorial.

PDF file boundaries

For a PDF file, Boundaries determine how many pages are included in each record. You can set this up in one of three ways: by giving a static number of pages; by checking a specific area on each page for text changes, specific text, or the absence of text; or by using an advanced script.

- **Record limit**: Defines how many records are displayed in the Data Viewer. To disable the limit, use the value 0 (zero).
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
  - **On page**: Defines a boundary on a static number of pages.
    - **Number of pages**: Defines how many pages go in each record.

- **On text**: Defines a boundary on a specific text comparison.
  - **Start coordinates (x,y)**: Defines the left and top coordinates of the data selection to compare with the text value.
  - **Stop coordinates (x,y)**: Defines the right and bottom coordinates.
  - **Use Selection**: Select an area in the Data Viewer and click the **Use selection** button to set the start and stop coordinates to the current data selection.

    Note: In a PDF file, all coordinates are in millimeters.
  - **Times condition found**: When the boundaries are based on the presence of specific text, you can specify after how many instances of this text the boundary can be effectively defined. For example, if a string is always found on the first and on the last page of a document, you could specify a number of occurrences of 2. This way, there is no need to inspect other items for whether it is on the first page or the last page. Having found the string two times is enough to set the boundary.
  - **Pages before/after**: Defines the boundary a certain number of pages before (-) or after (+) the current page. This is useful if the text triggering the boundary is not located on the first page of the record.
  - **Operator**: Selects the type of comparison (for example, "contains").
  - **Word to find**: Compares the text value with the value in the data source.
  - **Match case**: Makes the text comparison case sensitive.
- **On script:** Defines the boundaries using a custom JavaScript. For more information see "Setting boundaries using JavaScript" on page 535.
- **On all pages**: Sets a boundary after the last page, creating one source record.

Text file boundaries

For a text file, Boundaries determine how many 'data pages' are included in each record. These don't have to be actual pages, as is the case with PDF files. The data page delimiters are set in the "Text file boundaries" above.

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Selection/Text is based on bytes**: Select this option for text records with fixed width fields whose length is based on the number of bytes and not the number of

characters.

- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
    - **On delimiter:**Defines a boundary on a static number of pages.
        - **Occurrences**: The number of times that the delimiter is encountered before fixing the boundary. For example, if you know that your documents always have four pages delimited by the FF character, you can set the boundaries after every four delimiters.
    - **On text**: Defines a boundary on a specific text comparison.
        - **Location**:
            - **Selected area**:
                - **Select the area**button: Uses the value of the current data selection as the text value. Making a new selection and clicking on Select the area will redefine the location.
                - **Left/Right**: Defines where to find the text value in the row.
                - **Top/Bottom**: Defines the start and end row of the data selection to compare with the text value.
            - **Entire width**: Ignores the column values and compares using the whole line.
            - **Entire height**: Ignores the row values and compares using the whole column.
            - **Entire page**: Compares the text value on the whole page. Only available with`contains`,`not contains`,`is empty`and`is not empty`operators.
        - **Times condition found**: When the boundaries are based on the presence of specific text, you can specify after how many instances of this text the boundary can be effectively defined. For example, if a string is always found on the first and on the last page of a document, you could specify a number of occurrences of 2. This way, there is no need to inspect other items for whether it is on the first page or the last page. Having found the string two times is enough to set the boundary.
        - **Delimiters before/after**: Defines the boundary a certain number of data

pages before or after the current data page. This is useful if the text triggering the boundary is not located on the first data page of the record.

- **Operator**: Selects the type of comparison (for example, "contains").
- **Word to find**: Compares the text value with the value in the data source.
- **Use selected text** button: copies the text in the current selection as the one to compare to it.
- **Match case**: Makes the text comparison case sensitive.

- **On script:** Defines the boundaries using a custom JavaScript. For more information see "Setting boundaries using JavaScript" on page 535.

XML file boundaries

The delimiter for an XML file is a node. The Boundaries determine how many of those nodes go in one record. This can be a specific number, or a variable number if the boundary is to be set when the content of a specific field or attribute within a node changes (for example when the invoice_number field changes in the invoice node).

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero).
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
  - **On Element**: Defines a new record on each new instance of the XML element selected in the Input Data settings.
    - **Occurrences**: The number of times that the element is encountered, as a direct descendant of the same parent element, before the boundary is set.
  - **On Change**: Defines a new record when a specific field or attribute in the XML element has a new value.
    - **Field**: Displays the fields and (optionally) attributes in the XML element. The value of the selected field determines the new boundaries.
    - **Also extract element attributes**: Check this option to include attribute values in the list of content items that can be used to trigger a boundary.

JSON file boundaries

The delimiter for a JSON file is an object or array inside the selected parent element (see "JSON file boundaries" above). The Boundaries determine how many of them go in one record.

Note: Only arrays and objects can be seen as a record. It is not possible to split the JSON between key-value pairs.

- **Record limit**: Defines how many records are displayed in the Data Viewer. This does not affect output production; when generating output, this option is ignored. To disable the limit, use the value 0 (zero). The default value is 200.
- **Trigger**: Defines the type of rule that controls when a boundary is set, creating a new record.
    - **On element**: Creates a new record in the output for each object or array, or - if you set a higher number of occurrences - after every *n*-th object or array in the parent element.
        - **Occurrences**: The number of times that an element is encountered in the parent element before fixing the boundary.
    - **On change**: Creates a new record each time the value in a certain key-value pair changes.
        - **Field**: Displays the keys of key-value pairs that exist at the root of direct child elements of the selected parent element. The value of the selected field determines the new boundaries.

SQL Query Designer

The SQL Query Designer is used to design a custom SQL query to pull information from a database. It can be opened by clicking the **Custom SQL** button on the Settings pane, under Input Data.

- **Tables**: Lists all tables and stored queries in the database.
- **Custom Query**: Displays the query that retrieves information from a database. You may use variables and properties in the query, to make the selection dynamic. See "Using variables and properties in an SQL query" on the facing page.
  Each database type has their own version of the SQL query language. To learn how to build your own query, please refer to your database's user manual.
- **Test Query** button : Click to test the custom query to ensure it will retrieve the appropriate information.
- **Results**: Displays the result of the SQL query when clicking on **Test Query**.

Using variables and properties in an SQL query

When you use variables and properties in an SQL query, the selection will be dynamically adjusted each time the data mapping configuration is actually used in a Workflow process.

To create a dynamic SQL query:

- The query must start with **=**
- Any variable or property must be enclosed in curly brackets: **{** ... **}**. This effectively inserts a JavaScript statement in the query. Note that all other curly brackets must be escaped with a backslash.

Inside the brackets you may enter any of the following property fields defined in the Preprocessor step (see "Fixed automation properties" on page 489 and "Properties" on page 490):

- Fixed automation properties. These are retrieved via the `automation` object (see "Objects" on page 540), for example`automation.jobInfo.JobInfo9` or `automation.properties.OriginalFilename`.
- Properties that have their scope set to "Entire data". These are retrieved via the `data` object (see "data" on page 546), for example:`data.properties.myProperty`.
- Properties that have their scope set to "Automation variable". These are retrieved via `automation.variables`(see "Objects" on page 540), for example `automation.variables.FieldList`.

Properties that have their scope set to "Each record" and **can not** be used because the SQL query is executed before any record is created. For the same reason, variables declared in other Steps can not be used.

Example

```
= SELECT {automation.variables.FieldList} FROM
{automation.jobInfo.JobInfo9}
```

If the Workflow variable defined as FieldList contains the value "id,name" and Job Info 9 contains the value "MyTable", then this custom query, once parsed, yields the following SQL statement: `SELECT id,name FROM MyTable`, which is then executed.

Steps pane

The Steps tab displays the data mapping workflow: the process that prepares and extracts data. The process contains multiple distinct steps and is run for each of the records in the

source data. For more information about the steps and how to use them, please refer to Steps and "Data mapping workflow" on page 368.

Finding a step

To find a certain step in a large data mapping workflow, start typing the **name** of the step in the **Find Step** field at the top of the Steps pane, and press Enter. Use the buttons next to the number of search results to navigate through the matching steps.
Note that the search is case-insensitive.

Moving a step

To rearrange steps, simply drag & drop them somewhere else on the colored line in the Steps pane.
Alternatively you may right-click on a step and select **Cut Step** or use the Cut button in the **Toolbar**. If the step is **Repeat** or **Condition**, all steps inside it will also be placed on the clipboard. To place the step at its destination, right-click any step and select **Paste Step**, or use the Paste button in the toolbar. The pasted steps will be positioned below the selected step.

Viewing step details

Hovering over the task shows a tooltip that displays some of the details of that step. To see all details for a step, click on the step and take a look at the Step properties pane ("Step properties pane" on the facing page).
Clicking on any Extract step in the Steps pane highlights any area in the Data Viewer from which it extracts data.
You can also click on the Preprocessor step to select all the steps in the workflow to show a complete map of all the extracted data.

Window controls

The following controls appear at the top of the Steps pane:

- **Zoom In (CTRL +)**: Click to zoom in by increments of 10%
- **Zoom Out (CTRL -)**: Click to zoom out by increments of 10%

Contextual menu

You can access the contextual menu using a right-click anywhere inside the **Steps** pane.

- **Add a Step**: Adds a step to the process. More options are available when a **Repeat** or a **Condition** step is selected:
  - **Add Step in Repeat**: Adds a step to a **Repeat** loop.
  - **Add Step in True**: Adds a step to the True branch of a **Condition** step.
  - **Add Step in False**: Adds a step under the False branch of a **Condition** step.
  - **Add Multiple Conditions Step**: Adds a **Multiple Conditions** step.
  - **Add Case Step**: Adds a Case condition under the selected **Multiple Conditions** step.
- **Ignore Step/Reactivate Step**: Click to set the step to be ignored (aka disabled) or to reactivate it. Disabled steps are grayed and do not run, neither in the DataMapper nor when the data mapping configuration is executed in Workflow. However, they can still be modified normally.
- **Delete Step**: To remove a step, right-click on it and select **Delete step** from the contextual menu or use the Delete button in the **Toolbar**. If the step to be deleted is **Repeat** or **Condition**, all steps under it will also be deleted.
- **Copy/Paste Step**: To copy a step, right-click on it and select **Copy Step** or use the Copy step button in the Toolbar. If the step is **Repeat** or **Condition**, all steps under it will also be placed in the clipboard. To paste the copied step at its destination, right-click the step in the position before the desired location and select **Paste Step**, or use the Paste step button in the **Toolbar**.

Step properties pane

The Step Properties pane is used to adjust the properties of each Step in the process (see "Steps" on page 409).
The pane is divided in a few subsections depending on the Step and the data type. It always contains a subsection to name and document the selected Step. Other subsections allow you to edit or delete fields that belong to the Step (see "Fields" on page 425) or change the expected data format (see "Data Format" on page 498), for example.

Note: Step properties may also depend on the data sample's file type.

- "Preprocessor step properties" on the next page
- "Extract step properties" on page 492
- "Action step properties" on page 499

- "Repeat step properties" on page 507
- "Condition step properties" on page 515
- "Multiple Conditions step properties" on page 518
- "Goto step properties" on page 521
- "Postprocessor step properties" on page 525

Preprocessor step properties

The **Preprocessor** step does not run for every record in the source data. It runs once, at the beginning of the extraction workflow, before anything else; see "Preprocessor step" on page 410.

The properties described below become visible in the Step properties pane when the Preprocessor step is selected in the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Fixed automation properties

The **Fixed automation properties** subsection lists all the fixed **runtime parameters** available from OL Connect Workflow. These properties are equivalent to data within the OL Connect Workflow process in which the data mapping configuration is applied. They are fixed in the sense that the name of the property and the source of its value cannot be changed.

See also: "Properties and runtime parameters" on page 387.

For each property, the following is available:

- **Name**: A read-only field displaying the name of the property.
- **Scope**: A read-only field indicating that the scope of the property is **Runtime parameter**.
- **Type**: A read-only field indicating the data type for each property.

- **Debug Value**: Enter a debug value for the property. This value is overwritten by the actual value coming from OL Connect Workflow when the data mapping configuration is run using the **Execute Data Mapping** task.

The fixed runtime parameters are:

- **JobInfoX**: These properties are the equivalent of the Job Info values available in the OL Connect Workflow process see Job Info variables in the Workflow Online Help). To access these properties inside of any JavaScript code within the data mapping configuration, use the `automation.jobInfos.JobInfoX` (where X is the Job Info number, from 0 to 9).
- **OriginalFilename**: At **runtime**, this property contains the original file name that was captured by the OL Connect Workflow process and is equivalent to the `%o` variable in the process. To access this property inside of any JavaScript code within the data mapping configuration, use `automation.properties.OriginalFilename`.
  At **design** time, this property is set to the name of the current **sample data file**.
- **ProcessName**: This property contains the name of the process that is currently executing the data mapping configuration and is equivalent to the `%w` variable in the process. To access this property inside of any JavaScript code within the data mapping configuration, use `automation.properties.ProcessName`.
- **TaskIndex**: This property contains the index (position) of the task inside the process that is currently executing the data mapping configuration but it has no equivalent in OL Connect Workflow. To access this property inside of any JavaScript code within the data mapping configuration, use `automation.properties.ProcessName`.

In scripts, fixed automation properties are retrieved via the automation object (see "Objects" on page 540), for example `automation.jobInfo.JobInfo9` or `automation.properties.OriginalFilename`.

Note: Other variables used in a Workflow configuration are made available to the data mapping configuration via custom runtime parameters, defined in the **Properties** subsection of the Preprocessor step (see below).

Properties

The **Properties** subsection is used to create specific properties that are used throughout the workflow.

For more information see: "Properties and runtime parameters" on page 387.

Note: Properties are evaluated in the order they are placed in the list, so properties can use the values of previously defined properties in their expression.

- **Name**: The name of the property used to refer to its value.
- **Scope**: The scope of a property determines when the property is set and how it can be accessed (see "Accessing properties and runtime parameters" on page 389).
    - **Entire Data**: These properties are statically set at the start of a job, before anything else. They cannot be changed once they have been set, in other words they are **global constants**. They are mostly useful for static information such as folder locations or server addresses.
    - **Each Record**: These properties are evaluated and set at the beginning of each source record. Once they have been set, these properties can be modified via an Action step (see "Action step" on page 417), but they are always reset at the beginning of each source record.
- **Type**: The data type of the property. For more information see "Data types" on page 436.
- **Debug Value**: The initial value of the property. This is a JavaScript expression. See "DataMapper Scripts API" on page 531.

  Note: Since **Entire data** properties are evaluated before anything else, such as **Preprocessors**, **Delimiters** and **Boundaries** in the Settings pane (see "Data source settings" on page 370), these properties cannot read information from the data sample or from any records.

Actions

The **Actions** subsection defines what preprocessor tasks are performed on the data file before it is handed over to the data mapping workflow. Preprocessor actions can modify the data file in many ways, and each action is performed in turn, using the result of the previous one as an input.

- **Name**: The name to identify the preprocessor action. Click this field to rename the preprocessor action.
- **Type**: The type of preprocessor action. Currently there is only one type available: script.

The buttons to the right can be used to add, remove and reorder the tasks and to apply them to the current data file.

Note: The **Apply** button has an effect at design time only. It is ignored at runtime. When a data mapping configuration is executed on the server, all preprocessor actions are executed in original order.

The **Export** button exports the current **data file** in its current state. Preprocessing may cost a lot of time when the data file is very large. You could use the exported file instead of the original file to avoid preprocessing while developing the Workflow configuration.

Preprocessor definition

- **Expression**: Enter the JavaScript code to be performed on the data file. See "DataMapper Scripts API" on page 531.

Extract step properties

The **Extract** step takes information from the data source and places it in the record set that is the result of the extraction workflow. For more information see "Extract step" on page 411 and "Extracting data" on page 390.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Extraction Definition

- **Data Table**: Defines where the data will be placed in the extracted record. The root table is record, any other table inside the record is a detail table. For more information see "Extracting transactional data" on page 396.
- **Append values to current record**: When the **Extract** step is inside a loop, check this to ensure that the extraction will be done in the same **detail table** as any previous extractions within the same loop. This ensures that, if multiple extracts are present, only one detail table is created.

Field Definition

The following field definition settings are identical for all fields.

- **Field List**: The **Field List** displays each of the single fields that belong to the selected step in a drop-down. Fields can be re-ordered and re-named within the Order and rename fields dialog (see "Order and rename fields dialog" on page 499). Select one of the fields to make further settings for that field.

  > **Tip:** To change the name of a field quickly, right-click it in the Data Model and select **Rename**.

- **Add Unique ID to extraction field**: Check to have a unique identifier (a random set of numerical characters) added to the end of the extracted value if it is the same as a previously extracted value. For instance, if an extracted field named "FullName" contains the value "John.Smith" and another document record, later, also contains the same value, then this second "John.Smith" will have a unique identifier (e.g. "John.Smith187").

  This option is only available for String type data fields.

- **Mode**: Determines the origin of the data. Fields always belong to an Extract step, but they don't necessarily contain extracted data. See "Fields" on page 425 for more information.

  - **Location**: The contents of the data selection determine the value of the extracted field. The settings for location-based fields are listed separately, per file type:

    - "Settings for location-based fields in a Text file" on the facing page
    - "Settings for location-based fields in a PDF file" on page 495
    - "Settings for location-based fields in CSV and Database files" on page 496
    - "Settings for location-based fields in an XML file" on page 496
    - "Settings for location-based fields in a JSON file" on page 497

  - **JavaScript**: The result of the JavaScript Expression written below the drop-down will be the value of the extracted field. If the expression contains multiple lines, the last value attribution (variable = "value";) will be the value. See "DataMapper Scripts API" on page 531.

    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript Expression. If multiple lines or elements are selected, only the first one is used.

- **Use selection**: Click to use the value of the current data selection for the extraction.

  > **Note:** If the selection contains multiple lines, only the first line is selected.

- **Properties**: The value of the property selected below will be the value of the selected field.
  - **Property**: This drop-down lists all the currently defined properties (including system properties).
    Custom properties can be defined in the Preprocessor step; see "Preprocessor step" on page 410.
    For an explanation of the objects to which the properties belong, see "DataMapper Scripts API" on page 531.
  - **Choose a property** button: Click this button to open a filter dialog that lets you find a property based on the first few letters that you type.
- **Type**: The data type of the selected data; see "Data types" on page 436. Make sure that the data format that the DataMapper expects matches the actual format of the data in the data source; see "Data Format" on page 498.

Settings for location-based fields in a Text file

- **Left**: Defines the start of the data selection to extract, counting the number of characters from the left. E.g. 1 means: start with the first character from the left.
- **Right**: Defines the end of the data selection to extract, counting the number of characters from the left. E.g. 3 means: stop after the third character from the left.
- **Top offset**: The vertical offset (the number of lines) from the current pointer location in the Data Viewer).
- **Height**: The height of the selection box, in the number of lines. When set to 0, this instructs the DataMapper to extract all lines starting from the given position until the end of the record and store them in a single field.
- **Use selection**: Click to use the value (Left, Right, Top offset and Height) of the current data selection (in the Data Viewer) for the extraction.

  > **Note:** If the selection contains multiple lines, only the first line is extracted.

- **Post Function**: Enter a JavaScript expression to be run after the extraction.

A Post function script operates directly on the extracted data, and its results replace the extracted data. For example, the Post function script `replace("-", "");` would replace the first dash character that occurs inside the extracted string.

- **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.

> **Tip:** Click and drag the icon at the bottom right of the Post function area to make it larger.

- **Trim**: Select to trim empty characters at the beginning or the end of the field.
- **Concatenation string**: The (HTML) string used to concatenate lines when they are joined.
- **Split**: Separate the selection into individual fields based on the **Concatenation string**defined above.

Settings for location-based fields in a PDF file

These are the settings for location-based fields in a PDF file.

- **Left**: Defines the start of the data selection to extract, counting the number of characters from the left. E.g. 1 means: start with the first character from the left.
- **Right**: Defines the end of the data selection to extract, counting the number of characters from the left. E.g. 3 means: stop after the third character from the left.
- **Top offset**: The vertical offset (the number of lines) from the current pointer location in the Data Sample (Viewer).
- **Height**: The height of the selection box, in the number of lines.
- **Use selection**: Click to use the value (Left, Right, Top offset and Height) of the current data selection for the extraction.

> **Note:** If the selection contains multiple lines, the lines are by default joined and extracted into one field. To split the lines, select the option Split lines (see below).

- **Post Function**: Enter a JavaScript expression to be run after the extraction. For example`replace("-","")` would replace a single dash character inside the extracted string.
- **Trim**: Select to trim empty characters at the beginning or the end of the field.
- **Type**: The data type of the selected data; see "Data types" on page 436. If the selected data is split (see below), this setting is applied to the first extracted field. Make sure

that the data format that the DataMapper expects matches the actual format of the data in the data source; see "Data Format" on page 498.

- **Split**:
  - **Split lines**: Separate a multi-line selection into individual fields .
  - **Join lines**: Join the lines in the selection with the Concatenation string defined below.
- **Concatenation string**: The (HTML) string used to concatenate lines when they are joined.

Settings for location-based fields in CSV and Database files

These are the settings for location-based fields in CSV and Database files.

- **Column**: Drop-down listing all fields in the Data Sample, of which the value will be used.
- **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
  - **Use selection**: Click to use the value of the current data selection for the extraction.

    Note: If the selection contains multiple lines, only the first line is selected.
- **Post Function**: Enter a JavaScript expression to be run after the extraction. For example`replace("-","")` would replace a single dash character inside the extracted string.
  - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
- **Trim**: Select to trim empty characters at the beginning or the end of the field.

Settings for location-based fields in an XML file

These are the settings for location-based fields in an XML file.

- **XPath**: The path to the XML field that is extracted.
  - **Use selection**: Click to use the value of the current data selection for the extraction.

    Note: If the selection contains multiple lines, only the first line is selected.

- **Post Function**: Enter a JavaScript expression to be run after the extraction. For example`replace("-","")` would replace a single dash character inside the extracted string.
    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
- **Trim**: Select to trim empty characters at the beginning or the end of the field.

Settings for location-based fields in a JSON file

These are the settings for location-based fields in a JSON file.

- **JsonPath**: The path to the JSON element that is extracted. The JsonPath can be relative or absolute. An absolute path starts with **$** (the root), a relative path starts with **.** (the current element). For an overview of the JsonPath syntax, see https://github.com/json-path/jsonpath.

  Note: A JsonPath expression can define more than one item (for example: **.*** returns anything in the current element). If more than one item is returned, the Extract step will keep an array of all returned items.

  The full JsonPath to an element is displayed at the bottom left of the window when you select it. To copy the path, right-click it and select Copy.

    - **Use selection**: Click to use the value of the current data selection for the extraction.

  Note: If a key in a JSON file has a name that looks like a function (e.g. "TLIST (A1)"), then the Extract step has to use a JsonPath with bracket notation instead of the default dot notation. For information about the bracket notation see https://goessner.net/articles/JsonPath/.

- **Post Function**: Enter a JavaScript expression to be run after the extraction. For example`replace("-","")` would replace a single dash character inside the extracted string.
    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
- **Trim**: Select to trim empty characters at the beginning or the end of the field.

Data Format

Format settings can be defined in three places: in the user preferences ("DataMapper preferences" on page 1039), the current data mapping configuration ("Data format settings" on page 373) and per field via the Step properties.

Any format settings specified per field are always used, regardless of the user preferences or data source settings.

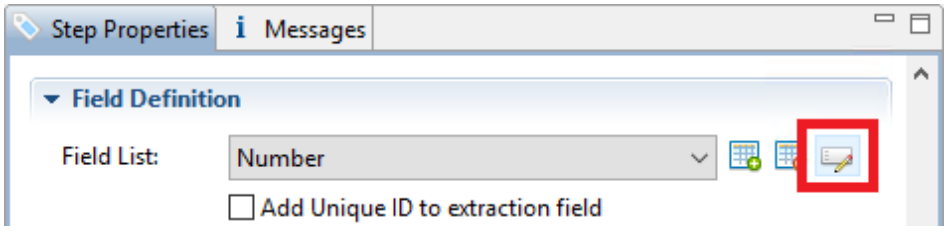
Note: Data format settings tell the DataMapper how to read and parse data **from the data source**. They don't determine how these data are formatted in the Data Model or in a template. In the Data Model, data are converted to the native data type. Dates, for example, are converted to a DateTime object. How they are displayed in the Data Model depends on the preferences (see "Default Format" on page 1039).

- **Negative Sign Before**: Any value in a numeric field that has a "-" sign is interpreted as a negative value.
- **Decimal Separator**: Set the decimal separator for a numerical value.
- **Thousand Separator**: Set the thousand separator for a numerical value.
- **Currency Sign**: Set the currency sign for a currency value.
- **Treat empty as 0**: A numerical empty value is treated as a 0 value.
- **Date/Time Format**: Set the date format for a date value.
  - **Automatic**: Select this option to parse dates automatically, without specifying a format. This is the default setting for new Date fields.
  - **ISO8601**: This setting allows for dates with different timestamp formats, or belonging to different time zones, to be parsed inside a single job. Dates that do not include a specific time are automatically considered to use the current locale's time zone.
    Select the ISO template to be used when parsing the timestamp. Other ISO8601 formats can be handled via the Custom option.
  - **Custom**: Set a custom date format. For the markers available in the DataMapper see "Date" on page 439.
- **Date Language**: Set the language for a date value (ex: If English is selected, the term May will be identified as the month of May).
- **Use offset from UTC**: Select the default time zone, which is to be used to extract any timestamp that does not already include time zone information with the time.

Order and rename fields dialog

The **Order and rename fields** dialog displays the extracted fields in the currently selected **Extract** step.
To open it, first select an Extract step on the Steps pane. Then, on the Step properties pane, under Field Definition, click the **Order and Rename Fields** button next to the Field List

drop-down.



Field extractions are executed from top to bottom.
In **JavaScript-based** fields, it is possible to refer to previously extracted fields if they are extracted higher in this list or in previous **Extract** steps in the extraction workflow.

- **Name**: The name of the field. Click the field name and enter a new name to rename the field.

  > **Note:** If you intend to use the field names as Metadata in a OL Connect Workflow process, do not add spaces to field names, as they are not permitted in Metadata field names.

- **Value**: Displays the value of the extract field in the current Record.
- **Remove** button: Click to remove the currently selected field.
- **Move Up** button: Click to move the selected field up one position.
- **Move Down** button: Click to move the selected field down one position.

> **Note:** The order of fields in an extraction step isn't necessarily the same as the order of those fields in the Data Model; see "Ordering and grouping fields in the Data Model" on page 422.

Action step properties

The **Action** step can run multiple specific actions one after the other in order; see "Action step" on page 417 for more information.

The properties of an Action step become visible in the Step properties pane when the Action step is selected on the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Actions

This subsection lists all actions executed by the step, and their types.

- **Name**: A name by which to refer to the action. This name has no impact on functionality.
- **Type**:
  - **Set property**: Sets the value of a record property which was created in the Preprocessor step (see "Preprocessor step" on page 410).
  - **Run JavaScript** : Runs a JavaScript expression, giving much more flexibility over the extraction process.
  - **Skip to next record**: When this option is selected, the extraction workflow stops processing the current record and moves on to the next one. The record remains visible, but with its data model pane greyed out, when it is being skipped as a visual clue to determine to which records the "Stop Processing Record" action is being applied.
    If fields were already extracted prior to encountering the Action step, then those fields are stored as usual.
    If no fields were extracted prior to encountering the Action step, then no trace of the record is saved in the database at run time.
  - **Stop data mapping**: The extraction workflow stops processing the data.
    If fields of the current record were already extracted prior to encountering the Action step, then those fields are stored as usual, but the rest of the data is skipped.
    If no fields were extracted prior to encountering the Action step, then no trace of the current record is saved in the database at run time.
  - **Break out of repeat loop**: The data mapping workflow moves to the first step following the Repeat loop, after executing any other actions defined in the same Action step. If this action is specified in an Action step that is not inside a Repeat loop, then it is ignored.

Set Property

Text and PDF Files

- **Property**: Displays a list of record properties set in the Preprocessor step (see "Preprocessor step" on page 410).
- **Type**: Displays the type of the property. This is a read-only field.
- **Based on**: Determines the origin of the data.
  - **Location**: The contents of the data selection set below will be the value of the extracted field. The data selection settings are different depending on the data sample type.
    - **Left**: Defines the start of the data selection to extract
    - **Right**: Defines the end of the data selection to extract
    - **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
    - **Height**: The height of the selection box.
    - **Use selection**: Click to use the value of the current data selection for the extraction.

      Note: If the selection contains multiple lines, only the first line is selected.

    - **Trim**: Select to trim empty characters at the beginning or the end of the field
  - **JavaScript** : The result of the JavaScript **Expression** written below the drop-down will be the value of the extracted field. If the expression contains multiple lines, the last value attribution (variable = "value";) will be the value. See "DataMapper Scripts API" on page 531.
    - **Expression**: The JavaScript expression to run.
    - **Use JavaScript Editor**: Click to display the Edit Script dialog (see "Using scripts in the DataMapper" on page 533 and "DataMapper Scripts API" on page 531).
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript Expression. If multiple lines or elements are selected, only the first one is used.

- **Use selection**: Click to use the value of the current data selection for the extraction.

  Note: If the selection contains multiple lines, only the first line is selected.

- **Data Format**: Data format settings tell the DataMapper how certain types of data are formatted **in the data source**. Make sure that this format matches the actual format of the data in the data source.
  - **Negative Sign Before**: Any value in a numeric field that has a "-" sign is interpreted as a negative value.
  - **Decimal Separator**: Set the decimal separator for a numerical value.
  - **Thousand Separator**: Set the thousand separator for a numerical value.
  - **Currency Sign**: Set the currency sign for a currency value.
  - **Treat empty as 0**: A numerical empty value is treated as a 0 value.
  - **Date/Time Format**: Set the date format for a date value.
    - **Automatic**: Select this option to parse dates automatically, without specifying a format. This is the default setting for new Date fields.
    - **ISO8601**: This setting allows for dates with different timestamp formats, or belonging to different time zones, to be parsed inside a single job. Dates that do not include a specific time are automatically considered to use the current locale's time zone.
      Select the ISO template to be used when parsing the timestamp. Other ISO8601 formats can be handled via the Custom option.
    - **Custom**: Set a custom date format. For the markers available in the DataMapper see "Date" on page 439.
  - **Date Language**: Set the language for a date value (ex: If English is selected, the term May will be identified as the month of May).
  - **Use offset from UTC**: Select the default time zone, which is to be used to extract any timestamp that does not already include time zone information with the time.

- **Property**: Displays a list of record properties set in the Preprocessor step (see "Preprocessor step" on page 410).
- **Type**: Displays the type of the property. Read only field.
- **Based on**: Determines the origin of the data.
    - **Location**: The contents of the data selection set below will be the value of the extracted field. The data selection settings are different depending on the data sample type.
        - **Column**: Drop-down listing all fields in the Data Sample, of which the value will be used.
        - **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
        - **Use selection**: Click to use the value of the current data selection for the extraction.

          Note: If the selection contains multiple lines, only the first line is selected.
        - **Trim**: Select to trim empty characters at the beginning or the end of the field
    - **JavaScript** : The result of the JavaScript **Expression** written below the drop-down will be the value of the extracted field. If the expression contains multiple lines, the last value attribution (variable = "value";) will be the value. See "DataMapper Scripts API" on page 531.
        - **Expression**: The JavaScript expression to run.
        - **Use JavaScript Editor**: Click to display the Edit Script dialog (see "Using scripts in the DataMapper" on page 533 and "DataMapper Scripts API" on page 531).
        - **Use selected text**: Inserts the text in the current data selection in the JavaScript Expression. If multiple lines or elements are selected, only the first one is used.
        - **Use selection**: Click to use the value of the current data selection for the extraction.

> **Note:** If the selection contains multiple lines, only the first line is selected.

- **Data Format**: Data format settings tell the DataMapper how certain types of data are formatted **in the data source**. Make sure that this format matches the actual format of the data in the data source.
  - **Negative Sign Before**: Any value in a numeric field that has a "-" sign is interpreted as a negative value.
  - **Decimal Separator**: Set the decimal separator for a numerical value.
  - **Thousand Separator**: Set the thousand separator for a numerical value.
  - **Currency Sign**: Set the currency sign for a currency value.
  - **Treat empty as 0**: A numerical empty value is treated as a 0 value.
  - **Date/Time Format**: Set the date format for a date value.
    - **Automatic**: Select this option to parse dates automatically, without specifying a format. This is the default setting for new Date fields.
    - **ISO8601**: This setting allows for dates with different timestamp formats, or belonging to different time zones, to be parsed inside a single job. Dates that do not include a specific time are automatically considered to use the current locale's time zone.
      Select the ISO template to be used when parsing the timestamp. Other ISO8601 formats can be handled via the Custom option.
    - **Custom**: Set a custom date format. For the markers available in the DataMapper see "Date" on page 439.
  - **Date Language**: Set the language for a date value (ex: If English is selected, the term May will be identified as the month of May).
  - **Use offset from UTC**: Select the default time zone, which is to be used to extract any timestamp that does not already include time zone information with the time.

XML and JSON files

- **Property**: Displays a list of Source Record properties set in the Preprocessor step (see "Preprocessor step" on page 410).
- **Type**: Displays the type of the Source Record property. Read only field.
- **Based on**: Determines the origin of the data.

- **Location**: The contents of the data selection set below will be the value of the extracted field. The data selection settings are different depending on the data sample type.
    - **XPath** / **JsonPath**: The path to the XML field / JSON element that is extracted.
      For an overview of XPath functions, see Mozilla: XPath Functions.
      For an overview of the JsonPath syntax, see https://github.com/json-path/jsonpath.
        - **Use selection**: Click to use the value of the current data selection for the extraction.

          > **Note:** If the selection contains multiple lines, only the first line is selected.

        - **Trim**: Select to trim empty characters at the beginning or the end of the field
- **JavaScript** : The result of the JavaScript **Expression** written below the drop-down will be the value of the extracted field. If the expression contains multiple lines, the last value attribution (variable = "value";) will be the value. See "DataMapper Scripts API" on page 531.
    - **Expression**: The JavaScript expression to run.
    - **Use JavaScript Editor**: Click to display the Edit Script dialog (see "Using scripts in the DataMapper" on page 533 and "DataMapper Scripts API" on page 531).
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript Expression. If multiple lines or elements are selected, only the first one is used.
    - **Use selection**: Click to use the value of the current data selection for the extraction.

      > **Note:** If the selection contains multiple lines, only the first line is selected.

- **Data Format**: Data format settings tell the DataMapper how certain types of data are formatted **in the data source**. Make sure that this format matches the actual format of the data in the data source.

- **Negative Sign Before**: Any value in a numeric field that has a "-" sign is interpreted as a negative value.
- **Decimal Separator**: Set the decimal separator for a numerical value.
- **Thousand Separator**: Set the thousand separator for a numerical value.
- **Currency Sign**: Set the currency sign for a currency value.
- **Treat empty as 0**: A numerical empty value is treated as a 0 value.
- **Date/Time Format**: Set the date format for a date value.
    - **Automatic**: Select this option to parse dates automatically, without specifying a format. This is the default setting for new Date fields.
    - **ISO8601**: This setting allows for dates with different timestamp formats, or belonging to different time zones, to be parsed inside a single job. Dates that do not include a specific time are automatically considered to use the current locale's time zone.
      Select the ISO template to be used when parsing the timestamp. Other ISO8601 formats can be handled via the Custom option.
    - **Custom**: Set a custom date format. For the markers available in the DataMapper see "Date" on page 439.
- **Date Language**: Set the language for a date value (ex: If English is selected, the term May will be identified as the month of May).
- **Use offset from UTC**: Select the default time zone, which is to be used to extract any timestamp that does not already include time zone information with the time.

Run JavaScript

Running a JavaScript expression offers many possibilities. The script could, for example, set record properties and field values using advanced expressions and complex mathematical operations and calculations.

- **Expression**: The JavaScript expression to run (see "DataMapper Scripts API" on page 531).
- **Use JavaScript Editor**: Click to display the Edit Script dialog.
- **Use selected text**: Inserts the text in the current data selection in the JavaScript Expression. If multiple lines or elements are selected, only the first one is used.

- **Use selection**: Click to use the value of the current data selection.

  Note: If the selection contains multiple lines, only the first line is selected.

Repeat step properties

The **Repeat** step adds a loop to the extraction workflow; see "Steps" on page 409 and "Extracting transactional data" on page 396.

The properties described below become visible in the Step properties pane when the Repeat step is selected in the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Repeat Definition

- **Repeat type**:
  - **While statement is true**: The loop executes while the statement below is true. The statement is evaluated before the loop so the loop may not run at all.
  - **Until statement is true**: The loop executes until the statement below is true. The statement is evaluated after the loop so the loop will always run at least once.
  - **Until no more elements (for Text, CSV, Database and PDF files only)**: The loop executes as long as there are elements left as selected below.
  - **For Each (for XML and JSON files only)**: The loop executes for all nodes (by default) or for selected nodes on the specified level. To have the loop execute only on selected nodes, you need to edit the Xpath/JsonPath in the Collection field (see below). If the path defines a rule that is true for more then one node/element, a collection of nodes/elements is returned and the For Each loop will go through all of them.

> **Note:** >When using a **For Each** loop in XML or JSON, it is not necessary to skip to the repeating node/element or to have a **Goto** step to jump to each sibling, as this loop takes care of it automatically.

- **Maximum iterations on each line**: Defines the maximum number of iterations occurring at the same position. This expression is evaluated once when entering the loop. The value returned by the expression must be an integer higher than 0.
  Note that in the event of an infinite loop, an error will only be raised after the set maximum number of iterations.
  - **Use JavaScript Editor**: Click to display the Edit Script dialog.
- **No Goto step required**: With **XML** and **JSON** data, this option prevents the Repeat step from automatically moving forward to the next sibling (with XML) or element (with JSON).
  With all other data types, it disables the DataMapper's infinite loop detection in case of a missing Goto step. It is meant to be used in situations that would otherwise require a dummy Goto step.
- **Collection** (only with **For Each**):
  - For **XML**: the **XPath** that specifies the level and (optionally) elements to select on that level. For an overview of XPath functions, see Mozilla: XPath Functions.
    To select elements you can either use static values, e.g. `./user [@lastname="Smith"]` or JavaScript statements, for example: `=./user [@lastname="{automation.jobInfo.JobInfo1}"]`.
    In order to use JavaScript: Note that since the XPath is a string, the return value of the JavaScript statement will be interpreted as a string.
  - For **JSON**: the **JsonPath** that specifies the level and (optionally) elements to select on that level. For an overview of the JsonPath syntax, see https://github.com/json-path/jsonpath.
    It is not possible to use Javascript in the JsonPath.

Rule Tree

The **Rule tree** subsection displays the full combination of rules (defined below under **Condition**) as a tree, which gives an overview of how the conditions work together as well as the result for each of these conditions for the current record or iteration.

Condition

First, the **Condition List** displays the conditions in list form, instead of the tree form above. Three buttons are available next to the list:

- **Add condition**: Click to create a new condition in the list. This will always branch the current condition as an "AND" operator.
- **Delete condition**: Delete the currently selected condition.
- To rename a **Condition**, double click on its name from the **Rule tree** subsection .

Conditions are made by comparison of two operands using a specific **Operator**.

> **Note:** Both the **Left** and **Right** operands have the same properties.

Text and PDF files

> **Note:** The Repeat Step expects lines of text in a PDF file to be horizontal (regardless of the orientation of the page). Vertical text will cause an error.

- **Based On**:
  - **Position**: The data in the specified position for the comparison.
    - **Left**: The start position for the data selection. Note that conditions are done on the current line, either the current cursor position, or the current line in a **Repeat** step.
    - **Right**: The end position for the data selection.
    - **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
    - **Height**: The height of the selection box.
    - **Use Selection**: Click to use the value of the current data selection for the extraction.
    - **Trim**: Select to trim empty characters at the beginning or the end of the field.
  - **Value**: A specified static text value.
    - **Value**: The text value to use in the comparison.
    - **Use selected text**: Uses the text in the current data selection as the **Value**. If multiple lines or elements are selected, only the first one is used.
  - **Field**: The contents of a specific field in the **Extracted Record**.
    - **Field**: The **Extracted Record** field to use in the comparison.

- **JavaScript** : The result of a JavaScript **Expression**.
    - **Expression**: The JavaScript line that is evaluated. Note that the last value attribution to a variable is the one used as a result of the expression.
    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript **Expression**. If multiple lines or elements are selected, only the first one is used.
- **Data Property**: The value of a data-level property set in the Preprocessor step (see "Preprocessor step" on page 410).
- **Record Property:** One of the local variables that you can create and that are reset for each document as opposed to data variables that are global because they are initialized only once at the beginning of each job.
- **Automation Property**: The current value of a Document-level property set in the Preprocessor step (see "Preprocessor step" on page 410).
- **Extractor Property**: The value of an internal extractor variable:
    - **Counter**: The value of the current counter iteration in a **Repeat** step.
    - **Vertical Position**: The current vertical position on the page, either in Measure (PDF) or Line (Text and CSV).

- **Operators**:
    - **is equal to**: The two specified value are identical for the condition to be **True**.
    - **contains**: The first specified value contains the second one for the condition to be **True**.
    - **is less than**: The first specified value is smaller, numerically, than the second value for the condition to be **True**.
    - **is greater than**: The first specified value is larger, numerically, than the second value for the condition to be **True**.
    - **is empty**: The first specified value is empty. With this operator, there is no second value.
    - **Invert condition**: Inverts the result of the condition. For instance, **is empty** becomes **is not empty**.

- **Based On**:
  - **Position**: The data in the specified position for the comparison.
    - **Column**: Drop-down listing all fields in the Data Sample, of which the value will be used.
    - **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
    - **Use Selection**: Click to use the value of the current data selection for the extraction.
    - **Trim**: Select to trim empty characters at the beginning or the end of the field.
  - **Value**: A specified static text value.
    - **Value**: The text value to use in the comparison.
    - **Use selected text**:Uses the text in the current data selection as the **Value**. If multiple lines or elements are selected, only the first one is used.
  - **Field**: The contents of a specific field in the **Extracted Record**.
    - **Field**: The **Extracted Record** field to use in the comparison.
  - **JavaScript** : The result of a JavaScript **Expression**.
    - **Expression**: The JavaScript line that is evaluated. Note that the last value attribution to a variable is the one used as a result of the expression.
    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript **Expression**. If multiple lines or elements are selected, only the first one is used.
  - **Data Property**: The value of a data-level property set in the **Preprocessor** step.
  - **Record Property:** One of the local variables that you can create and that are reset for each document as opposed to data variables that are global because they are initialized only once at the beginning of each job.
  - **Automation Property**: The current value of a Document-level property set in the **Preprocessor** step.
  - **Extractor Property**: The value of an internal extractor variable:

        - **Counter**: The value of the current counter iteration in a **Repeat** step.
        - **Vertical Position**: The current vertical position on the page, either in Measure (PDF) or Line (Text and CSV).
- **Operators**:
    - **is equal to**: The two specified value are identical for the condition to be **True**.
    - **contains**: The first specified value contains the second one for the condition to be **True**.
    - **is less than**: The first specified value is smaller, numerically, than the second value for the condition to be **True**.
    - **is greater than**: The first specified value is larger, numerically, than the second value for the condition to be **True**.
    - **is empty**: The first specified value is empty. With this operator, there is no second value.
    - **Invert condition**: Inverts the result of the condition. For instance, **is empty** becomes **is not empty**.

XML files

- **Based On**:
    - **Position**: The data in the specified position for the comparison.
        - **XPath**: The path to the XML field.
        - **Use Selection**: Click to use the value of the current data selection for the extraction.
        - **Trim**: Select to trim empty characters at the beginning or the end of the field.
    - **Value**: A specified static text value.
        - **Value**: The text value to use in the comparison.
        - **Use selected text**: Uses the text in the current data selection as the **Value**. If multiple lines or elements are selected, only the first one is used.
    - **Field**: The contents of a specific field in the **Extracted Record**.
        - **Field**: The **Extracted Record** field to use in the comparison.
    - **JavaScript** : The result of a JavaScript **Expression**.

    - **Expression**: The JavaScript line that is evaluated. Note that the last value attribution to a variable is the one used as a result of the expression.
    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript **Expression**. If multiple lines or elements are selected, only the first one is used.
- **Data Property**: The value of a data-level property set in the **Preprocessor** step.
- **Record Property:** One of the local variables that you can create and that are reset for each document as opposed to data variables that are global because they are initialized only once at the beginning of each job.
- **Automation Property**: The current value of a Document-level property set in the **Preprocessor** step.
- **Extractor Property**: The value of an internal extractor variable:
    - **Counter**: The value of the current counter iteration in a **Repeat** step.
    - **Vertical Position**: The current vertical position on the page, either in Measure (PDF) or Line (Text and CSV).
- **Operators**:
    - **is equal to**: The two specified values are identical for the condition to be **True**.
    - **contains**: The first specified value contains the second one for the condition to be **True**.
    - **is less than**: The first specified value is smaller, numerically, than the second value for the condition to be **True**.
    - **is greater than**: The first specified value is larger, numerically, than the second value for the condition to be **True**.
    - **is empty**: The first specified value is empty. With this operator, there is no second value.
    - **Invert condition**: Inverts the result of the condition. For instance, **is empty** becomes **is not empty**.

JSON files

- **Based On**:
  - **Position**: The data in the specified position for the comparison.
    - **JsonPath**: The path to the JSON element.
    - **Use Selection**: Click to use the value of the current data selection for the extraction.
    - **Trim**: Select to trim empty characters at the beginning or the end of the field.
  - **Value**: A specified static text value.
    - **Value**: The text value to use in the comparison.
    - **Use selected text**: Uses the text in the current data selection as the **Value**. If multiple lines or elements are selected, only the first one is used.
  - **Field**: The contents of a specific field in the **Extracted Record**.
    - **Field**: The **Extracted Record** field to use in the comparison.
  - **JavaScript** : The result of a JavaScript **Expression**.
    - **Expression**: The JavaScript line that is evaluated. Note that the last value attribution to a variable is the one used as a result of the expression.
    - **Use JavaScript Editor**: Click to display the "boundaries" on page 541 dialog.
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript **Expression**. If multiple lines or elements are selected, only the first one is used.
  - **Data Property**: The value of a data-level property set in the **Preprocessor** step.
  - **Record Property:** One of the local variables that you can create and that are reset for each document as opposed to data variables that are global because they are initialized only once at the beginning of each job.
  - **Automation Property**: The current value of a Document-level property set in the **Preprocessor** step.

- **Extractor Property**: The value of an internal extractor variable:
  - **Counter**: The value of the current counter iteration in a **Repeat** step.
  - **Vertical Position**: The current vertical position on the page, either in Measure (PDF) or Line (Text and CSV).

- **Operators**:
  - **is equal to**: The two specified values are identical for the condition to be **True**.
  - **contains**: The first specified value contains the second one for the condition to be **True**.
  - **is less than**: The first specified value is smaller, numerically, than the second value for the condition to be **True**.
  - **is greater than**: The first specified value is larger, numerically, than the second value for the condition to be **True**.
  - **is empty**: The first specified value is empty. With this operator, there is no second value.
  - **Invert condition**: Inverts the result of the condition. For instance, **is empty** becomes **is not empty**.

Condition step properties

A **Condition** step is used when the data extraction must be based on specific criteria. See "Condition step" on page 414 for more information.

The properties of a Condition step become visible in the Step properties pane when the Condition step is selected on the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Rule tree

The **Rule tree** subsection displays the full combination of rules (defined below under **Condition**) as a tree, which gives an overview of how the conditions work together as well as

the result for each of these conditions for the current record or iteration.

- To rename a rule, double click on its name from the **Rule tree** subsection.
- To change the way rules are combined, right-click "AND". Select OR or XOR instead. XOR means one or the other, but not both. (See XOR on Wikipedia.)

Condition

First, the **Condition List** displays the conditions in list form, instead of the tree form above. Three buttons are available next to the list:

- **Add condition**: Click to add a new rule. This will always branch the current condition as an "AND" operator.
- **Delete condition**: Delete the currently selected condition.

Conditions are made by comparison of two operands using a specific **Operator**.

> **Note:** Both the **Left** and **Right** operands have the same properties.

- **Based On**:
  - **Position**: The data in the specified position for the comparison.
    - **Left** (Txt and PDF only): The start position for the data selection. Note that conditions are done on the current line, either the current cursor position, or the current line in a **Repeat** step.
    - **Right** (Txt and PDF only): The end position for the data selection.
    - **Height** (Txt and PDF only): The height of the selection box.
    - **Column** (CSV and Database only): Drop-down listing all fields in the Data Sample, of which the value will be used.
    - **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
    - **XPath** (XML only): The path to the XML field that is extracted.
    - **JsonPath** (JSON only): The path to the JSON element that is extracted. (See: "JsonPath" on page 395.)
    - **Use Selection**: Click to use the value of the current data selection for the extraction.

        - **Trim**: Select to trim empty characters at the beginning or the end of the field.
    - **Value**: A specified static text value.
        - **Value**: The text value to use in the comparison.
        - **Use selected text**: Uses the text in the current data selection as the **Value**. If multiple lines or elements are selected, only the first one is used.
    - **Field**: The contents of a specific field in the **Extracted Record**.
        - **Field**: The **Extracted Record** field to use in the comparison.
    - **JavaScript**: The result of a JavaScript **Expression**.
        - **Expression**: The JavaScript line that is evaluated. Note that the last value attribution to a variable is the one used as a result of the expression.
        - **Use JavaScript Editor**: Click to display the Edit Script dialog (see "Using scripts in the DataMapper" on page 533).
        - **Use selected text**: Inserts the text in the current data selection in the JavaScript **Expression**. If multiple lines or elements are selected, only the first one is used.
    - **Data Property**: The value of a data-level property set in the Preprocessor (see "Preprocessor step" on page 410).
    - **Record Property:** One of the local variables that you can create and that are reset for each document as opposed to data variables that are global because they are initialized only once at the beginning of each job.
    - **Automation Property**: The current value of a Document-level property set in the Preprocessor step (see "Preprocessor step" on page 410).
    - **Extractor Property**: The value of an internal extractor variable:
        - **Counter**: The value of the current counter iteration in a **Repeat** step.
        - **Vertical Position**: The current vertical position on the page, either in Measure (PDF) or Line (Text and CSV).
- **Operators**:
    - **is equal to**: The two specified values are identical for the condition to be **True**.
    - **contains**: The first specified value contains the second one for the condition to be **True**.

- **is less than**: The first specified value is smaller, numerically, than the second value for the condition to be **True**.
- **is greater than**: The first specified value is larger, numerically, than the second value for the condition to be **True**.
- **is empty**: The first specified value is empty. With this operator, there is no second value.
- **is true**: The first specified value is true. With this operator, there is no second value. Note that the condition evaluates the left operand following JavaScript boolean rules: if the value is 0, -0, null, false, NaN, undefined, or the empty string (""), it is evaluated to `false`. *All* other values, including any object, an empty array ([]), and the string "false", evaluate to `true`. (See Boolean in Mozilla's documentation.)

Invert condition: Inverts the result of the condition. For instance, **is empty** becomes **is not empty**.

Multiple Conditions step properties

The **Multiple Conditions** step contains a number of Case conditions (one to start with) and a Default, to be executed when none of the other cases apply. Cases are executed from left to right. For more information see "Steps" on page 409.

The properties described below become visible in the Step properties pane when the Multiple Conditions step is selected in the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Condition

Left operand, Right operand

The **Left and right operand** can be **Based on**:

- **Position**: The data in the specified position for the comparison.
    - **Left** (Txt and PDF only): The start position for the data selection. Note that conditions are done on the current line, either the current cursor position, or the current line in a **Repeat** step.
    - **Right** (Txt and PDF only): The end position for the data selection.
    - **Height** (Txt and PDF only): The height of the selection box.
    - **Column** (CSV and Database only): Drop-down listing all fields in the Data Sample, of which the value will be used.
    - **Top offset**: The vertical offset from the current pointer location in the Data Sample (**Viewer**).
    - **XPath** (XML only): The path to the XML field that is extracted.
    - **JsonPath** (JSON only): The path to the JSON element that is extracted. (See: "JsonPath" on page 395.)
    - **Use Selection**: Click to use the value of the current data selection for the extraction.
    - **Trim**: Select to trim empty characters at the beginning or the end of the field.
- **Value**: A specified static text value.
    - **Value**: The text value to use in the comparison.
    - **Use selected text**: Uses the text in the current data selection as the **Value**. If multiple lines or elements are selected, only the first one is used.
- **Field**: The contents of a specific field in the **Extracted Record**.
    - **Field**: The **Extracted Record** field to use in the comparison.
- **JavaScript**: The result of a JavaScript **Expression**.
    - **Expression**: The JavaScript line that is evaluated. Note that the last value attribution to a variable is the one used as a result of the expression. See also: "DataMapper Scripts API" on page 531.
    - **Use JavaScript Editor**: Click to display the Edit Script dialog (see "Using scripts in the DataMapper" on page 533).
    - **Use selected text**: Inserts the text in the current data selection in the JavaScript **Expression**. If multiple lines or elements are selected, only the first one is used.

- **Data Property**: The value of a data-level property set in the Preprocessor step (see "Steps" on page 409).
- **Record Property:** One of the local variables that you can create and that are reset for each document as opposed to data variables that are global because they are initialized only once at the beginning of each job.
- **Automation Property**: The current value of a Document-level property set in the Preprocessor step (see "Steps" on page 409).
- **Extractor Property**: The value of an internal extractor variable:
  - **Counter**: The value of the current counter iteration in a **Repeat** step.
  - **Vertical Position**: The current vertical position on the page, either in Measure (PDF) or Line (Text and CSV).

Condition

The **Condition** drop-down displays the cases in list form. Three buttons are available next to the list:

- **Add case**: Click to add a new case to the step. It will be placed next to any existing cases.
- **Remove case**: Delete the currently selected case.
- **Order Cases**: Under the **Name** column, select a case, then click one of the buttons on the right (**Delete**, **Move Up**, **Move Down**) to delete or change the order of the cases in the list.

Operators

Case conditions are made by comparison of the two operands, left and right, using a specific **Operator**.

- **is equal to**: The two specified value are identical for the condition to be **True**.
- **contains**: The first specified value contains the second one for the condition to be **True**.
- **is less than**: The first specified value is smaller, numerically, than the second value for the condition to be **True**.
- **is greater than**: The first specified value is larger, numerically, than the second value for the condition to be **True**.

- **is empty**: The first specified value is empty. With this operator, there is no second value.
- **Invert condition**: Inverts the result of the condition. For instance, **is empty** becomes **is not empty**.

Goto step properties

The **Goto** step moves the pointer within the source data to a position that is relative to the top of the record or to the current position. See also: "Steps" on page 409.

The properties of the Goto step described in this topic become visible in the Step properties pane when you select the Goto step on the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Goto Definition

With each type of source data, the movement of the cursor is defined in a specific way.

Text file

- **Target Type**: Defines the type of jump.
  - **Line**: Jumps a certain number of lines or to a specific line.
    - **From**: Defines where the jump begins:
      - **Current Position**: The **Goto** begins at the current cursor position.
      - **Top of record**: The **Goto** begins at line 1 of the source record.
    - **Move by**: Enter the number of lines or pages to jump.
  - **Page**: Jumps between pages or to a specific page.
    - **From**: Defines where the jump begins:
      - **Current Position**: The **Goto** begins at the current cursor position.

        - **Top of record**: The **Goto** begins at line 1 of the source record.
    - **Move by**: Enter the number of lines or pages to jump.
- **Next line with content**: Jumps to the next line that has contents, either anywhere on the line or in specific columns.
    - **Inspect entire page width**: When checked, the **Next line with content** and **Next occurrence of** options will look anywhere on the line. If unchecked, options appear below to specify in which area of each line the **Goto** step checks in:
        - **Left**: The starting column, inclusively.
        - **Right**: The end column, inclusively.
        - **Use selection**: Click while a selection is made in the Data Viewer to automatically set the left and right values to the left and right edges of the selection.
- **Next occurrence of**: Jumps to the next occurrence of specific text or a text pattern, either anywhere on the line or in specific columns.
    - **Inspect entire page width**: When checked, the **Next line with content** and **Next occurrence of** options will look anywhere on the line. If unchecked, options appear below to specify in which area of each line the **Goto** step checks in:
        - **Left**: The starting column, inclusively.
        - **Right**: The end column, inclusively.
        - **Use selection**: Click while a selection is made in the Data Viewer to automatically set the left and right values to the left and right edges of the selection.
    - **Expression**: Enter the text or **Regex** expression to look for on the page.
    - **Use selection**: Click while a selection is made in the Data Viewer to copy the contents of the first line of the selection into the **Expression** box.
    - **Use regular expression**: Check so that the **Expression** box is treated as a regular expression instead of static text. For example, using the expression (aaa|bbb|ccc) will move the data pointer to the next line in which either "aaa", "bbb" or "ccc" occurs.
      For more information on using **Regular Expressions (regex)**, see the Regular-Expressions.info Tutorial.

PDF file

- **Target Type**: Defines the type of jump .
  - **Physical distance**:
    - **From**: Defines where the jump begins:
      - **Current Position**: The **Goto** begins at the current cursor position.
      - **Top of record**: The **Goto** begins at line 1 of the source record.
    - **Move by**: Enter distance to jump.
  - **Page**: Jumps between pages or to a specific page.
    - **From**: Defines where the jump begins:
      - **Current Position**: The **Goto** begins at the current cursor position.
      - **Top of record**: The **Goto** begins at line 1 of the source record.
    - **Move by**: Enter the number pages to jump.
  - **Next line with content**: Jumps to the next line that has contents, either anywhere on the line or in specific columns.
    - **Inspect entire page width**: When checked, the **Next line with content** and **Next occurrence of** options will look anywhere on the line. If unchecked, options appear below to specify in which area of each line the **Goto**step checks in:
      - **Left**: The starting column, inclusively.
      - **Right**: The end column, inclusively.
      - **Use selection**: Click while a selection is made in the Data Viewer to automatically set the left and right values to the left and right edges of the selection.
  - **Next occurrence of**: Jumps to the next occurrence of specific text or a text pattern, either anywhere on the line or in specific columns.
    - **Inspect entire page width**: When checked, the **Next line with content** and **Next occurrence of** options will look anywhere on the line. If unchecked, options appear below to specify in which area of each line the **Goto** step checks in:
      - **Left**: The starting column, inclusively.
      - **Right**: The end column, inclusively.

- **Use selection**: Click while a selection is made in the Data Viewer to automatically set the left and right values to the left and right edges of the selection.

- **Expression**: Enter the text or **Regex** expression to look for on the page.
- **Use selection**: Click while a selection is made in the Data Viewer to copy the contents of the first line of the selection into the **Expression** box.
- **Use regular expression**: Check so that the **Expression** box is treated as a regular expression instead of static text. For more information on using **Regular Expressions (regex)**, see the Regular-Expressions informational Tutorial.

CSV file

- **From (CSV files)**: Defines where the jump begins:
  - **Current Position**: The **Goto** begins at the current cursor position.
    - **Move by**: Enter the number of lines or pages to jump.
  - **Top of record**: The **Goto** begins at line 1 of the source record.
    - **Move to**: Enter the number of lines or pages to jump.

XML file

- **Destination (XML files)**: Defines what type of jump to make:
  - **Sibling element**: Jumps the number of siblings (nodes at the same level) defined in the **Move by** option.
  - **Sibling element with same name**: Jumps the number of same name siblings (nodes at the same level of which the node is the same name) defined in the **Move by** option.
  - **Element, from top of record**: Jumps to the specified node. The XPATH in the **Absolute XPATH** option starts from the root node defined by **/**.
  - **Element from current position**: Jumps to a position relative to the current position of the cursor. The XPATH in the **Relative XPATH** option defines where to go, **../** goes up a level, **./** refers to the current level.
  - **Level Up/Down**: Jumps up or down one node level (up to the parent, down to a child). The number of levels to change is defined in the **Move by** option. Use a

negative value to jump to a parent element, a positive value to go to a child element.

JSON file

- **Destination**: Defines what type of jump to make:
  - **Sibling element**: Jumps the number of siblings (elements at the same level) defined in the **Move by** option. Use a negative value to jump to the previous sibling, or a positive value to go to the next sibling. If there are not enough siblings to make the requested move, the cursor will not move.
  - **Element, from top of record**: Jumps to the specified element. The JsonPath in the **Absolute JsonPath** option starts from the root defined by **$**.
  - **Element from current position**: Jumps to a child element starting from the current position of the cursor. The JsonPath in the **Relative JsonPath** option defines where to go. Going up to a parent element is not possible with a relative JsonPath.If a JsonPath returns a collection of elements instead of a single element, then the GoTo step moves to the position of the first element in the collection. Other elements in the collection are ignored.
  - **Level Up/Down**: Jumps up or down a number of element levels (up to a parent, down to a child). The number of levels to change is defined in the **Move by** option. Use a negative value to jump to a parent element, or a positive value to go to a child element.

Postprocessor step properties

The Postprocessor step does not run for every Source Record in the Data Sample. It runs once, at the end of the Steps, after all records have been processed. For more information see "Postprocessor step" on page 418.

The properties described below become visible in the Step properties pane when the Postprocessor step is selected in the Steps pane.

Description

This subsection is collapsed by default in the interface, to give more screen space to other important parts.

Name: The name of the step. This name will be displayed on top of the step's icon in the **Steps** pane.

Comments: The text entered here will be displayed in the tooltip that appears when hovering over the step in the **Steps** pane.

Postprocessor

The Postprocessor subsection defines what postprocessors run on the Data Sample at the end of the data mapping workflow. Each Postprocessor runs in turn, using the result of the previous one as an input.

- **Name**: The name to identify the Postprocessor.
- **Type**: The type of Postprocessor. Currently there is a single type available.
  - **JavaScript**: Runs a JavaScript Expression to modify the Data Sample. See "DataMapper Scripts API" on page 531.
    - **Use JavaScript Editor**: Click to display the Edit Script dialog (see "Using scripts in the DataMapper" on page 533).
- **Add Postprocessor**: Click to add a new Postprocessor. Its settings can be modified once it is added.
- **Remove Postprocessor**: Click to remove the currently selected Postprocessor.
- **Move Up**: Click to move the Postprocessor up one position.
- **Move Down**: Click to move the Postprocessor down one position.
- **Apply**: Run the Postprocessor.

  Note: In the DataMapper and Designer, only one data record is active at any given time. Therefore, the changes made by the post-processes are only visible on the current data record (i.e. the one currently displayed). If you change data records, you must press the **Apply** button on the Postprocessor step once more. In order to test a case with multiple records you must either use the data mapping configuration in a print job through the Designer or in the Execute Data Mapping task in Workflow so that all records will be processed by the Postprocessor step.

  Note: The **Apply** button has an effect at design time only. It is ignored at runtime. When a data mapping configuration is executed on the server, all postprocessors are executed in original order.

Postprocessor definition

JavaScript

- **Expression**: The JavaScript expression that will run on the Data Sample. See "DataMapper Scripts API" on page 531.
- **Use JavaScript Editor**: Click to display the Script Editor dialog.
- **Use selected text**: Uses the text in the current data selection as the Value. If multiple lines or elements are selected, only the first one is used.

Toolbar

In the DataMapper module, the following buttons are available in the top toolbar.

File manipulation

- **New**: Displays the **New** wizard where a new data mapping configuration or a new template can be created.
- **Open**: Displays the **Open** dialog to open an existing data mapping configuration.
- **Save**: Saves the current data mapping configuration. If the configuration has never been saved, the **Save As...** dialog is displayed.

Step manipulation

Note: All steps except the Action step require an active data selection in the Data Viewer (see "Selecting data" on page 394 and "The Data Viewer" on page 458).

- **Add Extract Step**: Adds an Extract Step with one or more extract fields. If more than one line or field is selected in the Data Viewer, each line or field will have an extract field.
- **Add Goto Step**: Adds a Goto step that moves the selection pointer to the beginning of the data selection. For instance if an XML node is selected, the pointer moves to where that node is located.
- **Add Condition Step**: Adds a condition based on the current data selection. The "True" branch gets run when the text is found on the page. Other conditions are available in the step properties once it has been added.

- **Add Repeat Step**: Adds a loop that is based on the current data selection, and depending on the type of data. XML data will loop on the currently selected node, CSV loops for all rows in the record. In Text and PDF data, if the data selection is on the same line as the cursor position, the loop will be for each line until the end of the record. If the data selection is on a lower line, the loop will be for each line until the text in the data selection is found at the specified position on the line (e.g. until "TOTAL" is found).
- **Add Extract Field**: Adds the data selection to the selected **Extract** step, if an extract step is currently selected. If multiple lines, nodes or fields are selected, multiple extract fields are added simultaneously.
- **Add Multiple Conditions**: Adds a condition that splits into multiple case conditions.
- **Add Action Step**: Adds a step to create a custom JavaScript snippet. See "DataMapper Scripts API" on page 531 for more details.
- **Cut Step**: Removes the currently selected step and places it in the clipboard. If the step is a Repeat or a Condition, all steps under it are also placed in the clipboard. If there is already a step in the clipboard, it will be overwritten.
- **Copy Step**: Places a copy of the currently selected step in the clipboard. The same details as the Cut step applies.
- **Paste Step**: Takes the step or steps in the clipboard and places them after the currently selected step.
- **Delete Step**: Deletes the currently selected step. If the step is a Repeat or Condition, all steps under it are also deleted.
- **Ignore Step/Reactivate Step**: Click to set the step to be ignored (aka disabled) or to reactivate it. Disabled steps do not run when in DataMapper and do not execute when the data mapping configuration is executed in Workflow. However, they can still be modified normally.

Miscellaneous

- **Validate All Records**: Runs the process on all records and verifies that no errors are present in any of the records. Errors are displayed in the "Messages pane" on page 464.
- **Add Data Sample**: Displays a dialog to open a new Data Source to add it as a Data Sample in the data mapping configuration. Data Samples are visible in the "Settings

pane" on page 468.

- **Send to Workflow**: Opens the "Send to Workflow dialog" on page 1206 to send files to a local Workflow software installation.

Welcome Screen

The **Welcome Screen** appears when first starting OL Connect. It offers useful shortcuts to resources and to recent templates, data mapping configurations, and print presets.

If you are new to OL Connect and you don't know where to start, see "Welcome to OL Connect 2025.1" on page 15.

The Welcome Screen can be reopened in two ways:

- The **Home** button at the top right in the "Toolbars" on page 1264.
- From the Menus in **Help**, **Welcome Screen**.

To go back from the Welcome Screen to the template or data mapping configuration that you were working on:

- Close the Welcome Screen by clicking the cross next to the text 'Welcome' at the top.

Contents

Top bar:

- **OL Connect**: Opens the OL Connect website.
- **News** (bell icon): Displays a list of what's new in OL Connect.
- **Support**: Brings you to the customer portal login page.
- **Software activation**: Opens the Objectif Lune Web Activation Manager.

Menu on the left:

- **Home**: Resets the Welcome screen to its start page. The start page has:
  - One link to a blog post or tutorial in the Upland Objectif Lune resource center. To see more, click **Learn**.
  - A list of recently opened files.
- **New**: Lets you create a new OL Connect file or sample project.

- **Print**: Start designing a template for print output. See "Creating a Print template with a Wizard" on page 618.
- **Email**: Start designing a template for email output. See "Creating an Email template with a Wizard" on page 664.
- **Web**: Start designing a template for a web page. See "Creating a Web template with a Wizard" on page 683.
- **Data**: Start to build a data mapping configuration with a wizard; see "Data mapping configurations" on page 337.
- **Project**: Displays a list of available Sample Projects, producing a complete Connect project; see "Sample Projects" on page 1185.
- **Presets**: Start creating "Job Presets" on page 1646 and "Output Presets" on page 1648 to configure your print jobs.

Tip: Scroll down to **Online resources** to download a ready-made file with existing demo content as a starter for your file.

- **Open**: Lets you open an existing template, data mapping configuration, job preset, output preset, or package file.

Online resources:

- **Learn**: Displays a list of the most recent blog posts and tutorials in the Upland Objectif Lune resource center. Click a link to open the Objectif Lune resource center and read the article online.
- **Forum**: Ask your questions and share tips and tricks in the user forum (https://olforum.uplandsoftware.com/).
- **Online Help**: Opens this documentation.

Files:

- **Printer configurations**: Lists printer definition files (PDC) available for download. The selected files will be downloaded to the current user's Connect\workspace\configurations\ folder and can be imported via the Print Wizard; see "Adding print output Models to the Print Wizard" on page 1652.
- **Finishing configurations**: Lists HCF files available for download. High Capacity Feeders (HCF) are also commonly referred to as Inserters or Folder-Inserters. The Inserter options in the print settings are dependent upon the HCF model selected; see

"Inserter options" on page 1464.

- **My license**: Shows your current license's details.

DataMapper Scripts API

This page describes the different features available in scripts created inside DataMapper. See "Using scripts in the DataMapper" on page 533.

Objects

Name	Description	Available in scripts of type
"Objects" on page 540	A ScriptableAutomation object encapsulating the properties of the OL Connect Workflow process that triggered the current operation.	Boundaries, all steps except Goto
"boundaries" on page 541	An object encapsulating properties and methods allowing to define the boundaries of each document in the job.	Boundaries
"data" on page 546	A data object encapsulating properties and methods pertaining to the original data stream.	Boundaries, all steps except Goto
"db" on page 559	An object that allows to connect to a database.	Boundaries, all steps except Goto
"logger" on page 561	An object that allows to log error, warning or informational messages.	Boundaries, all steps except Goto
"record" on page 565	The current record in the main data set.	Extract, Condition, Repeat and Multiple Conditions steps
"region" on page 565	An object that defines a subsection of the input data.	Boundaries
"sourceRecord" on page 568	An object containing properties specific to the current source record being processed.	Boundaries, all steps except Goto and Post-processor
"steps" on page 568	An object encapsulating properties and methods pertaining to the current data mapping configuration.	Extract, Condition, Repeat and Multiple Conditions steps

Functions

These functions are available in Boundaries and Steps scripts.

Name	Description
"Functions" on page 576	Copies a file to the target file path, replacing it if it already exists.
"createGUID()" on page 577	Returns a unique 36-character string consisting of 32 alphanumeric, lower case characters and four hyphens (format: 8-4-4-4-12). Example: 123e4567-e89b-12d3-a456-426655440000.
"createHTTPRequest()" on page 577	Creates a new HTTP Request Object.
"createTmpFile()" on page 578	Creates a file with a unique name in the temporary work folder and returns a **file** object.
"deleteFile()" on page 579	Deletes a file.
"execute()" on page 580	Calls an external program and waits for it to end.
isRuntime()	Returns true if the data mapping process is currently running in runtime mode, or false if the configuration is running in debug mode (i.e. in the DataMapper).
"newByteArray()" on page 580	Returns a new byte array.
"newCharArray()" on page 580	Returns a character array.
"newDoubleArray()" on page 581	Returns a double array.
"newFloatArray()" on page 581	Returns a float array.
"newIntArray()" on page 581	Returns an integer array.
"newLongArray()" on page 581	Returns a long array.
"newStringArray()" on page 581	Returns a string array.
"nextRecord()" on page 582	A Step function that interrupts processing of the current record and instructs the DataMapper to move to the next record.
"openBinaryReader()" on page 582	Opens a file as a binary file for reading purposes.

Name	Description
"openBinaryWriter()" on page 583	Opens a file as a binary file for writing purposes.
"openTextReader()" on page 583	Opens a file as a text file for reading purposes.
"openTextWriter()" on page 584	Opens a file as a text file for writing purposes.
"stopProcessing()" on page 586	A Step function that interrupts the DataMapper to stop processing of the entire data stream.

Using scripts in the DataMapper

In the DataMapper every part of the extraction process can be customized using scripts.

A script can be used to set **boundaries** for a data source (see "Setting boundaries using JavaScript" on page 535). The script determines where a new record starts.

Scripts can also be used in different steps in the extraction workflow. You can:

- Modify the incoming data prior to executing the rest of the extraction workflow, via a **Preprocessor action** (see "Preprocessor step" on page 410).
- Edit extracted data in a field of the Data Model using a **Post function** script (entered on the Step properties pane, under Field Definition; see "Modifying extracted data" on page 430 and "Extract step properties" on page 492).

  Tip: Read more about the Post function in the following how-to: DataMapper tidbits.

- Enter a script in a **JavaScript-based field** (see "Expression-based field" on page 425). Note that the last value attribution to a variable is the one used as the result of the expression. It is possible to refer to previously extracted fields if they are extracted higher in this list or in previous **Extract** steps in the extraction workflow.
- Let an **Action** step run a JavaScript, for example to:
  - Extract data to fields at the root level or in a detail table. See Extracting data with an Action step script.

- Add a value to a custom property defined in the Preprocessor step. (See "Properties and runtime parameters" on page 387.) Note that only the value of properties of which the scope is set to "Each record" can be changed in a script.

- Change the left and right operands in a **Condition** step to a JavaScript expression. (On the Step properties pane, under Condition, set Based on to JavaScript; see "Condition step properties" on page 515 and "Multiple Conditions step properties" on page 518.)
- Further process the resulting record set after the entire extraction workflow has been executed, via a **Postprocessor** (see "Postprocessor step" on page 418).

The script can always be written directly in a small script area or in the **Edit script** dialog.

To invoke this dialog click the **Use JavaScript Editor** button .

> **Tip:** In the **Edit script** dialog, press **Ctrl** + **Space** to bring up the list of available JavaScript objects and functions (see "DataMapper Scripts API" on page 531). Use the arrow keys to select a function or object and press enter to insert it. Type a dot after the name of the function or object to see which features are subsequently available.

Keyboard shortcuts for the script editor are listed in the following topic: "Keyboard shortcuts" on page 1223.

Syntax rules

In the DataMapper, all scripts must be written in **JavaScript**, following JavaScript syntax rules. For example, each statement should end with **;** and the keywords that can be used, such as **var** to declare a variable, are JavaScript keywords. There are countless tutorials available on the Internet to familiarize yourself with the JavaScript syntax. For a simple script all that you need to know can be found on the following web pages: W3Schools website - JavaScript Syntax and https://www.w3schools.com/js/js_if_else.asp. A complete JavaScript guide for beginners can be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

DataMapper API

Certain features that can be used in a DataMapper script do not exist in the native JavaScript library. These are additional JavaScript features, designed for use in Connect only. All features designed for use in the DataMapper are listed in the DataMapper API (see "DataMapper Scripts API" on page 531).

External JavaScript libraries

The External JS Libraries box on the Settings pane lets you add JavaScript libraries to your configuration and displays all the libraries that have been imported (see "Settings pane" on page 468).
You can use JavaScript libraries to add more JavaScript functionality to your data mapping configuration. Any functions included in a JavaScript library that is imported in a data mapping configuration will be available in Preprocessor scripts as well as Action tasks, Post functions and JavaScript-based extraction steps.

Take the following JavaScript function, for example:

```
function myAddFunction(p1, p2) {
    return p1 + p2;
};
```

If this is saved as myFunction.js and imported, then the following would work anywhere in the configuration:

```
var result = myAddFunction(25, 12); // returns 37!
```

Setting boundaries using JavaScript

As soon as you select the **On Script** option as the trigger for establishing record boundaries (see "Record boundaries" on page 373), you are instructing the DataMapper to read the source file sequentially and to trigger an event each and every time it hits a delimiter. (What a delimiter is, depends on the source data and the settings for that data; see "Input data settings (Delimiters)" on page 371).
In other words, the script will be executed - by default - as many times as there are delimiters in the input data.

If you know, for instance, that a PDF file only contains documents that are 3 pages long, your script could keep count of the number of times it's been called since the last boundary was set (that is, the count of delimiters that have been encountered). Each time the count is a multiple of 3, it could set a new record boundary. This is basically what happens when setting the trigger to **On Page** and specifying 3 as the Number of Pages.

> **Note:** Remember that a boundary script is being called on each new delimiter encountered by the DataMapper parsing algorithm. If for instance a database query returns a million records, the script will be executing a million times! Craft your script

in such a way that it doesn't waste time examining all possible conditions. Instead, it should terminate as soon as any condition it is evaluating is false.

Accessing data

Data available inside each event

Every time a delimiter is encountered, an event is triggered and the script is executed. The event gives the script access to the data between the current location - the start of a row, line or page - and the next delimiter. So at the beginning of the process for a PDF or text file, you have access to the first page only, and for a CSV or for tabular data, that would be the first row or record.

This means that you can:

- Examine the data found in between delimiters for specific conditions.
- Examine specific regions of that data, or the available data as a whole.
- Compare the contents of one region with another.
- Etc.

To access this data in the script, use the **get()** function of the **boundaries** object. This function expects different parameters depending on the type of source file; see "get()" on page 543.

Getting access to other data

Data that is not passed with the event, but that is necessary to define the record boundaries, can be stored in the **boundaries** object using the **setVariable** function (see "boundaries" on page 541 and "setVariable()" on page 545). The data can be retrieved using the boundaries' **getVariable** function (see "getVariable()" on page 543).
This way the script can access values that were evaluated in previous pages or rows, across delimiters, so you can easily set record boundaries that span over multiple delimiters.

For more information on the syntax, please refer to "DataMapper Scripts API" on page 531.

Examples

Basic example using a CSV file

Imagine you are a classic rock fan and you want to extract the data from a CSV listing of all the albums in your collection. Your goal is to extract records that change whenever the artist OR the release year changes.

Here's what the CSV looks like:

"Artist","Album","Released"
"Beatles","Abbey Road",1969
"Beatles","Yellow Submarine",1969
"Led Zeppelin","Led Zeppelin 1",1969
"Led Zeppelin","Led Zeppelin 2",1969
"Beatles","Let it be",1969
"Rolling Stones","Let it bleed",1969
"Led Zeppelin","Led Zeppelin 3",1970
"Led Zeppelin","Led Zeppelin 4",1971
"Rolling Stones","Sticky Fingers",1971

Note: The first line is just the header with the names of the CSV columns. The data is already sorted per year, per artist, and per album.

Your goal is to examine two values in each CSV record and to act when either changes. The DataMapper GUI allows you to specify a **On Change** trigger, but you can only specify a single field. So for instance, if you were to set the record boundary when the "Released" field changes, you'd get the first four lines together inside a single record. That's not what you want since that would include albums from several different artists. And if you were to set it when the "Artist" field changes, the first few records would be OK but near the end, you'd get both the Led Zeppelin 3 and Led Zeppelin 4 albums inside the same record, even though they were released in different years.

Essentially, we need to combine both these conditions and set the record boundary when EITHER the year OR the artist changes.

Here's what the script would look like:

```
/* Read the values of both columns we want to check */
var zeBand = boundaries.get(region.createRegion("Artist"));
var zeYear = boundaries.get(region.createRegion("Released"));

/* Check that at least one of our variables holding previous values has been
 initialized already, before attempting to compare the values */

if (boundaries.getVariable("lastBand")!=null) {
        if (zeBand[0] != boundaries.getVariable("lastBand") || zeYear[0] !=
boundaries.getVariable("lastYear") )
        {
        boundaries.set();
        }
}
```

```
boundaries.setVariable("lastBand",zeBand[0]);
boundaries.setVariable("lastYear",zeYear[0]);
```

- The script first reads the two values from the input data, using the createRegion() method (see: "createRegion()" on page 566). For a CSV/database data type, the parameter it expects is simply the column name. The region is passed as a parameter to the get() method, which reads its contents and converts it into an array of strings (because any region, even a CSV field, may contain several lines).
- To "remember" the values that were processed the last time the event was triggered, we use variables that remain available in between events. Note that these variables are specific to the Boundary context and not available in any other scripting context in the DataMapper.
- The script first checks if those values were initialized. If they weren't, it means this is the first iteration so there's no need to compare the current values with previous values since there have been none yet. But if they have already been initialized, then a condition checks if either field has changed since last time. If that's the case, then a boundary is created through the set() method.
- Finally, the script stores the values it just read in the variables using the setVariables() method. They will therefore become the "last values encountered" until the next event gets fired. When called, setVariables() creates the specified variable if it doesn't already exist and then sets the value to the second parameter passed to the function.

You can try it yourself. Paste the data into the text editor of your choice and save the file to Albums.csv. Then create a new DataMapper configuration and load this CSV as your data file. In the Data Input Settings, make sure you specify the first row contains field names and set the **Trigger** to **On script**. Then paste the above JavaScript code in the **Expression** field and click the **Apply** button to see the result.

Basic example using a text file

This example is similar to the previous example, but now the data source is a plain text file that looks like this:

Beatles	Abbey Road	1969
Beatles	Yellow Submarine	1968
Led Zeppelin	Led Zeppelin 1	1969
Led Zeppelin	Led Zeppelin 2	1969
Beatles	Let it be	1970
Rolling Stones	Let it bleed	1969
Led Zeppelin	Led Zeppelin 3	1970
Led Zeppelin	Led Zeppelin 4	1971
Rolling Stones	Sticky Fingers	1971

The purpose of the script, again, is to set the record boundary when EITHER the year OR the artist changes.

The script would look like this:

```
/* Read the values of both columns we want to check */
var zeBand = boundaries.get(region.createRegion(1,1,30,1));
var zeYear = boundaries.get(region.createRegion(61,1,65,1));

/* Check that at least one of our variables holding previous values have been
 initialized already, before attempting to compare the values */

if (boundaries.getVariable("lastBand") != null) {
        (zeBand[0]!=boundaries.getVariable("lastBand") || zeYear[0] !=
boundaries.getVariable("lastYear"))
        {
        boundaries.set();
        }
}
boundaries.setVariable("lastBand",zeBand[0]);
boundaries.setVariable("lastYear",zeYear[0]);
```

This script uses the exact same code as used for CSV files, with the exception of parameters expected by the createRegion() method. The get method adapts to the context (the data source file) and therefore expects different parameters to be passed in order to achieve the same thing. Since a text file does not contain column names as a CSV does, the API expects the text regions to be defined using physical coordinates. In this instance: Left, Top, Right, Bottom.

To try this code, paste the data into a text editor and save the file to Albums.txt. Then create a new DataMapper configuration and load this Text file as your data file. In the Data Input Settings, specify **On lines** as the **Page delimiter** type with the number of lines set to 1. When you now set the boundary **Trigger** to **On script**, the file will be processed line per line (triggering the event on each line). Paste the above code in the JavaScript expression field and click the **Apply** button to see the result.

Note: The PDF context also expects physical coordinates, just like the Text context does, but since PDF pages do not have a grid concept of lines and columns, the above parameters would instead be specified in millimeters relative to the upper left corner of each page. So for instance, to create a region for the Year, the code might look like this:

```
region.createRegion(190,20,210,25)
```

which would create a region located near the upper right corner of the page.

That's the only similarity, though, since the script for a PDF would have to look through the entire page and probably make multiple extractions on each one since it isn't dealing with single lines like the TXT example given here.

For more information on the API syntax, please refer to "DataMapper Scripts API" on page 531.

Objects

automation

Returns a **ScriptableAutomation** object encapsulating the properties of the OL Connect Workflow process that triggered the current operation.

This `automation` object is available in all script types and with all file types.

Note: The `automation` object available in Designer scripts is not of the same type. It has different properties.

Properties

The following table lists the properties of the `automation` object. All properties are read-only.

Property	Description
jobInfo	Returns a ScriptableAutomation object containing Job Info 1 to 9 values from OL Connect Workflow (see "Fixed automation properties" on page 489, and Job Info variables in the Workflow Online Help).
properties	Returns a ScriptableAutomation object containing additional information (`ProcessName`, `OriginalFilename` and `TaskIndex`) from OL Connect Workflow (see "Fixed automation properties" on page 489).
parameters	Returns a ScriptableAutomation object containing the custom runtime parameters defined in the data mapping configuration (see "Properties and runtime parameters" on page 387). At runtime this object also contains any local and global variables defined by the user in OL Connect Workflow, to ensure compatibility with previous versions.
variables	**Deprecated**; use `automation.parameters` instead. Both properties contain the same list of elements, but the `variables` property is kept for backward compatibility and may eventually be removed.

Examples

To access JobInfo 1 to 9 defined in Workflow (see Job Info variables):

```
automation.jobInfo.JobInfo1;
```

To access ProcessName, OriginalFilename or TaskIndex from Workflow:

```
automation.properties.OriginalFilename;
```

To access Workflow variables (see "Properties and runtime parameters" on page 387):

```
automation.parameters.runtimeparametername;
```

boundaries

Returns a `boundaries` object encapsulating properties and methods allowing to define the boundaries of each document in the job.

This object is available when triggering document boundaries **On script**.

Properties

The following table lists the properties of the `boundaries` object.

Property	Description
currentDelim	A read-only 1-based index (number) of the current delimiter in the file. In other words, the Beginning Of File (BOF) delimiter equals 1. It indicates the position of the current delimiter relative to the last document boundary.

Methods

The following table describes the functions of the `boundaries` object. They are available with all file types.

Method	Description	Script type
"find()" on the facing page	Finds the first occurrence of a string starting from the current position.	Boundaries Preprocessor, Extract, Condition, Repeat, Action, and Postprocessor steps
"get()" on page 543	Retrieves an array of strings.	Boundaries

Method	Description	Script type
"getVariable()" on the next page	Retrieves a value of a variable stored in the `boundaries` object.	Boundaries
"set()" on page 544	Sets a new record boundary (see: "Record boundaries" on page 373).	Boundaries
"setVariable()" on page 545	Sets a `boundaries` variable to the specified value, automatically creating the variable if it doesn't exist yet.	Boundaries

find()

Method of the `boundaries` object that finds a string in a region of the data source file. The method returns the region in which the string was searched (PDF file) or the exact region in which the string was encountered (Text file).
To check if the call to `boundaries.find()` was successful, you can read the region object's property `found` (see "region" on page 565).

> **Note:** In OL Connect 1.8 and previous versions, the DataMapper's `boundaries.find()` function returned the region searched within PDF files, whereas for text files it returned the exact region where the text was found. In 2018.1 this was changed so that `boundaries.find()` on PDFs would return the exact region where the text was found, the same as for text files. However, it was subsequently found that this could cause issues with previously created templates using the function on PDF files. Consequently, this change was reverted in 2018.1.1.

find(stringToFind, in_Region)

Finds the string **stringToFind** in a rectangular region defined by **in_Region**.

stringToFind

String to find.

in_Region

The in_Region region can be created prior to the call to find() with the region.createRegion() method. It depends on the type of data source how a region is defined; see "createRegion()" on page 566.

When used to search through a Text file, the `find()` method returns a different `region` object (see "region" on page 565) whose `range` property is adjusted to point to the exact physical location where the match was found. This will always be a subset of the `in_`

`Region.range` property. It can be used to determine the exact location where the match occurred.

Use `boundaries.get()` to retrieve the actual text from the resulting region; see "get()" below.

Example

This script sets a boundary when the text TOTAL is found on the current page in a PDF file. The number of delimiters is set to 1, so the boundary is set on the next delimiter, which is the start of the next page.

```
if (boundaries.find("TOTAL", region.createRegion(10,10,215,279)).found) {
        boundaries.set(1);
}
```

get()

The `get()` method reads the contents of a `region` object and converts it into an array of strings (because any region may contain several lines).
How the `region` is defined, depends on the type of source data; see "region" on page 565 and "createRegion()" on page 566.

get(in_Region)

in_Region

A region object. What type of object this is depends on the type of source data, however in any case the `region` object can be created with a call to `region.createRegion()`; see "createRegion()" on page 566.

Example

This script retrieves all text from the Email_Address field in a CSV or database file.

```
boundaries.get(region.createRegion("Email_Address"));
```

getVariable()

Method that retrieves the value currently stored in a variable.

> Boundary variables are carried over from one iteration of the Boundaries script to the next, while native JavaScript variables are not.

getVariable(varName)

varName

String name of the variable from which the value is to be retrieved. If the variable does not exist, the value `null` is returned. It is considered good practice (almost mandatory, even) to always check whether a variable is defined before attempting to access its value.

set()

Sets a new DataMapper record boundary.

set(delimiters)

delimiters

Sets a new record boundary. The **delimiters** parameter is an offset from the current delimiter, expressed in an integer that represents a number of delimiters.
If this parameter is not specified, then a value of 0 is assumed. A value of 0 indicates the record boundary occurs on the current delimiter.
A negative value of -n indicates that the record boundary occurred -n delimiters before the current delimiter.
A positive value of n indicates that the record boundary occurs +n delimiters after the current delimiter.

> **Note:** Specifying a positive value not only sets the DataMapper record boundary but it also advances the current delimiter to the specified delimiter. That's where the processing resumes. This allows you to skip some pages/records when you know they do not need to be examined. Negative (or 0) values simply set the boundary without changing the current location.

Example

This script sets a boundary when the text TOTAL is found on the current page in a PDF file. The number of delimiters is set to 1, so the boundary is set on the next delimiter, which is the start of the next page.

```
if (boundaries.find("TOTAL", region.createRegion(10,10,215,279)).found) {
    boundaries.set(1);
}
```

Assume you want to set record boundaries whenever the text "TOTAL" appears in a specific region of the page of a PDF file, but the PDF file has already been padded with blank pages

for duplexing purposes. The boundary should therefore be placed at the end of the page where the match is found if that match occurs on an even page, or at the end of the next blank page, if the match occurs on an odd page. Recall that for PDF files, the natural delimiter is a PDF page. The JavaScript code would look something like the following:

```
var myRegion = region.createRegion(150,220,200,240);
if(boundaries.find("TOTAL", myRegion).found) {
    /* a match was found. Check if we are on an odd or even page and set the Boundary accordingly */
    if((boundaries.currentDelim % 2) !=0 ) {
        /* Total is on odd page, let's set the document Boundary one delimiter further, thereby skipping the next blank page */
        boundaries.set(1);
        } else {
        /* Total is on an even page, set the document Boundary to the current delimiter */
        boundaries.set();
        }
    }
}
```

setVariable()

This method sets a variable in the `boundaries` to the specified value, automatically creating the variable if it doesn't exist yet.

> **Note:** Boundary variables are carried over from one iteration of the Boundaries script to the next, while native JavaScript variables are not.

setVariable(varName, varValue)

Sets variable **varName** to value **varValue**.

varName

String name of the variable of which the value is to be set.

varValue

Object; value to which the variable has to be set.

Example

This script examines a specific region and stores its contents in a variable in the boundaries.

```
var addressRegion = region.createRegion(10, 30, 100, 50);
var addressLines = boundaries.get(addressRegion);
boundaries.setVariable("previousLines",addressLines);
```

data

Returns a `data` object encapsulating properties and methods pertaining to the original data stream.

Properties

The following table lists the properties of the data object.

Property	Description
filename	Returns the fully qualified file name of the data file, i.e. the temporary work file being processed.
properties	Returns an array of the custom properties defined in the **Preprocessor** step that have their Scope set to "Entire data". These properties are statically set at the start of the job. (See "Properties and runtime parameters" on page 387 for details)

Methods

The following table lists the methods of the data object.

Method	Description	Script type	File type
"extract()" on the next page	Extracts the text value from a rectangular region.	Extract, Condition, Repeat, and Action steps	All
"extractByIndex(index, rowOffset)" on page 553	Extracts the value from the specified column and row.	Extract, Condition, Repeat, and Action steps	CSV/ XLSX/ XLS
"extractMeta()" on page 554	Extracts the value of a metadata field.	Extract, Condition, Repeat, and Action steps	All
"fieldExists()" on page 554	Method that returns **true** if the specified metadata field, column or node exists.	Boundaries Preprocessor, Extract, Condition, Repeat, Action, and Postprocessor steps	All
"fieldExistsByIndex (index)" on page 555	Method that returns **true** if the column, specified by index, exists.	Boundaries Preprocessor, Extract, Condition, Repeat, Action, and Postprocessor steps	CSV/ XLSX/ XLS

Method	Description	Script type	File type
"find()" on page 556	Finds the first occurrence of a string starting from the current position.	Boundaries Preprocessor, Extract, Condition, Repeat, Action, and Postprocessor steps	All
"findRegExp()" on page 557	Finds the first match for a regular expression pattern starting from the current position.	Extract, Condition, Repeat, Multiple Conditions and Action steps	Text, PDF

extract()

Extracts the text value from selected data: a node path, column, or rectangular region, depending on the type of data source.

This method always returns a String.

extract(left, right, verticalOffset, regionHeight, separator)

Extracts a value from a position in a **text** file. Coordinates are expressed as characters (horizontally) or lines (vertically).

left

Number that represents the distance, measured in characters, from the left edge of the page to the left edge of the rectangular region. The leftmost character is character 1.

right

Number that represents the distance, measured in characters, from the left edge of the page to the right edge of the rectangular region.

verticalOffset

Number that represents the current vertical position, measured in lines.

regionHeight

Number that represents the total height of the region, measured in lines.
Setting the regionHeight to 0 instructs the DataMapper to extract all lines starting from the given position until the end of the record.
Specifying an extraction height that is longer than the number of remaining lines results in a "step out of bound" error message.

separator

String inserted between all lines returned from the region. If you don't want anything to be inserted between the lines, specify an empty string ("").

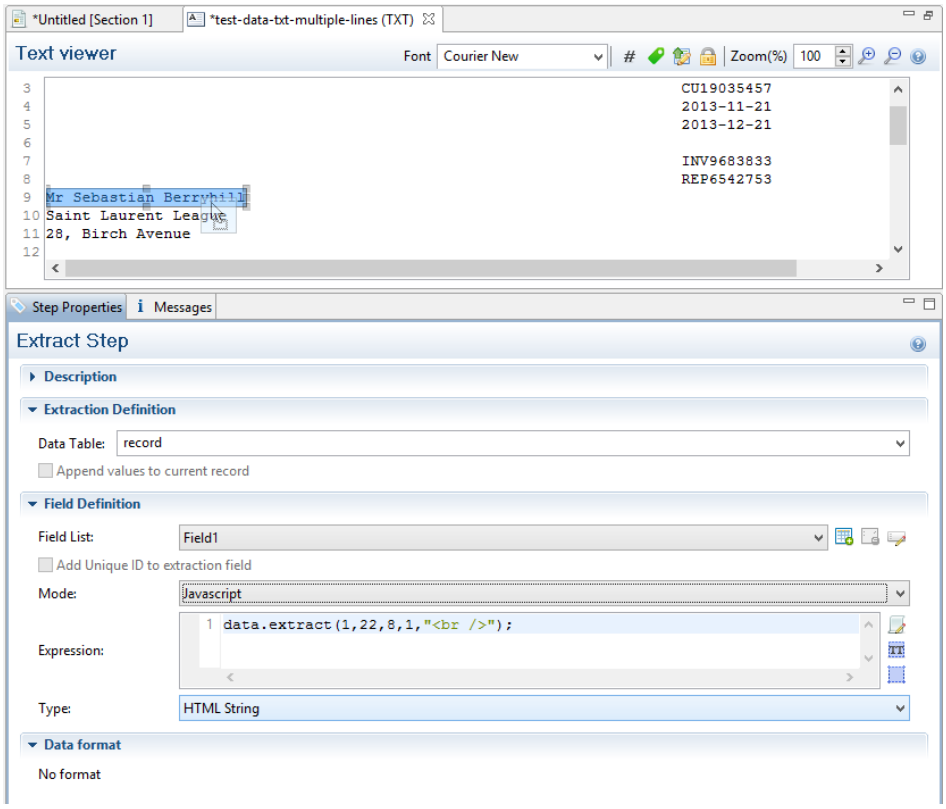
Tip:

- "<br/>" is a very handy string to use as a separator. When the extracted data is inserted in a Designer template, "<br/>" will be interpreted as a line break, because <br/> is a line break in HTML and Designer templates are actually HTML files.
- Setting the regionHeight to 0 makes it possible to extract a variable number of lines at the end of a record.

Examples

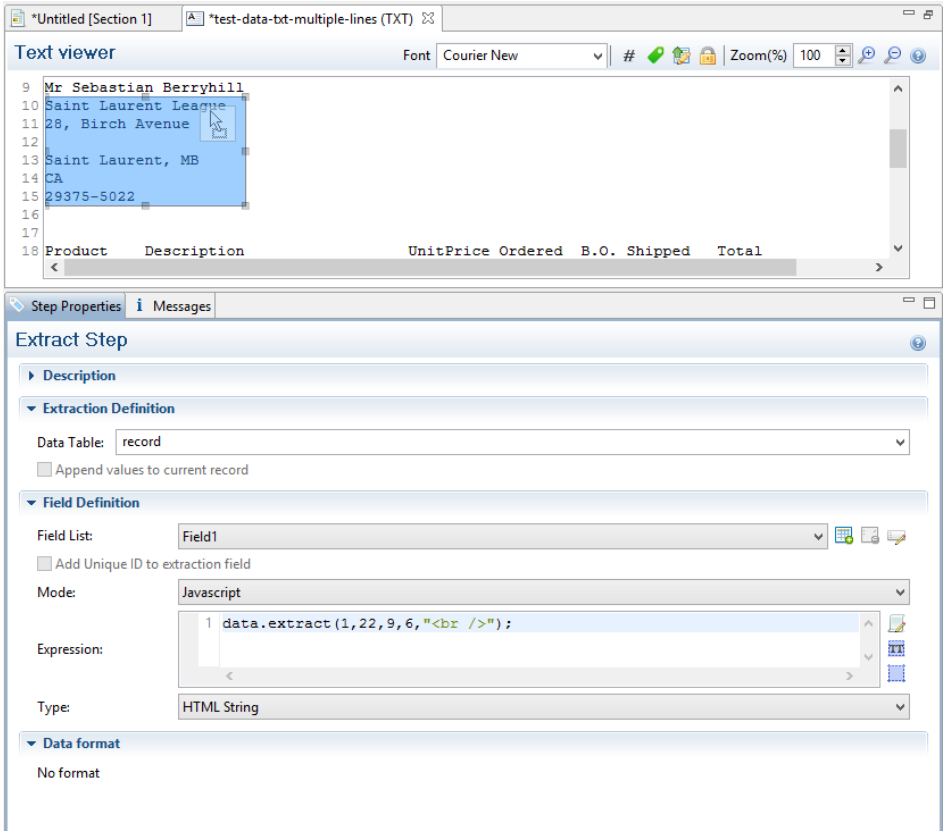
Example 1:

The script command data.extract(1,22,8,1,"<br>"); means that the left position of the extracted information is located at character 1, the right position at character 22, the offset position is 8 (since the line number is 9) and the regionHeight is 1 (to select only 1 line). Finally, the "<br>" string is used for concatenation.



Example 2:

The script command **data.extract(1,22,9,6,"<br>");** means that the left position of the extracted information is located at 1, the right position at 22, the offset position is 9 (since the first line number is 10) and the regionHeight is 6 (6 lines are selected). Finally, the "<br>" string is used for concatenation.



extract(xPath)

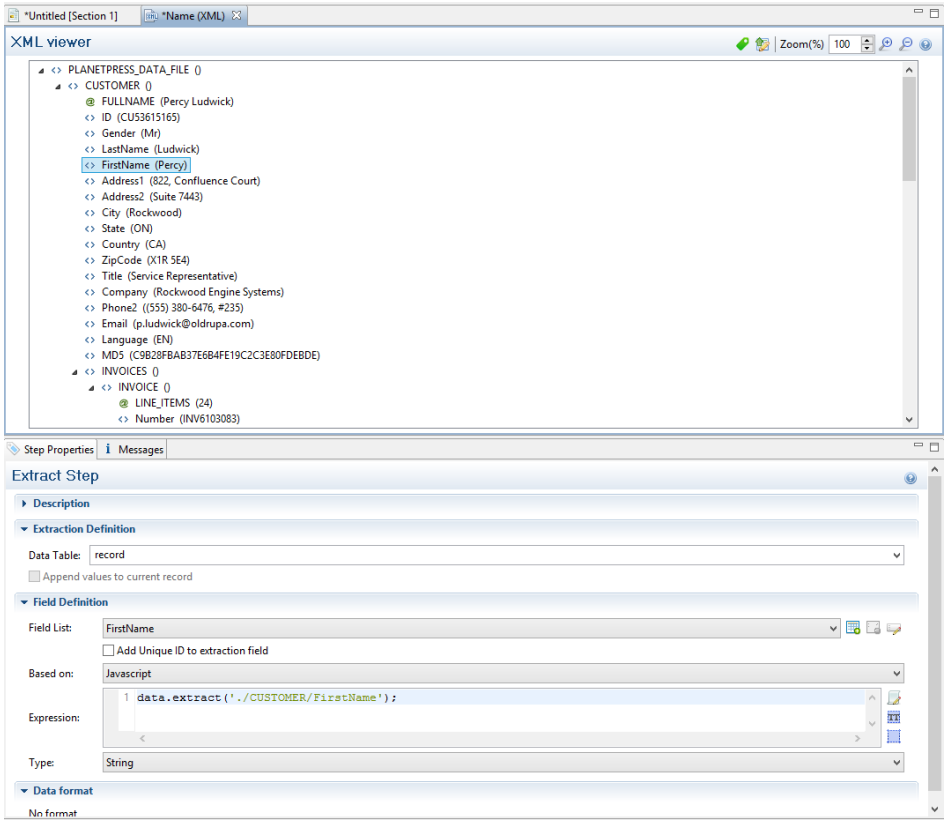
Extracts the text value of the specified node in an **XML** file.

xPath

String that can be relative to the current location or absolute from the start of the record.

Example

The script command **data.extract('./CUSTOMER/FirstName');** means that the extraction is made on the **FirstName** node under **Customer**.



extract(columnName, rowOffset)

Extracts the text value from the specified column and row in a **CSV/XLS/XLSX** file.

The column is specified by name. To extract data from a column specified by index, use "extractByIndex(index, rowOffset)" on page 553.

columnName

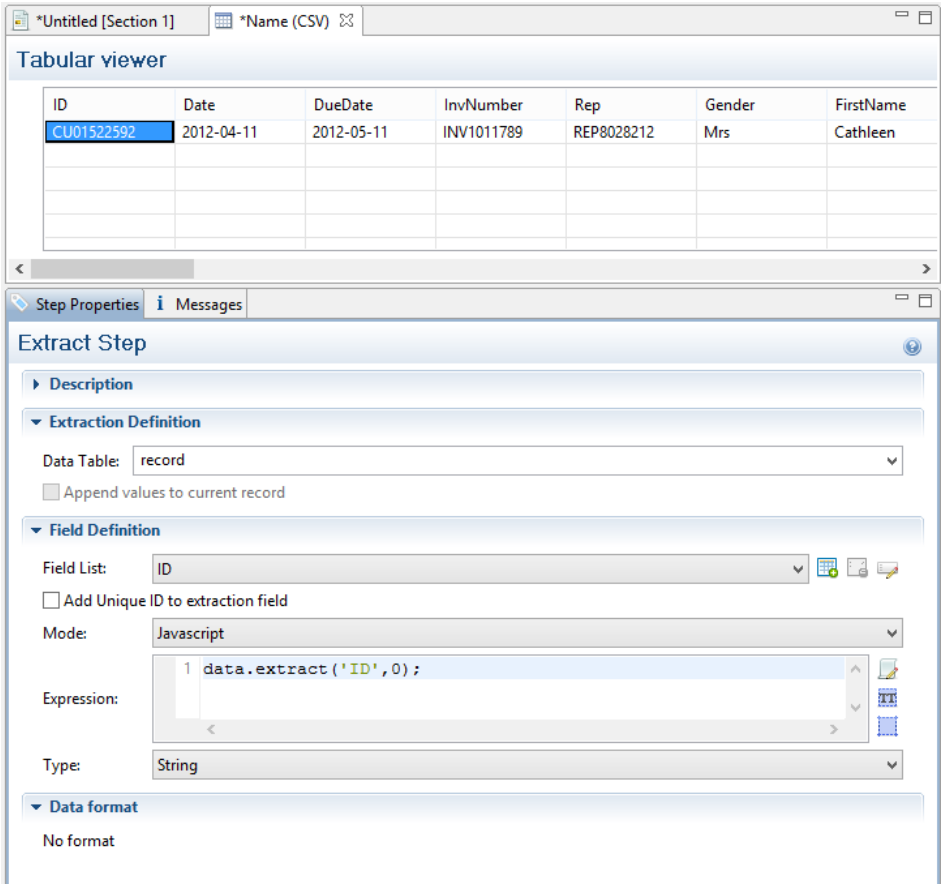
String that represents the column name.

rowOffset

Number that represents the row index (zero-based), relative to the current position. To extract the current row, specify 0 as the rowOffset. Use moveTo() to move the pointer in the source data file (see "moveTo()" on page 569).

Example

The script command **data.extract('ID',0);** means that the extraction is made on the **ID** column in the first row.



extract(left, right, verticalOffset, lineHeight, separator)

Extracts the text value from a rectangular region in a **PDF** file. All coordinates are expressed in millimeters.

left

Double that represents the distance from the left edge of the page to the left edge of the rectangular region.

right

Double that represents the distance from the left edge of the page to the right edge of the rectangular region.

verticalOffset

Double that represents the distance from the current vertical position.

lineHeight

Double that represents the total height of the region.

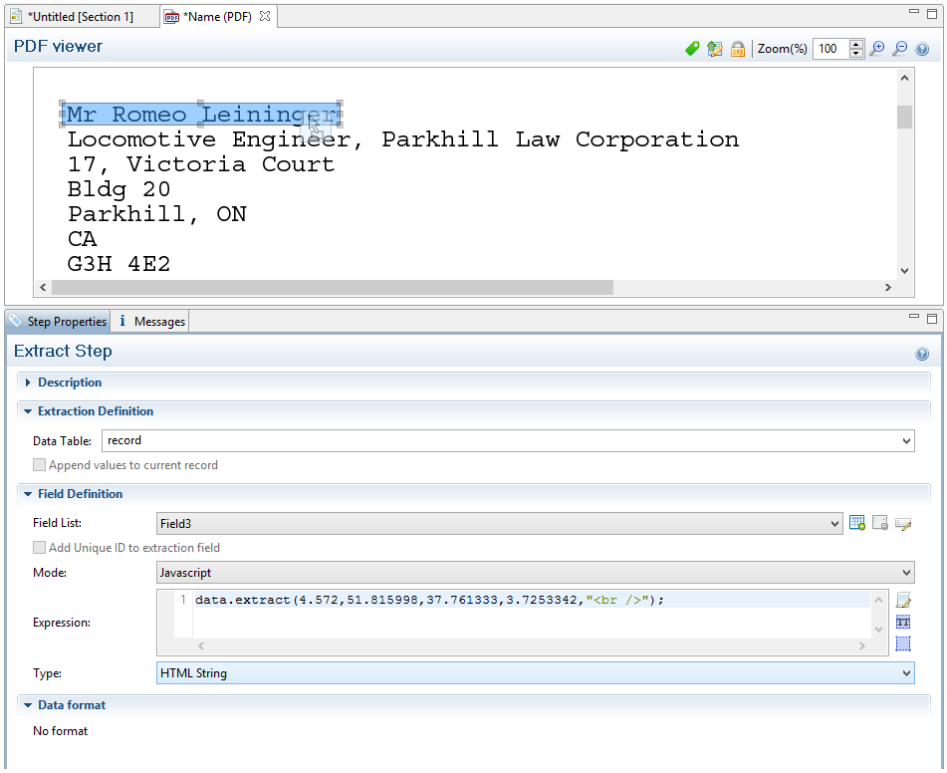
separator

String inserted between all lines returned from the region. If you don't want anything to be inserted between the lines, specify an empty string ("").

> **Tip:** "<br/>" is a very handy string to use as a separator. When the extracted data is inserted in a Designer template, it will be interpreted as a line break, because <br/> is a line break in HTML and Designer templates are actually HTML files.

Example

The script command **data.extract(4.572,51.815998,37.761333,3.7253342,"<br />");** means that the left position of the extracted information is located at 4.572mm, the right position at 51.815998mm, the vertical offset is 37.761333mm and the line height is 3.7253342mm. Finally, the "<br/>" string is used for concatenation.



extract(jPath)

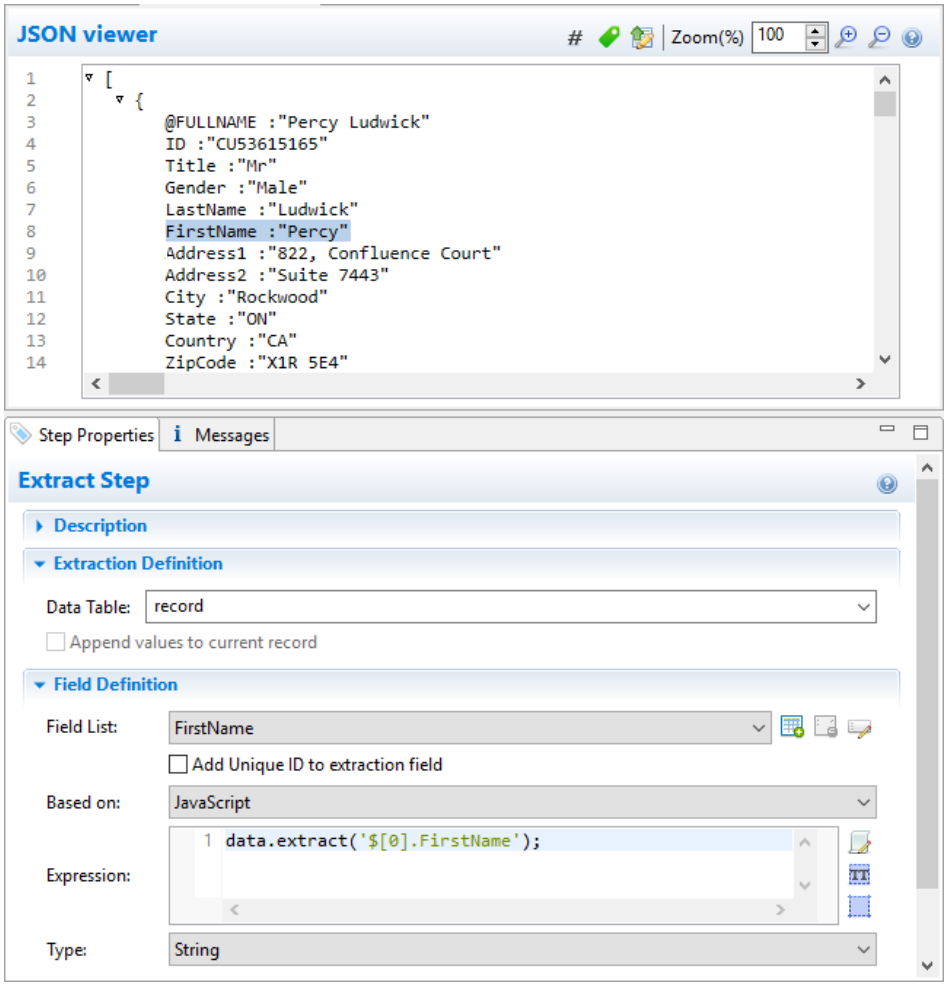
Extracts the text value of the specified element in a **JSON** file.

jPath

JsonPath expression (String) that can be relative to the current location or absolute from the start of the record. See also: "JsonPath" on page 395.

Example

The script command **data.extract('$[0].FirstName');** means that the extraction is made on the **FirstName** element found in the first element in the array at the root.



Note that in order to access an extracted object or array in script, the extracted value has to be parsed, for example:

```
var myData = JSON.parse(data.extract('$[0]'));
```

extractByIndex(index, rowOffset)

Extracts the value from the specified column and row.

This function can be used to extract data from CSV or XLS(X) files that have an identical structure but don't have the same column names.

index

Number that represents a column in a CSV or XLS(X) file (1-based).

rowOffset

Optional. Number that represents the row index (zero-based), relative to the current position. To extract the current row, specify 0 as the rowOffset. Use moveTo() to move the pointer in the source data file (see "moveTo()" on page 569). When omitted, the current row will be extracted.

Examples

The script command `data.extractByIndex(1,0);` means that the extraction is made on the first column in the first row, relative to the current position.

The following script puts the values of all columns of the current row into one array. The values could then be stored as a JSON array in a single extracted field.

```
var allFields=[];
var i=1;
while (data.fieldExistsByIndex(i)) {
   allFields.push(data.extractByIndex(i++));
}
```

extractMeta()

Method that extracts the value of a metadata field on a certain level in a PDF/VT. This method always return a String.

extractMeta(levelName String, propertyName String)

levelName

String, specifying a level in the PDF/VT or AFP. Case sensitive.

propertyName

String, specifying the metadata field.

fieldExists()

Method of the `data` object that returns **true** if a certain metadata field, column or node exists. (See "data" on page 546.)

fieldExists(levelName, propertyName)

This method returns **true** if the given metadata field exists at the given level in a **PDF/VT** or **AFP** file.

levelName

String, specifying a level in the PDF/VT or AFP file.

propertyName

String, specifying the metadata field.

fieldExists(fieldName)

This method returns **true** if a column with the specified name exists in the current record in a **CSV**, **XLS** or **XLSX** file.

To verify whether a column specified by index exists in a CSV, XLS or XLSX file, use "fieldExistsByIndex(index)" below.

fieldName

String that represents a field name (column) in a CSV, XLS or XLSX file.

fieldExists(xPath)

This method returns **true** if the specified node exists in the current record in an **XML** file.

xPath

String that specifies a node.

fieldExists(jPath)

This method returns **true** if the specified element exists in the current record in a **JSON** file.

jPath

String that specifies an element.

fieldExistsByIndex(index)

Method of the `data` object that returns **true** if a column, specified by index, exists in a **CSV**, **XLS** or **XLSX** file.

index

Number that represents a column in a CSV or XLS(X) file (1-based).

Example

The following script accesses all columns in a CSV file:

```
var allFields=[];
var i=1;while (data.fieldExistsByIndex(i)) {
   allFields.push(data.extractByIndex(i++));
}
```

See also: "extractByIndex(index, rowOffset)" on page 553.

find()

Method of the data object that finds the first occurrence of a string starting from the current position.

find(stringToFind, leftConstraint, rightConstraint)

Finds the first occurrence of a string starting from the current position. The search can be constrained to a series of characters (in a text file) or to a vertical strip (in a PDF file) located between the given constraints.
The method returns `null` if the string cannot be found. Otherwise it returns a RectValueText (if the data source is a text file) or RectValuePDF (if the data source is a PDF file) object. This object contains the absolute Left, Top, Right and Bottom coordinates of the smallest possible rectangle that completely encloses the first occurrence of the string. The coordinates are expressed in a number of characters if the data source is a text file, or in millimetres if the data source is a PDF file.

Partial matches are not allowed. The entire string must be found between the two constraint parameters.

The `data.find()` function only works on the current page. If the record contains several pages, you must create a loop that will perform a jump from one page to another to do a `find()` on each page.

> **Note:** Calling this method does not move the current position to the location where the string was found. This allows you to use the method as a look-ahead function without disrupting the rest of the data mapping workflow.

stringToFind

String to find.

leftConstraint

Number indicating the left limit from which the search is performed. This is expressed in characters for a text file, or in millimetres for a PDF file.

rightConstraint

Number indicating the right limit to which the search is performed. This is expressed in characters for a text file, or in millimetres for a PDF file.

Examples

To look for the word "text" on an entire Letter page (8 1/2 x 11 inch), the syntax is:

```
data.find("text", 0, 216);
```

The numbers 0 and 216 are in millimeters and indicate the left and right limits (constraints) within which the search should be performed. In this example, these values represent the entire width of a page. Note that the smaller the area is, the faster the search is. So if you know that the word "text" is within 3 inches from the left edge of the page, provide the following:

```
data.find("text", 0, 76.2); //76.2mm =  3*25.4 mm
```

The return value of the function is:

```
Left=26,76, Top=149.77, Right=40,700001, Bottom=154.840302
```

These values represent the size of the rectangle that encloses the string in full, in millimeters relative to the upper left corner of the current page.

findRegExp()

Finds the first occurrence of a string that matches the given regular expression pattern, starting from the current position.

findRegExp(regexpToFind, flags, leftConstraint, rightConstraint): rectValueText)

Finds the first match for a given regular expression pattern starting from the current position. Regular expression flags (**i,s,L,m,u,U,d**) are specified in the flags parameter. The search can be constrained to a vertical column of characters located between the left and right constraint, each expressed in characters (in a text file) or millimeters (in a PDF file). Partial matches are not allowed. The entire match for the regular expression pattern must be found between the two constraints.

The method returns `null` if the regular expression produces no match. Otherwise it returns a RectValueText object, containing the Left, Top, Right and Bottom coordinates - expressed in characters (in a text file) or millimeters (in a PDF file), relative to the upper left corner of the current page - of the smallest possible rectangle that completely encloses the first match for the regular expression.

Note: Calling this method does not move the current position to the location where the match occurred. This allows you to use the method as a look-ahead function without disrupting the rest of the data mapping workflow.

Note: This function evaluates the regular expression pattern for each individual line. Multiline regular expression patterns, i.e., patterns with tokens such as line end tokens (\n) are not supported.

regexpToFind

Regular expression pattern to find.

flags

- **i**: Enables case-insensitive matching. By default, case-insensitive matching assumes that only characters in the US-ASCII charset are being matched. Unicode-aware case-insensitive matching can be enabled by specifying the UNICODE_CASE flag (**u**) in conjunction with this flag.
- **s**: Enables dotall mode. In dotall mode, the expression **.** matches any character, including a line terminator. By default this expression does not match line terminators.
- **L**: Enables literal parsing of the pattern. When this flag is specified, then the input string that specifies the pattern is treated as a sequence of literal characters. Metacharacters or escape sequences in the input sequence will be given no special meaning. The CASE_INSENSITIVE (**i**) and UNICODE_CASE (**u**)flags retain their impact on matching when used in conjunction with this flag. The other flags become superfluous.
- **m**: Enables multiline mode. In multiline mode, the expressions ^ and $ match just after or just before, respectively, a line terminator or the end of the input sequence. By default, these expressions only match at the beginning and the end of the entire input sequence.
- **u**: Enables Unicode-aware case folding. When this flag is specified, then case-insensitive matching, when enabled by the CASE_INSENSITIVE flag (**i**), is done in a manner consistent with the Unicode Standard. By default, case-insensitive matching assumes that only characters in the US-ASCII charset are being matched.
- **U**: Enables the Unicode version of Predefined character classes and POSIX character classes. When this flag is specified, then the (US-ASCII only) Predefined character classes and POSIX character classes are in conformance with Unicode Technical Standard #18: Unicode Regular Expression Annex C: Compatibility Properties.
- **d**: Enables Unix lines mode. In this mode, only the '\n' line terminator is recognized in the behavior of ., ^, and $.

leftConstraint

Number indicating the left limit from which the search is performed. This is expressed in characters for a text file, or in millimeters for a PDF file.

rightConstraint

Number indicating the right limit to which the search is performed. This is expressed in characters for a text file, or in millimeters for a PDF file.

Examples

```
data.findRegExp(/\d{3}-[A-Z]{3}/,"i",50,100);
```

or

```
data.findRegExp("\\d{3}-[A-Z]{3}","i",50,100);
```

Both expressions would match the following strings: 001-ABC, 678-xYz.

Note how in the second version, where the regular expression is specified as a string, some characters have to be escaped with an additional backslash, which is standard in JavaScript.

db

Object that allows to connect to a database.

Methods

The following table describes the methods of the `db` object.

Method	Description	Available in	File type
"connect()" below	Method that returns a new database connection object.	Boundaries Preprocessor, Extract, Condition, Repeat, Action, and Postprocessor steps	all

connect()

Method that returns a new database connection object.

connect(url, user, password)

This method returns a new database Connection object after connecting to the given URL and authenticating the connection with the provided user and password information.
The function accepts arguments as described here: https://docs.oracle.com/javase/8/docs/api/java/sql/DriverManager.html#getConnection-java.lang.String-java.lang.String-java.lang.String-.
In the returned Connection object normally any public method should be available. The returned Connection object is described here: https://docs.oracle.com/javase/8/docs/api/java/sql/Connection.html.

Note: Make sure to close any Connection object created by `connect()` and any other closable resources created from the Connection instance (ResultSet, etc.).

url

String that represents a database url of the form jdbc:subprotocol:subname, e.g. 'jdbc:mysql://localhost:3306/mycompany'.
The syntax will be specific to the database, or more precisely the JDBC connector.
Currently Datamapper supports the following JDBC connectors:

- com.mysql.cj.jdbc.Driver
- sun.jdbc.odbc.JdbcOdbcDriver
- com.microsoft.sqlserver.jdbc.SQLServerDriver
- oracle.jdbc.OracleDriver

For older databases that only have an ODBC driver, a JDBC-ODBC bridge can be called using the **jdbc:olodbc** prefix.
The syntax is: `db.connect("jdbc:olodbc:...", ..., ...);`

Note that the JDBC-ODBC bridge can only execute a SELECT request; no write actions are allowed. For supported objects and methods of the returned `Connection` object, see JDBC-ODBC bridge: objects and methods.

This bridge replaces the legacy bridge which was deprecated. The legacy bridge can still be selected with `jdbc:odbc:`, and its functionality remains unchanged.

user

String that represents the name of the database user on whose behalf the connection is being made. This is used for authentication.

password

String that represents the user's password. This is used for authentication.

Example

Here's an example of how to connect to a mySQL database and retrieve a data value.

```
con = db.connect('jdbc:mysql://localhost:3306/mycompany','root','admin');
statement = con.createStatement();
values = statement.executeQuery("select * from datavalue");
if (values.next())
    values.getString("data");
else
    'nothing';
values.close();
statement.close();
con.close();
```

logger

Global object that allows logging messages such as error, warning or informational messages.

Methods

The following table describes the methods of the logger object.

Method	Parameters	Description
error()	message: String	Logs an error message
info()	message: String	Logs an informational message
warn()	message: String	Logs a warning message

record

The current record in the main data set.

Properties

Property	Description
copies	The total number of copies of the current record that must be created. By default, this is 1. This value is used when the record is saved, at the end of the data mapping process for each record.
fields	The field values that belong to this record. You can access a specific field value using either a numeric index or the field name.
index	The one-based index of this record, or zero if no data is available.*
tables	The detail tables that belong to this record. You can access a specific table using a numeric index or the table name; see "table" on page 573.

*The current **record index** and/or the total number of records in the record set can be displayed in a document using a Handlebars helper: "counter" on page 1010.
Also see the how-to: Handlebars helper: counter.

Methods

Note: These methods cannot be used in post-processor scripts, because they operate on the current record *before* it is written to the database.

Method	Description
"set(record)" below	Sets field values in the current record.
"setCopy(i, record)" on the next page	Sets field values in a copy of the current record.

set(record)

This method of the `record` object sets field values in the current record in the main data set, or in a row in a detail table. (See also: "record" on page 565.)

Note: This method cannot be used in post-processor scripts, because it operates on the current record *before* it is written to the database.

record

The mandatory `record` parameter is a JavaScript object that contains one or more fields specified in the Data Model.
The `record` parameter may contain a subset of the fields in the Data Model. Only the fields included in the `record` parameter are updated in the database, while the contents of all other fields remain unchanged.
The call fails if the parameter is omitted or empty, if any of the fields specified in the `record` doesn't exist in the Data Model, or if a value cannot be converted to the data type that is expected in a field.

About data types

Where possible, values are automatically converted into the data type of the respective data field.

Note: Dates must be passed as a Date object to allow them to be extracted into a Date field. See Date in the Mozilla help files.

Passing an improper data type triggers an error. For instance the following objects are all invalid:

`{ myBoolean : "true" }` - The myBoolean field is boolean and expects a boolean, not a string

`{ myDate : "2021-03-29" }` - The myDate field is a date and expects a Date object: `myDate: new Date(2021,2,29)`, not a string

`{ myPageCount : 2.5 }` - The myPageCount field is an integer and expects a number without any decimals

`{ myPrice : "19.99" }` - The myPrice field is a currency and expects a number value, not a string

Example

```
record.set({customerName : "John Doe", customerAddress : "123 test road"});
```

The following example adds a row to a detail table and sets some values in it.

```
var row = record.tables.myDescriptions[i].addRow();
record.tables.myDescriptions[row].set({customerName : "John Doe", customerAddress : "123 test road"});
```

setCopy(i, record)

This method of the `record` object sets field values in a copy of the current record in the main data set. (See also: "record" on page 565.)

> **Note:** This method cannot be used in post-processor scripts, because it operates on the current record *before* it is written to the database.

i

The index of the record copy (0-based; the original record has an index of 0).

record

The mandatory `record` parameter is a JavaScript object that contains one or more fields specified in the data model at the root level.
The `record` parameter may contain a subset of the fields in the Data Model. Only the fields included in the `record` parameter are updated in the database, while the contents of all other fields remain unchanged.
The call fails if the parameter is omitted or empty, if any of the fields specified in the

`record` doesn't exist in the Data Model, or if a value cannot be converted to the data type that is expected in a field.

> **Note:** In order to change values in the record copies, the fields to modify must either have a default value or have an extraction task earlier in the process that actually assigns a value to them.

About data types

Where possible, values are automatically converted into the data type of the respective data field.

> **Note:** Dates must be passed as a Date object to allow them to be extracted into a Date field. See Date in the Mozilla help files.

Passing an improper data type triggers an error. For instance the following objects are all invalid:

`{ myBoolean : "true" }` - The myBoolean field is boolean and expects a boolean, not a string

`{ myDate : "2021-03-29" }` - The myDate field is a date and expects a Date object: `myDate: new Date(2021,2,29)`, not a string

`{ myPageCount : 2.5 }` - The myPageCount field is an integer and expects a number without any decimals

`{ myPrice : "19.99" }` - The myPrice field is a currency and expects a number value, not a string

Example

```
record.copies = 3;var oneRec = {CopyFor:'',Note:''};

for(var i=0;i<record.copies;i++){
  switch (i) {
    case 0:
      oneRec.CopyFor = "Customer";
      oneRec.Note = "Payment due in 30 days";
      break;
    case 1:
      oneRec.CopyFor = "Administration";
      oneRec.Note = "Due date set to 30 days";
      break;
    case 2:
      oneRec.CopyFor = "Archive";
  }
  record.setCopy(i, oneRec);
}
```

record

The current record in the main data set.

Properties

Property	Description
copies	The total number of copies of the current record that must be created. By default, this is 1. This value is used when the record is saved, at the end of the data mapping process for each record.
fields	The field values that belong to this record. You can access a specific field value using either a numeric index or the field name.
index	The one-based index of this record, or zero if no data is available.*
tables	The detail tables that belong to this record. You can access a specific table using a numeric index or the table name; see "table" on page 573.

*The current **record index** and/or the total number of records in the record set can be displayed in a document using a Handlebars helper: "counter" on page 1010.
Also see the how-to: Handlebars helper: counter.

Methods

Note: These methods cannot be used in post-processor scripts, because they operate on the current record *before* it is written to the database.

Method	Description
"set(record)" on page 562	Sets field values in the current record.
"setCopy(i, record)" on page 563	Sets field values in a copy of the current record.

region

The region object defines a sub-section of the input data. Its properties vary according to the type of data.

This object is available when triggering document boundaries **On script**, with all file types; see "Setting boundaries using JavaScript" on page 535.

Properties

The following table describes the properties and methods of the `region` object.

Property/method	Description	Return Type
found	Field that contains a boolean value indicating if the last call to `boundaries.find()` was successful. Since the `find()` method always returns a region, regardless of search results, it is necessary to examine the value of `found` to determine the actual result of the operation.	Boolean
"range" on the next page	Read-only object containing the physical coordinates of the region.	Physical location of the region: x1 (left), y1 (top), x2 (right), y2 (bottom), expressed in characters for a text file or in millimeters for a PDF file. For a CSV file, it is the name of the column that defines the region.

Methods

The following table describes the methods of the `region` object.

Property/method	Description	Return Type
"createRegion()" below	Creates a `region` by setting the physical coordinates of the region object.	A `region` that has the specified coordinates.

createRegion()

This method sets the physical coordinates of the `region` object. The `region` is available when setting document boundaries using a script (see "region" on the previous page).

PDF and Text: createRegion(x1, y1, x2, y2)

Creates a region from the data, using the specified **left** (x1), **top** (y1), **right** (x2) and **bottom** (y2) parameters, expressed in characters for a text file or in millimeters for a PDF file.

x1

Double that represents the left edge of the region.

y1

Double that represents the top edge of the region.

x2

Double that represents the right edge of the region.

y2

Double that represents the bottom edge of the region.

Example

The following script attempts to match `((n,m))` or `((n))` against any of the strings in the specified region and if it does, a document boundary is set.

```
var myRegion = region.createRegion(170,25,210,35);
var regionStrings=boundaries.get(myRegion);
if (regionStrings) {
    for (var i=0;i<regionStrings.length; i++) {
        if (regionStrings[i].match(/(\({2}n,*m*\){2})/gi)){
            boundaries.set();
        }
    }
}
```

(The `match()` function expects a regular expression; see String match.)

CSV or database: createRegion(columnName)

Creates a region from the data in a CSV file, using the specified **columnName** parameter.

columnName

String containing the name of the column where the region is to be created.

Example

This script checks the first value in a certain column. If it is not the same value as in the previous record(s), a document boundary is set.

```
var recordValue = boundaries.get(region.createRegion('ID'))[0];
if(!(recordValue==boundaries.getVariable('lastValue'))){
    boundaries.setVariable('lastValue',recordValue);
    boundaries.set(0);
}
```

range

Read-only object containing the physical coordinates of the region (see "region" on page 565).

- For a **text file**, the range() method contains the physical coordinates of the region: **x1** (left), **y1** (top), **x2** (right), **y2** (bottom), expressed in **characters**.
- For a **PDF file**, the range() method contains the physical coordinates of the region: **x1** (left), **y1** (top), **x2** (right), **y2** (bottom), expressed in **millimeters**.
- For a **CSV file**, the `range` contains the name of the column that defines the region.

sourceRecord

Returns a **sourceRecord** object containing custom properties specific to the current source record being processed.

These are the custom properties defined in the **Preprocessor** step that have their Scope set to "Each record". See: "Properties and runtime parameters" on page 387.

Properties

```
sourceRecord.properties.property;
```

Property	Description
properties	Returns an array of properties defined in the Preprocessor step with the Record Scope (i.e. dynamically reset with each new record).

steps

Returns a `steps` object encapsulating properties and methods pertaining to the current DataMapper process.

This object is available in an Extract, Condition, Repeat or Multiple Conditions step script.

Methods and properties

The following table lists the methods and properties of the `steps` object. These are available in Extract, Condition, Repeat, and Action steps, depending on the file type.

Method	Description	File type
currentPosition	Returns the current position of the pointer in the data. Depending on the type of data being processed, the return value may be a string (e.g. XPath value in XML), an integer (e.g. line numbers in text ot tabular data), or a measure in millimeters(e.g. PDF data).	All
currentLoopCounter	An integer value representing the current iteration of the containing loop. When loops are nested, you have access to the iteration for the current loop but not to any of the parent loops. **Note:** This variable is a counter so it starts at **1** as opposed to an index which usually starts at 0.	All
"currentPage" on the next page	Returns an integer value representing the current page where the current position is located, inside the current record.	Text, PDF
currentPageHeight	The height of the current page in millimeters.	PDF

Method	Description	File type
currentPageWidth	The width of the current page in millimeters.	PDF
lines	An integer value representing the number of lines in the current record of data.	CSV, TEXT
"moveTo()" below	Moves the pointer in the source data file to another position.	All
"moveToNext()" on page 571	Moves the position of the pointer in the source data file to the next line, row or node. The behavior and arguments are different for each emulation type: text, PDF, tabular (CSV), or XML.	All
totalPages	An integer value representing the total number of pages inside the current record.	Text, PDF

Example

```
if(steps.currentPage > curPage) {
    steps.moveTo(0, steps.currentPosition+14);
    /* Moves the current position to 14 lines below the current position of the pointer in the data */
    curPage++;
} else if(curLine.startsWith("LOAD FACTOR")) {
/* Extracts data to the curLine variable until the string "LOAD FACTOR" is encountered */
    break;
} else {
    lineArray.push(curLine);
    /* Adds the current line value (extraction) to the array */
}
```

currentPage

Property of the `steps` object (see "steps" on the previous page) containing an integer value that represents the page where the pointer is located, inside the current record.

In this example, an extraction area is assigned to the variable **curLine** each time the current page value has changed.

```
curPage = steps.currentPage;
for(i=0;i<100;i++) {
    if(steps.currentPage > curPage) {
        let curLine = data.extract(51, 88, 0, 1, "").trim();
        curPage++;
    }
}
```

moveTo()

Moves the position of the pointer in the source data file. This is a method of the steps object (see "steps" on the previous page).

moveTo(scope, verticalPosition)

Moves the current position in a **text file** to verticalPosition where the meaning of verticalPosition changes according to the value specified for scope.

scope

Number that may be set to: 0 or steps.MOVELINES1 or steps.MOVEDELIMITERS2: next line with content

verticalPosition

Number. What it represents depends on the value specified for scope.

With the scope set to 0 or steps.MOVELINES, verticalPosition represents the **index of the line** to move to **from the top of the record**.

With the scope set to 1 or steps.MOVEDELIMITERS, verticalPosition represents the **index of the delimiter** (as defined in the Input Data settings) to move to **from the top of the record**.

With the scope set to 2, verticalPosition is not used. The position is moved to the next line after the current position that contains any text.

Example

The following line of code moves the current position in a text file 14 lines down from the current vertical position (steps.currentPosition) of the pointer in the data, as long as it is on the same page.

```
if(steps.currentPage > curPage) {
    steps.moveTo(0, steps.currentPosition+14);
    curPage++;
}
```

moveTo(scope, verticalOffset)

Moves the current position in a **PDF file** to verticalOffset where the meaning of verticalOffset changes according to the value specified for scope.

scope

Number that may be set to:

- 0 or`steps.MOVEMEASURE`
- 1 or`steps.MOVEPAGE`

verticalOffset

Double. What it represents depends on the value specified for scope.

With the scope set to 0 or steps.MOVEMEASURE, verticalOffset represents the number of **millimeters** to move the current position, relative to the top of the record (NOT the top of the current page).

With the scope set to 1 or steps.MOVEPAGES, verticalOffsetrepresents the **index of the target page**, relative to the top of the record.

moveTo(xPath)

Moves the current position in a XML file to the first instance of the given node, relative to the top of the record.

xPath

String that defines a node in the XML file.

> **Tip:** The **XML elements** drop-down (on the Settings pane, under Input Data) lists xPaths defining nodes in the current XML file.

moveTo(row)

Moves the current position in a **CSV file** to the given row number.

row

Number that represents the index of the row, relative to the top of the record.

moveTo(jPath)

Moves the current position in a JSON file to the first instance of the given element, relative to the top of the record if the jPath is absolute (starts with **$** which is the root) or relative to the current position if the jPath is relative (starts with **.** which is the current element).

jPath

JsonPath expression (String) that defines the path to an element in the JSON file. See also: "JsonPath" on page 395.

> **Tip:** The full JsonPath to an element is displayed at the bottom left of the window when you select it. To copy the path, right-click it and select Copy.

moveToNext()

Moves the position of the pointer in the source data file to the next line, row or node. The behavior and arguments are different for each emulation type: text, PDF, tabular (CSV), or XML.

This is a method of the steps object (see "steps" on page 568).

moveToNext(scope)

Moves the current position in a **text file** or **XML file** to the next instance of `scope`. What `scope` represents depends on the emulation type: text or XML.

Text

scope

Number that may be set to:

- 0 or `steps.MOVELINES`: the current position is set to the next line.
- 1 or `steps.MOVEDELIMITERS`: the current position is set to the next delimiter (as defined in the Input Data settings).
- 2 (next line with content): the current position is set to the next line that contains any text.

Example: The following line of code moves the current position to the next line that contains any text.

```
steps.moveToNext(2);
```

XML

scope

Number that may be set to:

- 0 or `steps.MOVENODE`: The current position is set to the next node in the XML, regardless of its level in the hierarchy; the next node may be a child element, a sibling node, or a node that is higher in the XML hierarchy than the current node.
- 1 or `steps.MOVESIBLING`: the current position is set to the next sibling node in the XML hierarchy.

JSON

Moves the current position to the next element in the JSON hierarchy based on the specified scope.

scope

Number that may be set to:

- 0 or `steps.MOVENODE`: the current position is set to the first available next element element in the JSON. The next element is looked for in the following order: first child, next sibling, parent's sibling.
- 1 or `steps.MOVESIBLING`: the current position is set to the next element in the same parent, i.e. the next element in an array or the next key-value pair in an object.

moveToNext(left, right)

Moves the current position in a **PDF file** to the next line that contains any text, the search for text being contained within the **left** and **right** parameters, expressed in millimeters.

left

Double that represents the left edge (in millimeters) of the text to find.

right

Double that represents the right edge (in millimeters) of the text to find.

moveToNext()

Moves the current position in a **CSV file** to the next row, relative to the current position.

table

The `table` object holds a detail table that exists in a record.
The detail table is retrieved by name, using `record.tables.<table>`. For example: `record.tables.myDetailTable`.

Properties

Property	Description
length	Returns the count of rows in the detail table.

Methods

Method	Description
"addRow(record)" on the facing page	Adds a row to the detail table.
"set(record)" on page 575	Sets field values in an existing detail table row.

addRow(*record*)

This method of the `table` object adds a record to an existing detail table (see "table" on the previous page). The detail table must already exist in the data model, otherwise the call fails.

> **Note:** Until OL Connect 2023.2 the **addRow()** functionality did not work for detail tables that were manually added to the Data Model. It now does, but it is important to note that any detail table without extracted data is recognized as an empty table, with a length of 0.
>
> To check if a detail table exists and contains any data in the post-2023.2 version, first perform a check using the `in` operator, then perform a check on the table's length to determine if it contains any records. For example:
>
> ```
> if (detailTable in record.tables) {
>  if (record.tables.detailTable.length != 0)
>  { ... }
> }
> ```

The call returns the **index** of the newly added record.

record

The optional `record` parameter is a JavaScript object that contains one or more fields specified in the Data Model for this detail table.
The `record` parameter may contain a subset of the fields in the Data Model. Only the fields included in the `record` parameter are updated in the database, while the contents of all other fields remain unchanged.
The call fails if the parameter is omitted or empty, if any of the fields specified in the `record` doesn't exist in the Data Model, or if a value cannot be converted to the data type that is expected in a field.

About data types

Where possible, values are automatically converted into the data type of the respective data field.

> **Note:** Dates must be passed as a Date object to allow them to be extracted into a Date field. See Date in the Mozilla help files.

Passing an improper data type triggers an error. For instance the following objects are all invalid:

`{ myBoolean : "true" }` - The myBoolean field is boolean and expects a boolean, not a string

`{ myDate : "2021-03-29" }` - The myDate field is a date and expects a Date object: `myDate: new Date(2021,2,29)`, not a string

`{ myPageCount : 2.5 }` - The myPageCount field is an integer and expects a number without any decimals

`{ myPrice : "19.99" }` - The myPrice field is a currency and expects a number value, not a string

Example

This script adds an empty row to a detail table called 'myDescriptions'.

```
row = record.tables.myDescriptions.addRow();
```

set(record)

The `set()` method of the `table` object sets field values in an existing detail table row. The row is specified by its index in the detail table:

```
record.tables.<table>[index].set(<record>)
```

If the index is invalid, the call fails.

record

The mandatory `record` parameter is a JavaScript object that contains one or more fields specified in the data model for the detail table.
The `record` parameter may contain a subset of the fields in the Data Model. Only the fields included in the `record` parameter are updated in the database, while the contents of all other fields remain unchanged.
The call fails if the parameter is omitted or empty, if any of the fields specified in the `record` doesn't exist in the Data Model, or if a value cannot be converted to the data type that is expected in a field.

About data types

Where possible, values are automatically converted into the data type of the respective data field.

Note: Dates must be passed as a Date object to allow them to be extracted into a Date field. See Date in the Mozilla help files.

Passing an improper data type triggers an error. For instance the following objects are all invalid:

`{ myBoolean : "true" }` - The myBoolean field is boolean and expects a boolean, not a string

`{ myDate : "2021-03-29" }` - The myDate field is a date and expects a Date object: `myDate: new Date(2021,2,29)`, not a string

`{ myPageCount : 2.5 }` - The myPageCount field is an integer and expects a number without any decimals

`{ myPrice : "19.99" }` - The myPrice field is a currency and expects a number value, not a string

Examples

This script sets the 'customerName' and 'customerAddress' fields of a detail table row.

```
record.tables.details[3].set({customerName : "John Doe", customerAddress : "123 test road"});
```

The following script adds an empty row to the table and then extracts some data into the Description field of the new row.

```
row = record.tables.myDescriptions.addRow();
record.tables.myDescriptions[row].set( {Description : data.extract((52+i*12),(66+i*12),0,1,"<br>")} );
```

Functions

copyFile()

Function that copies a file to the target file path, replacing it if it already exists.

copyFile(source, target)

source

String that specifies the source file path and name.

target

String that specifies the target file path and name.

Example

This script copies the file test.txt from c:\Content into the c:\out folder.

```
copyFile("c:\Content\test.txt","c:\out\")
```

createGUID()

This function returns a unique 36-character string consisting of 32 alphanumeric, lower case characters and four hyphens.

Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (8-4-4-4-12 characters).|
Example: 123e4567-e89b-12d3-a456-426655440000.

The function produces unique strings on each and every call, regardless of whether the call occurs within the same data mapper or not, or on concurrent threads.

createHTTPRequest()

Function that creates a new ScriptableHTTPRequest object, in order to issue REST/AJAX calls to external servers and fetch data through AJAX calls.

This function allows the data mapping process to complement its extraction process with external data, including data that could be provided by an HTTP process in Workflow, for instance a process that retrieves certain values from Workflow's Data Repository. Another possible use is to have a Postprocessor that writes the results of the extraction process to a file and immediately uploads that file to a Workflow process.

The returned ScriptableHTTPRequest has a selection of the properties and methods of the standard JavaScript XMLHTTPRequest object (see https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest).
Supported properties and methods are listed below.

> **Note:** It is not possible to use the **async** mode, which can be set via the `open()` function of the ScriptableHTTPRequest (see https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open) in a data mapping configuration. Async-related properties and methods of the ScriptableHTTPRequest object - for example .onreadystatechange, .readyState and .ontimeout - are **not supported**.
> The reason for this is that by the time the response comes back from the server, the DataMapper script may have finished executing and gone out of scope.

Supported properties

- response
- status

- statusText
- timeout (ms). Default: 1 minute.

Supported methods

create()	Creates a new instance of ScriptableHTTPRequest.
• open(String method, String url, String user, String password) • open(String verb, String url, String userName, String password, String[] headers, String[] headervalues, String requestBody)	Opens a HTTP request. **Note:** If you don't use a user name and password, pass empty strings: `request.open ("GET",url,"","");`
• send() • send(String requestBody)	Sends an HTTP request and returns the HTTP status code. Blocked call.
getResponseHeader(String header)	Gets the ResponseHeader by name.
getResponseHeaders()	Returns the full response headers of the last HTTP request.
getRequestBody()	Gets the HTTP request body (for POST and PUT).
setRequestHeader(String requestHeader, String value)	Adds an additional HTTP request header.
getResponseBody()	Returns the full response body of the last HTTP request.
setRequestBody(String requestBody)	Sets the HTTP request body (for POST and PUT).
getPassword()	Gets the password for HTPP basic authentication
setPassword(String password)	Sets the password for HTPP basic authentication
getTimeout()	Gets the time to wait for the server's response
setTimeout(int timeout)	Sets the time (in ms.) to wait for the server's response.
getUsername()	gets the username for basic HTTP authentication.
setUsername(String userName)	sets the username for basic HTTP authentication
abort()	Aborts the request.

createTmpFile()

Function that creates a file with a unique name in the temporary work folder and returns a **file** object. This file stores data temporarily in memory or in a buffer. It is used to prevent

multiple input/output access to a physical file when writing. In the end, the contents are transferred to a physical file for which only a single input/output access will occur.

In the following script, the contents of the data sample file are copied in uppercase to a temporary file.

```
try{
    // Open a reader
    var reader = openTextReader(data.filename);
    // Create a temporary file
    var tmpFile = createTmpFile();
    // Open a writer on the temporary file
    var writer = openTextWriter(tmpFile.getPath());
    try{
        var line = null;
        // Current line
        /* read line by line and readLine will return null at the end of the file */
        while( (line = reader.readLine()) != null ){
            // Edit the line
            line = line.toUpperCase();
            // Write the result in the temporary file
            writer.write(line);
            // add a new line
            writer.newLine();
        }
    }
    finally{
        // Close the writer of the temporary file
        writer.close();
    }
}
finally{
    // Close the reader
    reader.close();
}
deleteFile(data.filename);
tmpFile.move(data.filename);
tmpFile.close();
```

deleteFile()

Function that is used to delete a file.

deleteFile(filename)

filename

String that specifies the path and file name of the file to be deleted.

Examples

Example: Deleting a file in a local folder:

```
deleteFile("c:\Content\test.txt");
```

Example: Deleting the sample data file used in the DataMapper:

```
deleteFile(data.filename);
```

execute()

Function that calls an external program and waits for it to end.

execute(command)

Calls an external program and waits for it to end.

command

String that specifies the path and file name of the program to execute.

How to get the exit value or output

It is possible to retrieve the exit value or the output of the `execute()` function by executing the `getExitValue()` function or `getInputStreamReader()` function, respectively.

Example

```
var ex = execute("ipconfig");
var exitValue = ex.getExitValue(); // Exit value is equal to 0 when OK
var reader = ex.getInputStreamReader(); // Reader for outputvar error
Reader = ex.getErrorStreamReader(); // Reader for error
// Read all output lines into text
var txt = "";
var line = reader.readLine();
while (line != null) {
    txt += line.trim();
    line = reader.readLine();
}
```

newByteArray()

Function that returns a new byte array.

newByteArray(size)

Returns a new byte array of of the specified number of elements.

size

Integer that represents the number of elements in the new array.

newCharArray()

Function that returns a new Char array.

newCharArray(size)

Returns a new Char array of the specified number of elements.

size

Integer that represents the number of elements in the new array.

newDoubleArray()

Function that returns a new double array.

newDoubleArray(size)

Returns a new Double array of the specified number of elements.

size

Integer that represents the number of elements in the new array.

newFloatArray()

Function that returns a new float array.

newFloatArray(size)

Returns a new Float array of the specified number of elements.

size

Integer that represents the number of elements in the new array.

newIntArray()

Function that returns a new array of Integers.

newIntArray(size)

Returns a new Integer array of the specified number of elements.

size

Integer that represents the number of elements in the new array.

newLongArray()

Function that returns a new long array.

newLongArray(size)

Returns a new Long array of the specified number of elements.

size

Integer that represents the number of elements in the new array.

newStringArray()

Function that returns a new string array.

newStringArray(size)

Returns a new String array of the specified number of elements.

size

Integer that represents the number of elements in the new array.

nextRecord()

A Step function that interrupts processing of the current record and instructs the DataMapper to move to the next record.

It behaves the same as the "Actions" on page 500 step **Skip to the next record**.

> **Note:** The *nextRecord()* and *stopProcessing()* functions are non-blocking calls. The script that follows these functions will be executed to completion. However, the next step in the DataMapper flow will not be executed.
>
> If this behavior is not desired, JavaScript can use an *if-else* statement to ignore execution of the unwanted part of the script, as shown in the next example:

```
if(record.Field1 === 'CA'){
 steps.stopProcessing();
} else {
 // script to run
}
```

> **Note:** The *nextRecord()* function skips the current record if called before a field is extracted in the current record.
> If the *nextRecord()* function is called after at least one field has been extracted, then the current record will be present in the dataset (output).

openBinaryReader()

Function that opens a file as a binary file for reading purposes. The function returns a DataInputStream (see DataInputStream).

openBinaryReader(filename)

filename

String that represents the name of the file to open.

openBinaryWriter()

Function that opens a file as a binary file for writing purposes. The function returns a DataOutputStream(see DataOutputStream).

openBinaryWriter(filename, append)

filename

String that represents the name of the file to open.

append

Boolean parameter that specifies whether the file pointer should initially be positioned at the end of the existing file (append mode) or at the beginning of the file (overwrite mode).

openTextReader()

Function that opens a file as a text file for reading purposes. The function returns a "TextReader" on the facing page object. Please note that the temporary file must be closed at the end.

openTextReader(filename,encoding)

filename

String that represents the name of the file to open.

encoding

String that specifies the encoding of the file to read (UTF-8, ISO-8859-1, etc.).

Example

In the following example, the `openTextReader()` function is used to open the actual data sample file in the DataMapper for reading.

```
var fileIn = openTextReader(data.filename);
var tmp = createTmpFile();
var fileOut = openTextWriter(tmp.getPath());
var line;

while((line = fileIn.readLine())!=null){
    fileOut.write(line.replace((subject),""));
    fileOut.newLine();
}

fileIn.close();
fileOut.close();
deleteFile(data.filename);
tmp.move(data.filename);
```

TextReader

The TextReader object, returned by the openTextReader() function, allows to open, parse, read and close a text file. (See: "openTextReader()" on the previous page.)

Methods

The following table describes the methods of the TextReader object.

Method	Description
close()	Closes the stream and releases resources.
open(inStream, inEncoding)	Creates a reader from an input stream. Parameters: • inStream: the input stream to read • inEncoding: the encoding to use when reading the file
open(inFileName, inEncoding)	Creates a reader on the specified file. Parameters: • inFilename: the path of the file to read • inEncoding: the encoding to use when reading the file
parseCharset(inEncoding)	Returns a character set (Charset). Character sets are used to convert a binary data file into readable text. Parameters: • inEncoding: the encoding to use (e.g. ISO-8859-1, UTF-8)
read()	Reads and returns the next character, or -1 if the end of the stream has been reached.
readLine()	Reads and returns the next line, or null if the end of the stream has been reached.
skip(offset)	Skips the specified number of characters. Parameters: • offset: the number of characters to skip

openTextWriter()

This function opens a file as a text file for writing purposes. The function returns a "TextWriter" on the next page object. This must be closed at the end.

openTextWriter(filename, encoding, append)

filename

String that represents the name of the file to open.

encoding

String specifying the encoding to use (UTF-8, ISO-8859-1, etc.)..

append

Boolean parameter that specifies whether the file pointer should initially be positioned at the end of the existing file (append mode) or at the beginning of the file (overwrite mode).

Example

In the following example, the `openTextWriter()` function is used to open the newly created temporary file for writing:

```
var fileIn = openTextReader(data.filename);
var tmp = createTmpFile();
var fileOut = openTextWriter(tmp.getPath());
var line;

while ((line = fileIn.readLine())!=null){
    fileOut.write(line.replace((subject),""));
    fileOut.newLine();
}

fileIn.close();
fileOut.close();
deleteFile(data.filename);
tmp.move(data.filename);
tmp.close();
```

TextWriter

The TextWriter object, returned by the openTextWriter() function, allows to open a text file, write to it and close it.

Methods

The following table describes the methods of the TextWriter object.

Method	Description
close()	Close the stream.
newLine()	Creates a new line in the file.
open(filename)	Creates a new writer on a file to write at the beginning of the file. Parameters: • filename: the path of the file to open

Method	Description
open(inFilename, inEncoding, append)	Creates a new writer on a file. Parameters: • inFileName: the path of the file • inEncoding: the encoding to be used when writing to the file • append: boolean; true if the data will be written at the end of the file
write(c)	Writes a character in the file
write(value)	Writes a string in the file. Parameters: • value: the string value to write

stopProcessing()

A Step function that interrupts the DataMapper to stop processing of the entire data stream.

It behaves the same as the "Actions" on page 500 step **Stop data mapping**.

Note: The *nextRecord()* and *stopProcessing()* functions are non-blocking calls. The script that follows these functions will be executed to completion. However, the next step in the DataMapper flow will not be executed.

If this behavior is not desired, JavaScript can use an *if-else* statement to ignore execution of the unwanted part of the script, as shown in the next example:

```
if(record.Field1 === 'CA'){
 steps.stopProcessing();
} else {
 // script to run
}
```

The Designer

The Designer is a WYSIWYG (what you see is what you get) editor that lets you create templates for various output channels: Print, Email and Web.

A template may contain designs for multiple output channels: a letter intended for print and an e-mail variant of the same message, for example. Content, like the body of the message or letter, can be shared across these contexts.

Templates are personalized using scripts and variable data. More advanced users may edit the underlying HTML, CSS and JavaScript directly.

The following topics will help to quickly familiarize yourself with the Designer.

- "Designer basics" below. These are the basic steps for creating and developing a template.
- "Features" on the facing page. These are some of the key features in the Designer.
- "Designer User Interface" on page 1128. This part gives an overview of all elements in the Designer User Interface, like menus, dialogs and panes.

More help can be found here:

- Learn: Visit the OL Resource Center to learn about OL Connect from blog posts and practical how-tos.
- Forum: Browse the forum and feel free to ask questions about the use of OL Connect software.

Designer basics

With the Designer you can create templates for personalized letters, emails and web pages, and generate output from them.

These are the basic steps for creating and developing a template:

1. **Create a template**
   Create a template, using one of the Template Wizards. See "Creating a template" on page 589.
2. **Fill the template**
   Add text, images and other elements to the template, and personalize and style them. See:
   - "Content elements" on page 761
   - "Personalizing content" on page 923

- "Styling and formatting" on page 879

For most templates, an **address block** will be your starting point. See "Address Block" on page 955.

You can create **data mapping configurations** to extract data from a variety of data sources. See "DataMapper basics" on page 336.

3. **Generate output**
   Adjust the settings, test the template and generate output through the Designer. Or, let an automation tool generate the output, using your data mapping configurations, templates and (in the case of a Print template) print presets.
   See "Generating output" on page 1637.

   You can automate your customer communications using Workflow and/or Node-RED (see "OL Connect Automate" on page 317).

Tip: Alternatively you could start with a **Sample Project** which creates an entire Connect solution: a Workflow configuration, as well as any Connect templates, data mapping configurations, Job Presets and Output Presets that are used in that configuration. See: "Sample Projects" on page 1185.

Features

The Designer is Connect's module to create templates for personalized customer communications. These are some of the key features in the Designer:

"Templates" on the next page. Start creating, using and sharing templates.

"Context" on page 610. A context contains one or more designs for one output channel.

- "Print" on page 616. This topic helps you design and fill sections in the Print context.
- "Email" on page 659. This topics helps you design an email template.
- "Web" on page 682. This topic helps you design a web page.

"Sections" on page 612. Sections in one context are designed for the same output channel.

"Content elements" on page 761. Elements make up the biggest part of the content of each design.

"Snippets" on page 867. Snippets help share content between contexts, or insert content conditionally.

"Styling and formatting" on page 879. Make your Designer templates look pretty and give them the same look and feel with style sheets.

"Personalizing content" on page 923. Personalize your customer communications using variable data.

"Writing your own scripts" on page 1064. Scripting can take personalization much further. Learn how to script via this topic.

"Generating output" on page 1637. Learn the ins and outs of generating output from each of the contexts.

Templates

The Designer is a WYSIWYG (what you see is what you get) tool to create templates. This topic gets you started. It explains how to create a template, what is found in a template file, and how output can be generated.

Creating a template

In the **Welcome** screen that appears after startup, get off to a flying start choosing **New**, at the left. Then select the output channel that will be prevalent in your template.

You can either select a Template Wizard or scroll down to **Online resources** to download a ready-made file with existing demo content as a starter for your file.

The Template Wizards can also be accessed from the menu: click **File**, click **New**, expand the **Template** folder, and then expand one of the **templates** folders.

There are Wizards for each output channel, or **contexts** as they are called in the Designer: Email, Print and Web.

See:

- "Creating an Email template with a Wizard" on page 664
- "Creating a Print template with a Wizard" on page 618
- "Creating a Web template with a Wizard" on page 683

Tip: The quickest way to create a Print template based on a PDF file is to **right-click the PDF** file in the Windows Explorer and select **Enhance with Connect**.

After creating a template you can add the other contexts (see "Context" on page 610), as well as extra sections (see "Sections" on page 612), to the template.
It is, however, not possible to use a Template Wizard when adding a context or section to an existing template.

Tip: When you add an Email context to an existing template you get a 'basic action email'. This is one of the 4 types of email that you can choose from when you start a template with an Email Template Wizard; see "Creating an Email template with a Wizard" on page 664.

Opening a template

To open a template from the Welcome screen, select **Open**.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

To open a template from the menu, select **File > Open**.
Then select the template file. A template file has the extension .OL-template.

The most recently opened templates, data mapping configurations and print presets are listed on the Welcome screen and in the menu: **File > Open Recent**.
To clear these lists, select **Window > Clear Recent Files Lists**.

Caution: A template created in an older version of the software can be opened in a newer version. However, opening and saving it in a newer version of the software will convert the template to the newest file format. The converted template can't be opened in older versions of the software.

Opening a package file

Templates can also be stored in a package file (see "Creating package files" on page 593). To open a package file, switch the file type to Package files (*.OL-package) in the Open File dialog. If the package contains Print Presets, you will be asked if you want to import them into the proper repositories.

Saving a template

A Designer template file has the extension **.OL-template**. It is a zip file that includes up to 3 contexts, all the related resources and scripts, and (optionally) a link to a data mapping configuration.

To save a template, select **File > Save** or press **Ctrl**+**S**. The first time you'll have to give the template a name. **File > Save as** allows you to save the template with a different name. The **Save as** dialog also appears when saving a template that is read-only.

Tip: To quickly copy the name of any other file, set **Save as type** to **Any file (*.*)** in the Save dialog. Select a file to put its name in the File name field. Then set **Save as type** to **Template files (*.OL-template)** and save the template.

When more than one resource is open and the Designer software is closed, the Save Resources dialog appears. This dialog displays a list of all open resources with their names and file location. Selected resources will be saved, deselected resources will have all their changes since they were last saved dismissed.

Saving older templates

Saving a template in a newer version of the software will convert the template to the newest file format. This makes it unreadable to older versions of the software.
The warning message that is displayed in this case can be disabled.
To re-enable this message (and all other warning dialogs), go to **Window > Preferences > General**, and click the **Reset All Warning Dialogs** button at the bottom.

Read-only templates

A lock icon displayed next to the file name indicates that the file is read-only. Saving a read-only file opens the Save as dialog, allowing you to save the file under a different name. The new file is then opened and can be edited in the Designer.

Associated data file or data mapping configuration

When you save a template, any data file or mapping configuration that is currently open will be associated with the template by saving the path to the data file or data mapping configuration in the template file.
The next time you open the template the Designer will attempt to reopen the data file or data mapping configuration as well.

To change which data file or data mapping configuration is linked to the template, open both the template and the data file or data mapping configuration that should be linked to it; then save the template.

Auto Save

After a template has been saved for the first time, Connect Designer can auto save the template with a regular interval. To configure Auto Save:

1. Select the menu option **Window > Preferences > Save**.
2. Under **Auto save**, check the option **Enable** to activate the Auto Save function.
3. Change how often it saves the template by typing a number of minutes.

Auto Backup

Connect Designer can automatically create a backup file when you **manually** save a template. To configure Auto Backup:

1. Select the menu option **Window > Preferences > Save**.
2. Under **Auto backup**, check the option **Enable** to activate the Auto Backup function.
3. Type the number of revisions to keep.
4. Select the directory in which the backups should be stored.

Backup files have the same name as the original template with two underscores and a progressive number (without leading zeros) at the end: **originalname__1.OL-template**, **originalname__2.OL-template**, etc.

> **Note:** The Auto Save function does **not** cause backup files to be created.

File properties

On the menu, select **File > Properties** to view and complement the file properties. See "File Properties dialog" on page 1150.

File properties can be used in a template in two ways:

- Via a Handlebars expression. For example: {{@properties.Company}} inserts the value of the Company name property in the template. See also: "Variable data in text: expressions" on page 999 and "Handlebars expressions" on page 1005.
- In scripts, via the `template` object; see the Designer API: "template" on page 1613. If you are not familiar with writing scripts, refer to "Writing your own scripts" on page 1064.

Saving a copy / down-saving a template

The Connect software is backwards compatible: templates that were made with an older version of Connect can always be opened with the newest version of the software.
But newer templates cannot be opened with an older version of the software.
Connect, however, allows you to save a template in the format of a previous version of the software, so that you may restore a template that was accidentally opened and saved in a newer version, or share a template with users of a previous version of Connect.
Note that it may not always be possible to down-save a template. If a template uses a feature that did not exist in a certain older version, it won't be possible to down-save it to that older version.

To **down-save** a template, select **File > Save a Copy**, from the menu. Select the software version and click OK. Next you can select a location and give the template a name, as usual.

To **save a copy**, follow the same procedure without selecting a previous version of the software.

Creating package files

A package file (*.OL-package) contains one or more templates, data mapping configurations and Print Presets.

A data mapping configuration file contains the information needed to extract data from a certain type of data file. For more information see "Data mapping configurations" on page 337.

Print Presets make it possible to do such things as filtering and sorting records, grouping documents and splitting the print jobs into smaller print jobs, as well as the more standard selection of printing options, such as binding, OMR markings and the like. See "Print Presets" on page 1644 for more details.

Package files can be opened by other Connect users. They can also be imported into Workflow and sent to the Connect Server.

To open the Package dialog, select **File > Package...**. For an explanation of the options in the Package dialog, see "Package dialog" on page 1171.

Generating output from the Designer

Output can be generated directly from the Designer, through Workflow, or by means of the Connect Server REST API; see "Generating Print output" on page 1639, "Generating Email output" on page 1673 and "Generating Web output" on page 1682.

To test a template first, select **Context > Preflight**. Preflights execute the template without actually producing output and it displays any issues once it's done (see also: "Testing scripts" on page 1074).

Sending files to Workflow

Workflow can generate output from a template in automated processes. For this, the template has to be sent to Workflow. Alternatively you may create a Package file (see "Creating package files" above) and import that into Workflow.

The Send to Workflow dialog sends templates, data mapping configurations and Print Presets to the Workflow server.
A data mapping configuration file contains the information necessary for data mapping: the settings to read the source file (Delimiter and Boundary settings), the data mapping workflow with its extraction instructions ('Steps'), the Data Model and any imported data

samples. For more information see "Data mapping configurations" on page 337.
Print Presets make it possible to do such things as filtering and sorting records, grouping documents and splitting the print jobs into smaller print jobs, as well as the more standard selection of printing options, such as binding, OMR markings and the like. See "Job Preset" on page 1359 and "Output Presets" on page 1375 for more details.

To open the dialog, select **File > Send to Workflow**. For an explanation of the options in it, see "Send to Workflow dialog" on page 1206.

Sending files to Connect Server or to another server

When OL Connect plugins in a Workflow process need templates and other configuration files, Workflow sends the necessary resources to the Connect Server. The Send files to Server dialog provides a way to send templates, data mapping configurations and print presets to the Connect Server directly, or to another server. This removes the need to upload them via the REST API in a Run Script plugin in Workflow, or via a tool like the Postman app, in case the OL Connect plugins are not used or when you are using a different automation tool like Node-RED.

To open the dialog, select **File > Send to Server**. For an explanation of the options in it, see Send to Server dialog.

Exporting a template report

A template report can be used for archiving purposes or to provide information about the template to people who do not have access to Connect. Such a report can be exported in PDF or XML format. By default it contains a summary of the template with an overview of all the settings and resources that are used in the template: media, master pages, contexts, sections, images, scripts, runtime parameters, etc. The file properties are included as well (see "File Properties dialog" on page 1150).

To open the Export Template Report wizard, select **File > Export Report**. For a description of all options, see "Export Template Report wizard" on page 1149.

Creating a custom template report

The Export Template Report wizard also offers the possibility to export custom template reports (in PDF format only). A custom template report could contain another selection of information and present that differently, e.g. with the logo of your company.

To create a custom template report, you need two files: A template design with the desired layout and variable data. This .OL-template file has to be made in the Designer. A data mapping configuration provides the variable data. You could use the data mapping configuration made for the standard template report, or create another one in the DataMapper module, using the standard XML template report as data sample. The

DataMapper is included only in OL Connect Professional and OL Connect Enterprise. Data mapping configurations have the extension .OL-datamapper. The following zip file contains both the template and data mapping configuration that are used to generate the standard template report: report-template.zip.

Creating an Email template with a Wizard

With the Designer you can design Email templates as well as PDF attachments. PDF attachments are designed in the Print context; see "Print context" on page 624.

It is strongly recommended to start creating an Email template with a Wizard, because designing HTML email that displays properly on a variety of devices and screen sizes is challenging. Email clients can, and will, interpret the same HTML and (inline) CSS in totally different ways (see "Designing an Email template" on page 660).

With an Email Template Wizard you can easily create an Email template that outputs emails that look good on virtually any email client, device and screen size.

After creating an Email template, the other contexts can be added to it, as well as other sections (see "Context" on page 610 and "Email templates" on page 669).

To create an Email template with a Template Wizard:

1. In the **Welcome** screen that appears after startup:

a. Choose **New** at the left, then **Email** at the right.

b. Select **Blank template**, **Basic message** (with a call-to-action button) or **Invoice** (with a detail table); or scroll down and select one of the Email templates from the **online resources**.

> **Tip:** Click the Home icon at the top right to reopen the Welcome screen.

Alternatively, on the menu select **File > New**, and expand the **Template** folder.
Now you can select **Email Template**; this starts the Basic Action Email wizard. Or scroll further down and expand the **Basic Email templates** folder, the **Banded Email templates** folder, or the **Slate: Responsive Email Templates by Litmus** folder, and select one of those templates.

See "Email Template Wizards" on page 597 for information about the various types of Template Wizards.

2. Make adjustments to the initial (or last used) settings. The options for each type of template are listed below.
Click **Next** to go to the next settings page if there is one.

3. Click **Finish** to create the template.

The Wizard creates:

- An Email context with one section in it. The section contains dummy text and one or more **expressions** for variable data, for example: "Hello {{first}}". You will want to replace those by the names of fields in your data. See "Variable data in the text" on page 924.

  The Invoice email template also contains a Dynamic Table; see "Dynamic Table" on page 974.
  Note that this table does not use one of the default table styles, and that the style sheet with the default table styles is not present in the template. To add that style sheet to the template, insert a table using the Dynamic Table wizard.

- A style sheet, named context_htmlemail_styles.css, and another style sheet depending on which Template Wizard was used. The style sheets can be found in the **Stylesheets** folder on the **Resources** pane.

The Wizard opens the Email section, so that you can fill it with text and other elements; see "Content elements" on page 761, "Email context" on page 667, "Email templates" on page 669 and "Email header settings" on page 672.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Note that the contents of the email are arranged in tables. The many tables in an Email template ensure that the email looks good on virtually any email client, device and screen size. As the tables have no borders, they are initially invisible.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

Tip: To get started quickly, you can import various resources (master pages, media, sections, images, style sheets, scripts, and the data model) from an existing template. See "Import Resources dialog" on page 1155.

There are Wizards for three kinds of Email templates: for **Basic Email**, for **Banded Email**, and **Slate** templates for responsive email designed by Litmus.

Slate: Responsive Email Templates by Litmus

Scroll past the Web Template Wizards to see the Slate: Responsive Email templates, created by Litmus (see https://litmus.com/resources/free-responsive-email-templates).

More than 50% of emails are opened on mobile. These five responsive HTML email templates are optimized for small screens and they look great in any inbox. They've been tested in Litmus and are completely bulletproof.

> **Tip:** After creating the email template, click the Responsive Design View icon at the top of the workspace to see how the email looks on different screen sizes.

The only thing you can set in advance for a Slate template is the color of the call-to-action button. Click the small colored square, right next to the field that holds the default color value, to open the Color dialog and pick a color (see "Color Picker" on page 1143).
The color can be changed later; see "Colors" on page 913.

Basic Email and Banded Email

The difference between Basic and Banded email is that the contents of a Basic email extend to the email's margin, rather than to the edge of the window in which it is read, as the contents of Banded emails do.
The Banded Email **Action** Template is a simple call-to-action email with a message, header and a button linking to a website, such as an informational or landing page.
The Banded Email **Invoice** Template is an invoice with an optional Welcome message and Pay Now button.

Settings

For a **Blank** email you can not specify any settings in the Wizard.

For an **Action** or **Invoice** email, the Email Template Wizard lets you choose:

- The subject. You can change and personalize the subject later, see "Email header settings" on page 672.
- The text for the header. The header is the colored part at the top. The text can be edited later.

- The color of the header and the color of the button. Click the small colored square, right next to the field that holds the default color value, to open the Color dialog and pick a color (see "Color Picker" on page 1143). The color can be changed later; see "Colors" on page 913.
- A logo. Use the Browse button to select an image from disk, or enter a URL to use a remote image.
- The web address where the recipient of the email will be taken after clicking the button in the email. Type the URL in the **Link** field.

In addition, for an **Invoice** email you can change the following content settings:

- **Show Welcome Message**. Check this option to insert a salutation and one paragraph with dummy text in the email.
- **Detail Table Name**. Type the name of a detail table to fill the lines of the invoice with data. In the Designer, a detail table is a field in the Data Model that contains a variable number of items (usually transactional data).

Creating a Web template with a Wizard

With the Designer you can design Web templates and output them through Workflow or as an attachment to an email when generating Email output.

Capture On The Go templates are a special kind of Web templates; see "Capture OnTheGo template wizards" on page 717.

A Web Template Wizard helps you create a Web page that looks good on virtually any browser, device and screen size.

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop computers, tablets, and mobile phones. Foundation is tested across many browsers and devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

For more information about the use of Foundation in the Designer, see "Using Foundation" on page 721.

After creating a Web template, the other contexts can be added, as well as other sections (see "Adding a context" on page 611 and "Adding a Web page" on page 689).

To create a Web template with a Template Wizard:

1. In the **Welcome** screen that appears after startup:

a. Choose **New** at the left, then **Web** at the right.

b. Select **Blank template** or scroll down and select one of the Web templates from the **online resources**.

> **Tip:** Click the Home icon at the top right to reopen the Welcome screen.

Alternatively, on the menu select **File > New**, and expand the **Template** folder. Now you can select **Web Page Template**.
Or scroll further down, expand the **Foundation Web Page Starter** folder and select one of those templates.

There are 4 types of Foundation Web Template Wizards:

- Blank
- Contact Us
- Jumbotron
- Thank You

If you don't know what template to choose, see "Web Template Wizards" on page 601 further down in this topic, where the characteristics of each kind of template are described.

2. Click **Next** and make adjustments to the settings. The wizard remembers the settings that were last used for a Foundation Web template.

- **Section**:
  - **Name**: Enter the name of the Section in the Web context. This has no effect on output.
  - **Description**: Enter the description of the page. This is the contents of a <meta name="description"> HTML tag.
- **Top bar** group:
  - **Set width to Grid**: Check this option to limit the width of the top bar contents to the Foundation Grid, instead of using the full width of the page.
  - **Stick to the top of the browser window**: Check to lock the top menu bar to the top of the page, even if the page has scroll bars. This means the menu

bar will always be visible in the browser.

- **Background color**: Enter a valid hexadecimal color code for the page background color (see w3school's color picker), or click the colored circle to the right to open the Color Picker.

- **Colors** group: Enter a valid hexadecimal color code (see w3school's color picker) or click the colored square to open the Color Picker dialog (see "Color Picker" on page 1143), and pick a color for the following elements:
  - **Primary**: links on the page.
  - **Secondary**: secondary links on the page.
  - **Text**: text on the page contained in paragraphs (<p>).
  - **Headings**: all headings (<h1> through <h6>) including the heading section's subhead.

3. Click **Finish** to create the template.

The Wizard creates:

- A Web context with one web page template (also called a **section**) in it. The web page contains a Header, a Section and a Footer element with dummy text, and depending on the type of web page, a navigation bar, button and/or Form elements.
- Resources related to the Foundation framework (see "Web Template Wizards" on the next page): style sheets and JavaScript files. The style sheets can be found in the **Stylesheets** folder on the **Resources** pane. The JavaScript files are located in the **JavaScript** folder on the **Resources** pane, in a **Foundation** folder.
- A collection of Snippets in the **Snippets** folder on the Resources pane. The Snippets contain ready-to-use parts to build the web page. Double-click to open them. See "Snippets" on page 867 for information about using Snippets.
- Images: icons, one picture and one thumbnail picture. Hover your mouse over the names of the images in the **Images** folder on the **Resources** pane to get a preview.

The Wizard opens the Web section, so that you can fill it with text and other elements; see "Content elements" on page 761, "Web Context" on page 687 and "Web pages" on page 688.

Web pages can be personalized just like any other type of template; see "Personalizing content" on page 923.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

Tip: To get started quickly, you can import various resources (master pages, media, sections, images, style sheets, scripts, and the data model) from an existing template. See "Import Resources dialog" on page 1155.

Web Template Wizards

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop computers, tablets, and mobile phones. Foundation is tested across many browsers and devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

Jumbotron

The name of the Jumbotron template is derived from the large screens in sports stadiums. It is most useful for informative or marketing-based websites. Its large banner at the top can display important text and its "call to action" button invites a visitor to click on to more information or an order form.

Contact Us

The Contact Us template is a contact form that can be used on a website to receive user feedback or requests. It's great to use in conjunction with the Thank You template, which can recap the form information and thank the user for feedback.

Thank You

The Thank You template displays a thank you message with some text and media links.

Blank web page

The Blank Web Page template is a very simple Foundation template that contains a top bar menu and some basic contents to get you started.

Capture OnTheGo template wizards

With the Designer you can create Capture OnTheGo (COTG) templates. COTG templates are used to generate forms for the Capture OnTheGo mobile application. For more information about this application, see the website: Capture OnTheGo.

A Capture OnTheGo Form is actually just a Web Form, that you could add without a wizard, but the COTG Template Wizards include the appropriate JavaScript files for the Capture OnTheGo app, and styles to create user-friendly, responsive forms. They are built upon the Foundation framework.

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop computers, tablets, and mobile phones. Foundation is tested across many browsers and devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

For more information about the use of Foundation in the Designer, see "Using Foundation" on page 721.

After creating a COTG template, the other contexts can be added, as well as other sections (see "Adding a context" on page 611 and "Adding a Web page" on page 689).

> **Tip:** If the COTG Form replaces a paper form, it can be tempting to stick to the original layout. Although that may increase the recognizability, it is better to give priority to the user-friendliness of the form. Keep in mind that the COTG form will be used on a device and don't miss the chance to make it as user-friendly as possible. See "Designing a COTG Template" on page 714.

Creating a COTG template using a wizard

To create a COTG template with a template wizard:

1.
   - In the **Welcome** screen that appears after startup and when you click the Home icon at the top right, choose **Browse Template Wizards**. Scroll down until you see the **Capture OnTheGo Starter** Template Wizards.
   - Alternatively, on the **File** menu, click **New**, expand the **Template** folder, and then expand the **Capture OnTheGo Starter** folder.
2. Select a template. There are 8 types of Web Template Wizards:
   - **Blank**. The Blank COTG Template has some basic design and the appropriate form, but no actual form or COTG elements.
   - **Bill of Lading**. The Bill of Lading Template is a transactional template that includes a Dynamic Table with a checkmark on each line, along with Signature and Date COTG elements. Use this wizard as a way to quickly start any new Zurb Foundation based form for Capture OnTheGo.
   - **Event Registration**. The Event Registration Template is a generic registration form asking for name, phone, email, etc.
   - **Event Feedback**. The Event Feedback Template is a questionnaire containing different questions used to rate an experience.
   - **Membership Application**. The Membership Application Template is a signed generic request form that can be used for memberships such as gyms, clubs, etc.
   - **Patient Intake**. The Patient Intake Template is a generic medical questionnaire that could potentially be used as a base for insurance or clinic form.
   - **Kitchen Sink**. The Kitchen Sink Template includes a wide range of basic form and COTG form elements demonstrating various possibilities of the software.
   - **Time Sheet**. The Time Sheet Template is a single page application used to add time entries to a list. This template demonstrates the dynamic addition of lines within a COTG template, as the Add button creates a new time entry. There is no limit to the number of entries in a single page. Submitted data are grouped using arrays (see "Grouping data using arrays" on page 701).
3. Click **Next** and make adjustments to the settings. The wizard remembers the settings that were last used for a COTG template.
   - **Create Off-Canvas navigation menu**: an Off-Canvas menu is a Foundation component that lets you navigate between level 4 headings (<h4>) in the form. Check this option to add the menu automatically.

- **Submit URL**: enter the URL where the form data should be sent. The URL should be a server-side script that can accept COTG Form data.
- The **Title** and the **Logo** that you choose will be displayed at the top of the Form.
- **Colors**: Click the colored square to open the Color Picker dialog (see "Color Picker" on page 1143) and pick a color, or enter a hexadecimal color code (see w3school's color picker) for the page background color.
  Do the same for the background color of the navigation bar at the top and for the buttons on the Form.

4. Click **Next** to go to the next settings page if there is one.
5. Click **Finish** to create the template.

The Wizard creates:

- A **Web context** with one web page template (also called a section) in it. The web page contains an 'off-canvas' Div element, Header, a Section and a Footer element with dummy text, and depending on the type of web page, a navigation bar, button and/or Form elements.
- **Style sheets** and **JavaScript files** related to the COTG form itself and others related to the Foundation framework (see above). The style sheets can be found in the Stylesheets folder on the Resources pane. The JavaScript files are located in the JavaScript folder on the Resources pane.
- A collection of **snippets** in the Snippets folder on the Resources pane. The snippets contain ready-to-use parts to build the web form. Double-click to open them. See "Snippets" on page 867 and "Loading a snippet via a script" on page 1091 for information about using Snippets.

The Wizard opens the Web section, so that you can fill the Capture OnTheGo form.

6. Make sure to set the action and method of the form: select the form and then enter the action and method on the Attributes pane.

The **action** of a Capture OnTheGo form should specify the Workflow HTTP Server Input task that receives and handles the submitted data. The action will look like this: **http://127.0.0.1:8080/action** (8080 is Workflow's default port number; 'action' should be replaced by the HTTP action of that particular HTTP Server Input task).

The **method** of a Capture OnTheGo form should be **POST** to ensure that it doesn't hit a data limit when submitting the form. The GET method adds the data to the URL, and the length of a URL is limited to 2048 characters. Especially forms containing one or more Camera inputs may produce a voluminous data stream that doesn't fit in the URL. GET also leaves

data trails in log files, which raises privacy concerns. Therefore POST is the preferred method to use.

Filling a COTG template

Before inserting elements in a COTG Form, have the design ready; see "Designing a COTG Template" on page 714.

In a Capture OnTheGo form, you can use special Capture OnTheGo Form elements, such as a Signature and a Barcode Scanner element. For a description of all COTG elements, see: "COTG Elements" on page 835. To learn how to use them, see "Using COTG Elements" on page 729.

> **Tip:** If you have started creating your Capture OnTheGo template using a COTG Template Wizard, you can find ready-made elements in the Snippets folder on the Resources pane.

Foundation, the framework added by the COTG template wizards, comes with a series of features that can be very useful in COTG forms; see "Using Foundation" on page 721.

Naturally, Web Form elements can also be used on COTG Forms (see "Forms" on page 842 and "Form Elements" on page 847) as well as text, images and other elements (see "Content elements" on page 761).

Capture OnTheGo templates can be personalized just like any other type of template; see "Personalizing content" on page 923.

> **Tip:** Use the **Outline** pane at the left to see which elements are present in the template and to select an element.
>
> Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.
>
> Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

> **Tip:** Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

> **Tip:** If you have started creating your Capture OnTheGo template using a COTG Template Wizard, you can find ready-made elements in the Snippets folder on the Resources pane.

Resources

This page clarifies the difference between Internal, External and Web resources that may be used in a template, and explains how to refer to them in HTML and in scripts.

All resources that a template uses are displayed in the "Resources pane" on page 1250.

Tip: The Text Filter box at the top of the Resources pane allows to look for text in the name of resources and in the source of any text-based files: HTML (including sections, master pages, and snippets), JSON, JS and CSS.

Tip: To get started quickly, you can import various resources (master pages, media, sections, images, style sheets, scripts, and the data model) from an existing template. See "Import Resources dialog" on page 1155.

Internal resources

Internal resources are files that are added to and saved with the template.

To add images, fonts, style sheets, and snippets to your template, you can **drag** or **copy/paste** them into the Resources Pane. See also: "Images" on page 852, "Snippets" on page 867, "Styling templates with CSS files" on page 881 and "Fonts" on page 917.
Resource files can also be dragged or copy/pasted **out** of the the application to save them on a local hard drive.

Alternatively you could **import** resources and scripts from another template; click **File > Import Resources...** in the menu and select a template to import resources from (see).

Once added or imported, internal resources are accessed using a relative path, depending where they're called from. Resources can be located in the following folders:

- images/ contains the files in the Images folder.
- fonts/ contains the files in the Fonts folder.
- css/ contains the files in the Stylesheets folder.
- js/ contains the files in the JavaScripts folder.
- snippets/ contains the files in the Snippets folder.

When refering to them, normally you would simply use the path directly with the file name. The structure within those folders is maintained, so if you create a "signatures" folder within the "Images" folder, you need to use that structure, for example in HTML: <img src="images/signatures/johnsmith.gif">. In scripts, you can refer to them in the same way, for example:

```
results.loadhtml("snippets/en/navbar.html");
```

See also: "Loading a snippet via a script" on page 1091 and "Writing your own scripts" on page 1064.

Note: When referring to images or fonts from a CSS file, you need to remember that the current path is css/, meaning you can't just call images/image.jpg. Use a relative path, for example: #header { background-image: url('../images/image.jpg'); }

External resources

External resources are not stored in the template, but on the local hard drive or on a network drive. They are accessed using a path. The path must have forward slashes, for example

```
<img src="file:///c:/resources/images/signatures/johnsmith.gif">
```

or

```
var json_variables = loadjson("file:///d:/jsondata/variables.json");.
```

The complete syntax is: file://<host>/<path>. If the host is "localhost", it can be omitted, as it is in the example, resulting in **file:///<path>**. The empty string is interpreted as `the machine from which the URL is being interpreted'.

Network paths are similar:

```
results.loadhtml("file://///servername/sharename/folder/snippet.html");
```

If the file is located on another server in your network, the path must contain five slashes after "file:".

Note: Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file://///myserver/myfolder/file.txt).

Some limitations

- Style sheets cannot refer to external resources.
- The OL Connect Server user needs access to whichever network path is used. If the

network path is on a domain, the OL Connect Server must be identified with domain credentials that have access to the domain resources.

For more information on network paths, please see this Wikipedia entry: file URI scheme.

Web resources

Web resources are simply accessed using a full URL. This URL needs to be publicly accessible: if you type in that URL in a browser on the server, it needs to be visible. Authentication is possible only through URL Parameters:

```
(http://www.example.com/data.json?user=username&password=password)
```

or through HTTP Basic Auth:

```
(http://username:password@www.example.com/data.json).
```

Resources can also be called from a OL Connect Workflow instance:

- "Static Resources", as set in the preferences, are accessed using the resource path, by default something like http://servername:8080/_iRes/images/image.jpg. (For guidance on setting the preferences, search for 'HTTP Server Input 2' in the OL Connect Workflow help files on: OL Connect Workflow Help).
- Resources can also be served by processes: http://servername:8080/my_process?filename=image.jpg (assuming "my_process" is the action in the HTTP Server Input).

Runtime parameters

Runtime parameters can pass values from an automation tool - OL Connect Workflow, usually - to a template, data mapping configuration or Job Preset. The actual values of the parameters may be different from one time that a process runs to the next, which means the output could be different even when the same template, data mapping configuration and/or Job Presets are used.

Defining runtime parameters for a template

Runtime parameters in a template must be defined on the **Parameters** pane. This can be found next to the Data Model pane. If it isn't visible, use the menu: **Window > Show View > Parameters** to make it visible.

> **Note:** In OL Connect Workflow, the variable that contains the value of a runtime parameter must be selected in the Runtime Parameters section of the OL Connect task that uses the template: the All In One, Create Print Content, Create Email

Content, Create Web Content, or Create Preview PDF task. For more information see "OL Connect tasks" on page 303.

To add a parameter, start by opening the template to which the parameter should be added. Make sure the template is visible in the workspace; then open the Parameters pane.

Note: Runtime parameters are always added to the file currently visible in the workspace.

To add a **single** parameter:

1. Click the **Add** button.
2. Give the parameter a **name**.
3. Select the data **type**. This impacts the way the data can be handled in a script; for example, if a parameter's type is Number, its value can be used directly in calculations, without having to parse it first.
   In the Parameters pane, the type of a runtime parameter can be recognized by its icon.
4. Optionally, set a ***default* value**. A default value will only be used in the case that there is no actual value coming from the automation tool, e.g. one of the Content Creation tasks in OL Connect Workflow, at **runtime**. Since runtime parameters cannot be empty, this prevents an error.
   At **design-time** runtime parameters are undefined, unless you add values manually in order to test a template. For instructions, see below.
5. Click **OK**.

To add **multiple** parameters or to **set values**:

1. Click the **Set Values** button. This opens the Set Values dialog which allows to load JSON data.
   Any previously defined runtime parameters are displayed in the dialog as a JSON object. Parameter names appear as keys; default values appear as values.
2. Either browse to the location of a JSON file and select it, or paste/write/edit the JSON directly in the box below the File field.
   Note that JSON arrays are not supported.
3. Click **OK**. The keys of the JSON object appear in the Name column, the values in the Value column.

4. For new parameters an attempt is made to derive the data type from the value, but this is only possible to a limited extent. Double-click any new parameter to set its desired data type.

Editing a runtime parameter

To edit a runtime parameter, either:

- Double-click the parameter; or right-click and choose **Edit**; or select the parameter and click the **Edit** button, and modify it in the Edit Parameter dialog.
- Click the **Set Values** button and edit the JSON. Note that this method cannot be used to change the name of a runtime parameter. You can only change the values and add new runtime parameters this way.

If the Data Model contains a JSON Record Data List (see "A JSON Record Data List" on page 1160), you could edit the runtime parameters by adding or editing the `parameters` object using the "JSON sample data dialog" on page 1158.

Accessing runtime parameters

Runtime parameters can be used in a template in two ways:

- Via a Handlebars expression. For example: {{@parameters.jobId}} inserts the value of the jobID parameter in the template. See also: "Variable data in text: expressions" on page 999 and "Handlebars expressions" on page 1005.
- In scripts, via `merge.template.parameters`. (See "Standard Script API" on page 1481 and "template" on page 1613.)
  A script can read and could also change the values in the `parameters` array. Note however that any runtime parameter value will be reset with each new record that the template is being merged with.

  If you are not familiar with writing scripts, refer to "Writing your own scripts" on page 1064.

Context

Contexts are parts of a template that are each used to generate a specific type of output: Email, Print, or Web.

- The Print context outputs documents to either a physical printer or a PDF file; see "Print context" on page 624.

- The Email context outputs HTML email, composed of HTML code with embedded CSS. See "Email context" on page 667.
- The Web context outputs an HTML web page. See "Web Context" on page 687.

When a new template is made, the Context appropriate to that new template is automatically created, including one section. After a template has been created, the other two contexts can be added to it; see "Adding a context" below.

Tip: When you add an Email context to an existing template you get a 'basic action email'. This is one of the 4 types of email that you can choose from when you start a template with an Email Template Wizard; see "Creating an Email template with a Wizard" on page 664.

Outputting and combining contexts

All contexts can be present in any template and they can all be used to output documents; see "Generating Email output" on page 1673, "Generating Print output" on page 1639 and "Generating Web output" on page 1682.

They can even be combined in output.

If present in the same template, a Print context and a Web context can be attached to an Email context.

You could select Print sections based on a value in the data with a Conditional Print Sections script; see "Conditional Print sections" on page 962.

Outputting other combinations of contexts, and selecting other sections based on a value in the data, can be done via a Control Script; see "Control Scripts" on page 1098.

Adding a context

To add a context, right-click the **Contexts** folder on the **Resources** pane and click **New print context**, **New email context** or **New web context**. Or use **Context > Add** in the main menu. Only one context of each type can be present in a template. Each context, however, can hold more than one section; see "Sections" on the facing page.

Importing a context

To import a context, click **File > Import Resources...** in the menu. See: "Import Resources dialog" on page 1155.

Deleting a context

To delete a context, right-click the context on the **Resources** pane and click **Delete**.

Caution: If you don't have a backup of the template, the only way to recover a deleted section, is to click **Undo** on the **Edit** menu, until the deleted section is restored. After closing and reopening the template it is no longer possible to restore the deleted context this way.
In the Saving Preferences you can set whether a backup file should be created when you save the template; see "Save preferences" on page 1059.

Sections

Sections are parts of one of the contexts in a template: Print, Email or Web. They contain the main text flow for the contents. In each of the contexts there can be multiple sections. A Print context, for example, may consist of two sections: a covering letter and a policy.

Adding a section

To add a section to a context, right-click the context (Email, Print or Web) on the **Resources** pane, and then click **New section**.
The new section has the same settings as the currently active section in the same context, or the first section in the same context if another context is active.

It is not possible to use a Template Wizard when adding a section to an existing template.

Tip: When you add an Email context to an existing template you get a 'basic action email'. This is one of the 4 types of email that you can choose from when you start a template with an Email Template Wizard; see "Creating an Email template with a Wizard" on page 664.

Tip: Print sections can be *cloned* (copied) on the fly, so that a document can have a dynamic number of sections, based on a detail table in the data. For more information see "Cloning sections" on page 965.

Importing a section

To import a section from another template, click **File > Import Resources...** in the menu. See: "Import Resources dialog" on page 1155.
Remember to copy the related source files, such as images, to the other template as well.

Editing a section

To open a section, expand the **Contexts** folder on the **Resources** pane, expand the respective context (**Print**, **Email** or **Web**) and double-click a section to open it.

Each section can contain text, images and many other elements (see "Content elements" on page 761), including variable data and other dynamic elements (see "Personalizing content" on page 923).

To preview a section, open the Preview tab in the Workspace (see "Workspace" on page 1261).

Copying a section

To copy a section:

1. Open the context on the **Resources** pane.
2. Right-click the section and select **Duplicate**.

   Alternatively you could first select **Copy**, then **Paste**. Note that sections cannot be pasted to another context.

The copy will have the same settings as the original.

Alternatively you may copy the contents of a section manually:

1. Open the section that you want to copy and go to the **Source** tab in the workspace.
2. Copy the contents of the **Source** tab (press **Ctrl+A** to select everything and then **Ctrl+C** to copy the selection).
3. Add a new section (see "Adding a section" on the previous page, above).
4. Go to the **Source** tab and paste the contents of the other section here (press **Ctrl+V**).
5. When copying a section to another template, add the related source files, such as images, to the other template as well.

Tip: The easiest way to copy a section to another template, is to use the **Import Resources** dialog in the other template. See: "Import Resources dialog" on page 1155.

Cloning a Print section

If documents in the output need to have a **variable number of sections**, based on data, Print sections can be cloned (copied) on the fly. See "Cloning a section using the wizard" on page 966.

Deleting a section

To delete a section:

- On the **Resources** pane, expand the **Contexts** folder, expand the folder of the respective context, right-click the name of the section, and then click **Delete**.

Caution: No backup files are maintained in the template. The only way to recover a deleted section, is to click **Undo** on the **Edit** menu, until the deleted section is restored, or by reverting to the last saved state (click File > Revert, on the menu). After closing and reopening the template it is no longer possible to restore the deleted context this way.

Renaming a section

To rename a section:

- On the **Resources** pane, expand the **Contexts** folder, expand the folder of the respective context, right-click the name of the section, and then click **Rename**.

  Note: Naming this element with a purely numeric name will cause issues with scripting. We recommend using a Alphanumeric name.

Section properties

Which properties apply to a section, depends on the context it is part of. See also: "Print sections" on page 629, "Email templates" on page 669, and "Web pages" on page 688.

To change the properties for a section:

1. On the **Resources** pane, expand the **Contexts** folder.
2. Expand the folder of the respective context.
3. Right-click the name of the section, and then click one of the options.

Applying a style sheet to a section

In order for a style sheet to be applied to a specific section, it needs to be included in that section. There are two ways to do this.

Drag & drop a style sheet

1. Click and hold the mouse button on the style sheet on the **Resources** pane.
2. Move the mouse cursor within the **Resources** pane to the section to which the style sheet should be applied.
3. Release the mouse button.

Using the Includes dialog

1. On the Resources pane, right-click the section, then click **Includes**.
2. From the **File types** dropdown, select **Stylesheets**.

3. Choose which CSS files should be applied to this section. The available files are listed at the left. Use the arrow buttons to move the files that should be included to the list at the right.
4. You can also change the order in which the CSS files are read: click one of the included CSS files and use the **Up** and **Down** buttons. Note that moving a style sheet up in the list gives it **less** weight. In case of conflicting rules, style sheets read later will override previous ones.

Note: Style sheets are applied in the order in which they are included in a section. The styles in each following style sheet add up to the styles found in previously read style sheets. When style sheets have a conflicting rule for the same element, class or ID, the **last** style sheet 'wins' and overrides the rule found in the previous style sheet.

Note: Style sheets that are linked to (i.e. included in) a section show a chain icon in the Resources pane (see "Resources pane" on page 1250).

Arranging sections

Changing the order of the sections in a context can have an effect on how they are outputted; see: "Print sections" on page 629, "Email templates" on page 669 and "Web pages" on page 688.

To rearrange sections in a context:

- On the **Resources** pane, expand the Contexts folder, expand the folder of the respective context, and then drag and drop sections to change the order they are in.
- Alternatively, right-click a section and click **Arrange**. In the Arrange Sections dialog you can change the order of the sections in the same context by clicking the name of a section and moving it using the **Up** and **Down** buttons.

Outputting sections

Which sections are added to the output, depends on the type of context they are in.

When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record. The sections are added to the output in the order in which they appear on the **Resources** pane. See "Generating Print output" on page 1639.

In Email and Web output, only one section can be outputted at a time. The section that will be output is the section that has been set as the 'default'. See "Generating Web output" on page 1682 and "Web pages" on page 688 and "Generating Email output" on page 1673 and "Email templates" on page 669. The 'default' section is always executed when the template

is run using the Create Email Content task in Workflow (see Workflow Help: Create Email Content).

It is, however, possible to include or exclude Print sections when the output is generated, depending on a value in the data. A Control Script can do this; see "Control Scripts" on page 1098.

See "Generating output" on page 1637 to learn how to generate Print documents, Web pages, or Email.

Print

Connect supports a number of different types of print outputs. These include:

- AFP (Enterprise edition only)
- IPDS (Enterprise edition only) It produces a spool file, with printer control available through the "Print Manager" on page 1689)
- PCL (Printer Control Language)
- PDF (Portable Document Format)
- PostScript (including the VIPP and VPS variants)
- PPML (Personalized Print Markup Language)

With the Designer you can create one or more Print templates and merge the template with a data set to generate personal letters, invoices, policies, or any other type of letter you can think of.

The Print **context** is the folder in the Designer that can contain one or more Print sections.

Print *sections*) are part of the Print context. They are meant to be printed directly to a printer or a printer stream/spool file, or to a PDF file (see "Generating Print output" on page 1639).
The Print context can also be added to Email output as a PDF attachment; see "Generating Email output" on page 1673.
When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record.

When a Print template is created or when a Print context is added to an existing template the Print context folder is created along with other folders and files that are specific to a Print context (see "Creating a Print template with a Wizard" on page 618, "Adding a context" on page 611 and "Print context" on page 624).

Only one Print section is created at the start, but you can add as many Print sections as you need; see "Print sections" on page 629.

Pages

Unlike emails and web pages, Print sections can contain multiple *pages*. Pages are naturally limited by their size and margins. If the content of a section does not fit on one page, the overflow goes to the next page. This happens automatically, based on the section's page size and margins; see "Page settings: size, margins and bleed" on page 642. The minimum number of pages can be set via the Print section properties; see "Print section properties" on page 1200.

Although generally the same content elements can be used in all three contexts (see "Content elements" on page 761), the specific characteristics of pages make it possible to use special elements, such as page numbers; see "Page numbers " on page 644.

See "Pages" on page 641 for an overview of settings and elements that are specific for pages.

Headers, footers, tear-offs and repeated elements (Master page)

In Print sections, there are often elements that need to be repeated across pages, like headers, footers and logos. In addition, some elements should appear on each first page, or only on pages in between the first and the last page, or only on the last page. Examples are a different header on the first page, and a tear-off slip that should show up on the last page.

This is what Master Pages are used for. Master Pages can only be used in the Print context.

See "Master Pages" on page 648 for an explanation of how to fill them and how to apply them to different pages.

Stationery (Media)

When the output of a Print context is meant to be printed on paper that already has graphical and text elements on it (called stationery, or preprinted sheets), you can add a copy of this media, in the form of a PDF file, to the Media folder.

Media can be applied to pages in a Print section, to make them appear as a background to those pages. This ensures that elements added to the Print context will correspond to their correct location on the preprinted media.

Note: When both Media and a Master Page are used on a certain page, they will both be displayed on the Preview tab of the workspace, the Master Page being 'in front' of the Media and the Print section on top. To open the Preview tab, click it at the bottom of the Workspace or select **View > Preview View** on the menu.

The Media will not be printed, unless this is specifically requested through the printer settings in the Print Wizard; see "Generating Print output" on page 1639.

See "Media" on page 652 for further explanation about how to add Media and how to apply them to different pages.

Copy Fit

Copy Fit is a feature to automatically adjust the font size of text to make it fit the available space. It could be used for the name of a person on a greeting card, for instance, or for the name of a product on a shelf talker. This feature is only available with Box and Div elements in Print sections.

For more information about this feature see "Copy Fit" on page 896.

Creating a Print template with a Wizard

A Print template may consist of various parts, such as a covering letter and a policy. Start with one of the Template Wizards for the first part; other parts (called 'sections') can be added later.

Print template wizards can be found in the Welcome screen and on the File menu.

In the **Welcome** screen that appears after startup:

1. Choose **New** at the left, then **Print** at the right.
2. Select **Blank template**, **PDF-based**, or Microsoft **Word-based**; or scroll down and select one of the Print templates from the **online resources**.

> **Tip:** Click the Home icon at the top right to reopen the Welcome screen.

Alternatively, on the menu select **File > New**, and expand the **Template** folder.
Now you can select **PDF-based Print** or Microsoft **Word-based Print**.
Or expand the **Basic Print templates** or **ERP templates** folder, and select a template type.

> **Tip:** The quickest way to create a Print template based on a PDF file is to **right-click the PDF** file in the Windows Explorer and select **Enhance with Connect**.

The various template types and their options are described below.

See "Print context" on page 624 and "Print sections" on page 629 for more information about Print templates.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Tip: To get started quickly, you can import various resources (master pages, media, sections, images, style sheets, scripts, and the data model) from an existing template. See "Import Resources dialog" on page 1155.

PDF-based Print template

Tip: The quickest way to create a Print template based on a PDF file is to **right-click the PDF** file in the Windows Explorer and select **Enhance with Connect**.

The PDF-based Print template wizard creates a document from an existing PDF file: a brochure, voucher, letter, etc. The PDF is used as the background image of the Print section (see "Using a PDF file or other image as background" on page 634). Variable and personalized elements, like a reseller address, voucher codes and so on, can be added in front of it (see "Personalizing content" on page 923).

By default, the PDF itself is added to the **Image** folder located in the **Resources** pane. Uncheck the option **Save with template** if the PDF should not be imported in the template. If it isn't saved with the template, the image remains external. Note that external images need to be available when the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced. External images are updated (retrieved) at the time the output is generated.

After clicking **Next**, you can change the settings for the page. The initial page size and bleed area are taken from the selected PDF.

When you click **Finish**, the Wizard creates:

- A Print context with one section in it; see "Print context" on page 624 and "Print sections" on page 629. The selected PDF is used as the background of the Print section; see "Using a PDF file or other image as background" on page 634. For each page in the PDF one page is created in the Print section.

- One empty Master Page. Master Pages are used for headers, footers, images and other elements that have to appear on more than one page, and for special elements like tear-offs. See "Master Pages" on page 648.
- One empty Media. Media, also called Virtual Stationery, can be applied to all pages in the Print section. See "Media" on page 652.

Tip: For an example, see this howto: Creating a print template from a PDF.

Word-based Print template

The Word-based Print template wizard creates a document from an existing **Microsoft Word** (**.docx**) file: a brochure, voucher, letter, mail merge document, etc. The images, lists, tables, and other content within the document are imported into Designer and used for a new template.

After clicking **Next**, you can change the settings for the page. The initial page size, margins, and orientation are taken from the original document settings, and you can adjust them on the dialog. You can also select the number of sections and a minimum number of pages. After the template is created, you can change the size of images and text boxes, and adjust the page size and margins.

Note: When importing from Word we substitute the font “Aptos” (a proprietary cloud font that cannot be used by a browser) with the more general “Calibri”.

When you click **Finish**, the Wizard creates:

- A Print context with the specified number of sections in it; see "Print context" on page 624 and "Print sections" on page 629. For each page in the document one page is created in the Print section.

The following will be added to the template:

- Image files for each image in the original file. As with other templates, you can place images on the master page if you want them to appear in every document in the same place.
- A folder to contain the imported image files.
- A separate .css file in the Stylesheets for every stylesheet that is imported with the MS Word file. The css files are linked to the active section. They can be edited (see "Styling your templates with CSS files" on page 885).

Microsoft Word documents with mail merge fields

If the Microsoft Word document contains mail merge fields, these mail merge fields (or markers) are added to the Data Model of the OL Connect template.

Select **File > Data > Open Data File...** to import the corresponding data.

Or create a data mapping configuration to fill the Data Model with actual data.

In the **template**, the mail merge fields are replaced with *expressions*. (See: "Handlebars expressions" on page 1005.)

- The brackets from the mail merge fields are converted to double curly braces.

Limitations

The conversion of a Word file to a template will most likely not be perfect. Here are some known limitations:

- If an image is cropped it will appear stretched in OL Connect.
- The position of an object is incorrect if it is positioned relative to something else (e.g. 100 points below a paragraph).
- Text wrapping is not supported in OL Connect. What you see in OL Connect generally corresponds to the layout option "In Line with Text" in MS Word.
- Lines and borders around objects are not supported.
- Headers and footers are not supported.

Basic Print template wizards

There are two 'basic' Print Template wizards: one for a formal letter, and one for a postcard.

Postcard

The Postcard Wizard lets you choose a page size and two background images, one for the front and one for the back of the postcard.

When you click **Finish**, the Wizard creates:

- A Print context with one section in it, that has duplex printing (printing on both sides) enabled. See "Printing on both sides" on page 627.
- Two Master Pages that each contain a background image. The first Master Page is applied to the front of every page in the Print section. The second Master Page is applied to the back of every page in the Print section. See "Master Pages" on page 648.

- **Scripts** and **selectors** for variable data. The **Scripts** pane shows, for example, a script called "first_name". This script replaces the text "@first_name@" on the front of the postcard by the value of a field called "first_name" when you open a data set that has a field with that name. See "Variable data in the text" on page 924.
- A script called Dynamic Front Image Sample. This script shows how to toggle the image on the front page dynamically. See also "Writing your own scripts" on page 1064.
- One empty Media. Media, also called Virtual Stationery, can be applied to all pages in the Print section. See "Media" on page 652.

The Wizard opens the Print section, so that you can fill it with text and other elements; see "Content elements" on page 761. It already has two Positioned Boxes on it: one on the front, for text, and one on the back, for the address.

See "Print context" on page 624 and "Print sections" on page 629 for more information about Print templates.

Formal letter

The Formal Letter Wizard first lets you select the page settings, see "Page settings: size, margins and bleed" on page 642.

These settings are fairly self-explanatory, except perhaps these:

- Duplex means double-sided printing.
- The margins define where your text flow will go. The actual printable space on a page depends on your printer.
- The bleed is the printable space **around** a page. It can be used on some printers to ensure that no unprinted edges occur in the final trimmed document. Printers that can't print a bleed will misinterpret this setting. Set the bleed to zero to avoid this.
- The number of sections is the number of parts in the Print context. Although this Template wizard can add multiple Print sections to the Print context, it will only add content to the first section.

On the next settings page (click **Next** to go there), you can type a subject, the sender's name and the sender's title. These will appear in the letter. You can also:

- Click the **Browse** button to select a signature image. This image will appear above the sender's name and title.
- Select Virtual Stationery: a PDF file with the letterhead stationery. Also see Media.

When you click **Finish**, the Wizard creates:

- A Print context with one section in it; see "Print context" on the facing page and "Print sections" on page 629.
- One empty Master Page. Master Pages are used for headers and footers, for images and other elements that have to appear on more than one page, and for special elements like tear-offs. See "Master Pages" on page 648.
- One Media. You can see this on the **Resources** pane: expand the **Media** folder. **Media 1** is the Virtual Stationery that you have selected in the Wizard. It is applied to all pages in the Print section, as can be seen in the Sheet Configuration dialog. (To open this dialog, expand the **Contexts** folder on the **Resources** pane; expand the **Print** folder and right-click "Section 1"; then select **Sheet Configuration**.) See "Media" on page 652.
- **Selectors** for variable data, for example: @Recipient@. You will want to replace these by the names of fields in your data. See "Variable data in the text" on page 924.

The Wizard opens the Print section. You can add text and other elements; see "Content elements" on page 761.

The formal letter template already has an address on it. The address lines are paragraphs, located in one cell in a table with the ID **address-block-table**. As the table has no borders, it is initially invisible. The address lines will stick to the bottom of that cell, even when the address has fewer lines. See "Styling and formatting" on page 879 to learn how to style elements.

> **Tip:** Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

ERP templates

The ERP template wizard creates a business document. There is a collection of business documents that you can choose from: Sales Invoice, Purchase Order, Collection Letter, etc.. Currently all of these documents follow the corporate style designed by Microspective.

The first page of the wizard lets you select the page settings, see "Page settings: size, margins and bleed" on page 642. A few clarifications:

- Duplex means double-sided printing.
- The margins define where your text flow will go. The actual printable space on a page depends on your printer.
- The bleed is the printable space **around** a page. It can be used on some printers to ensure that no unprinted edges occur in the final trimmed document. Printers that can't print a bleed will misinterpret this setting. Set the bleed to zero to avoid this.

- The number of sections is the number of parts in the Print context. Although this Template wizard can add multiple Print sections to the Print context, it will only add content to the first section.

On the next settings page (click **Next** to go there):

- Choose the desired type of business document from the **General** drop-down.
- Select a color for the colored parts of the document; see "Color Picker" on page 1143.
- Enter your contact details.
- Click the **Browse** button to select a logo, or select to use a placeholder logo or no logo at all.
- Select a PDF file with the letterhead stationery. Also see "Media" on page 652.

Tip: Your info and preferences are saved and will be reused the next time you create an ERP template.

When you click **Finish**, the Wizard creates:

- A Print context with one section in it; see "Print context" below and "Print sections" on page 629.
- One Master Page. Master Pages are used for headers and footers, for images and other elements that have to appear on more than one page, and for special elements like tear-offs. See "Master Pages" on page 648.
- One Media. You can see this on the **Resources** pane: expand the **Media** folder. **Media 1** is the Virtual Stationery that you have selected in the Wizard. It is applied to all pages in the Print section, as can be seen in the Sheet Configuration dialog. (To open this dialog, expand the **Contexts** folder on the **Resources** pane; expand the **Print** folder and right-click "Section 1"; then select **Sheet Configuration**.) See "Media" on page 652.
- **Selectors** for variable data, for example: @Name@, @Amount@. You will want to replace these by the names of actual fields in your data. See "Variable data in the text" on page 924.

The Wizard opens the Print section. You can add text and other elements; see "Content elements" on page 761. See "Styling and formatting" on page 879 to learn how to style elements..

Print context

The Print context is the folder in the Designer that can contain one or more Print templates.

Print *sections*) are part of the Print context. They are meant to be printed directly to a printer or a printer stream/spool file, or to a PDF file (see "Generating Print output" on page 1639).
The Print context can also be added to Email output as a PDF attachment; see "Generating Email output" on page 1673.
When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record.

Creating the Print context

You can start creating a Print template with a Wizard (see "Creating a Print template with a Wizard" on page 618). Or you can add the Print context to an existing template (see "Adding a context" on page 611).

Tip: Editing PDF files in the Designer is not possible, but when they're used as a section's background, you can add text and other elements, such as a barcode, to them.
The quickest way to create a Print template based on a PDF file is to right-click the PDF file in the Windows Explorer and select **Enhance with Connect**. Alternatively, start creating a new Print template with a Wizard, using the PDF-based Print template (see "Creating a Print template with a Wizard" on page 618).
To use a PDF file as background image for an existing section, see "Using a PDF file or other image as background" on page 634.

When a Print template is created, the following happens:

- The Print context is created and one **Print section** is added to it. You can see this on the **Resources** pane: expand the **Contexts** folder, and then expand the **Print** folder.
  The Print context can contain multiple sections: a covering letter and a policy, for example, or one section that is meant to be attached to an email as a PDF file and another one that is going to be printed out on paper. Only one Print section is added to it at the beginning, but you can add as many print sections as you need; see "Adding a Print section" on page 630.
  See "Print sections" on page 629 to learn how to fill a Print section.
- One **Master Page** is added to the template, as can be seen on the **Resources** pane, in the **Master Page** folder.

  In Print sections, there are often elements that need to be repeated across pages, like headers, footers and logos. In addition, some elements should appear on each first page, or only on pages in between the first and the last page, or only on the last page.

Examples are a different header on the first page, and a tear-off slip that should show up on the last page.

This is what Master Pages are used for. Master Pages can only be used in the Print context.

See "Master Pages" on page 648.

Initially, the (empty) master page that has been created with the Print context will be applied to all pages in the Print section, but more Master Pages can be added and applied to different pages.

- One **Media** is added to the template, as is visible on the **Resources** pane, in the **Media** folder. This folder can hold the company's stationery in the form of PDF files. When applied to a page in a Print section, Media can help prevent the contents of a Print section from colliding with the contents of the stationery. See "Media" on page 652 to learn how to add Media and, optionally, print them.
  Initially, the (empty) media that has been created with the Print context, is applied to all pages in the Print section. You can add more Media and apply them each to different pages.
- One **Stylesheet**, named context_print_styles.css, is added to the template, as you can see on the Resources pane, in the **Stylesheets** folder. This stylesheet is meant to be used for styles that are only applied to elements in the Print context. See also "Styling templates with CSS files" on page 881.

Print settings in the Print context and sections

The following settings in the Print context and Print sections have an impact on how the Print context is printed.

Arranging and selecting sections

The Print context can contain one or more Print sections. When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record. The sections are added to the output in the order in which they appear on the **Resources** pane. This order can be changed; see "Print sections" on page 629.

It is also possible to exclude sections from the output, or to include a section only on a certain condition that depends on a value in the data; see "Conditional Print sections" on page 962.
This can also be done using a Control Script; see "Control Scripts" on page 1098.

Printing on both sides

To print a Print section on both sides of the paper, that Print section needs to have the Duplex printing option to be enabled; see "Enabling double-sided printing (Duplex, Mixplex)" on page 639. This setting can not be changed in a Job Preset or an Output Preset.

> **Note:** Your printer must support duplex for this option to work.

Setting the binding style for the Print context

The Print context , as well as each of the Print sections, can have its own Finishing settings. In printing, Finishing is the way pages are bound together after they have been printed. Which binding styles can be applied depends on the type of printer that you are using.

To set the binding style of the Print **context**:

1. On the **Resources** pane, expand the **Contexts** folder; then right-click the **Print** context and select **Finishing**. Alternatively, select **Context > Finishing** on the main menu. This option is only available when editing a Print section in the Workspace.
2. Choose a Binding style and, if applicable, the number of holes. For an explanation of all Binding and Hole making options, see "Finishing options" on page 1366.

To set the binding style of a Print **section**, see "Setting the binding style for a Print section" on page 639.

Overriding binding styles in a Job Preset

A *Job Preset* can override the binding styles set for the Print sections and for the Print context as a whole. To bind output in another way than defined in the template's settings:

1. Create a Job Preset that overrides the settings of one or more sections: select **File > New** and see "Job Preset" on page 1359 for more details.
2. Select that Job Preset in the Print wizard, or send it to Workflow and select it in the task that creates Print content; see "Generating Print output" on page 1639.

Setting the bleed

The **bleed** is the printable space around a page. It can be used on some printers to ensure that no unprinted edges occur in the final trimmed document. The bleed is one of the settings for a section. See "Page settings: size, margins and bleed" on page 642.

Overprint and black overprint

Normally, when two colors overlap in Print output, the underlying color is not printed. It is "knocked out", for two reasons: firstly, the underlying color may affect the top color, especially if the top color is lighter than the underlying color. Secondly, not printing an underlying color, which is not visible anyway, will save ink or toner.
However, there are cases when underlying colors should not be knocked out:

- If the top color is a **special ink or toner**, such as varnish or UV, it should go over any other colors, as it is meant to be transparent and go over other content. In Connect you may enable overprint when you define a spot color; see "Defining colors, spot colors and tints" on page 913.
- If **small black text** is printed over a colored area, mis-registration may cause white areas to be visible around the text if the underlying color is knocked out.

The option to print small black text over other colors is referred to as **black overprint**. To enable black overprint for text smaller than a given size:

1. Right-click the **Print context** in the **Resources** pane and select **Color Output**, or select the Menu **Context > Color Output** option.Select the Menu **Context > Color Output** option.
2. In the **Text smaller than** field, enter a text size (in points, for example: 3pt).

Note: Black overprint only works with CMYK black.

Converting RGB black to CMYK black

In Print output, black is by default output as a CMYK color. RGB black is converted into CMYK black.
If that is not desirable for any reason, it is possible to prevent it:

1. Right-click the **Print context** in the **Resources** pane and select **Color Output**, or select the Menu **Context > Color Output** option.Select the Menu **Context > Color Output** option.
2. Enable the **Keep RGB black in output** option.

In Connect versions prior to 2018.2, RGB black was not automatically converted to CMYK black. Therefore, this option is by default enabled in templates made with an earlier version. In new templates, this option is disabled by default.

Print sections

Print *sections*) are part of the Print context. They are meant to be printed directly to a printer or a printer stream/spool file, or to a PDF file (see "Generating Print output" on page 1639).
The Print context can also be added to Email output as a PDF attachment; see "Generating Email output" on page 1673.
When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record.

Pages

Unlike emails and web pages, Print sections can contain multiple *pages*. Pages are naturally limited by their size and margins. If the content of a section does not fit on one page, the overflow goes to the next page. This happens automatically, based on the section's page size and margins; see "Page settings: size, margins and bleed" on page 642.
The minimum number of pages can be set via the Print section properties; see "Print section properties" on page 1200.

Although generally the same content elements can be used in all three contexts (see "Content elements" on page 761), the specific characteristics of pages make it possible to use special elements, such as page numbers; see "Page numbers " on page 644.

See "Pages" on page 641 for an overview of settings and elements that are specific for pages.

Using headers, footers, tear-offs and repeated elements

In Print sections, there are often elements that need to be repeated across pages, like headers, footers and logos. In addition, some elements should appear on each first page, or only on pages in between the first and the last page, or only on the last page. Examples are a different header on the first page, and a tear-off slip that should show up on the last page.

This is what Master Pages are used for. Master Pages can only be used in the Print context.

See "Master Pages" on page 648 for an explanation of how to fill them and how to apply them to different pages.

Using stationery (Media)

When the output of a Print context is meant to be printed on paper that already has graphical and text elements on it (called stationery, or preprinted sheets), you can add a copy of this media, in the form of a PDF file, to the Media folder.

Media can be applied to pages in a Print section, to make them appear as a background to those pages. This ensures that elements added to the Print context will correspond to their correct location on the preprinted media.

> **Note:** When both Media and a Master Page are used on a certain page, they will both be displayed on the Preview tab of the workspace, the Master Page being 'in front' of the Media and the Print section on top. To open the Preview tab, click it at the bottom of the Workspace or select **View > Preview View** on the menu.

See "Media" on page 652 for a further explanation about how to add Media and how to apply them to different pages.

Note: The Media will not be printed, unless this is specifically requested through the printer settings; see "Generating Print output" on page 1639.

Copy Fit

Copy Fit is a feature to automatically adjust the font size of text to make it fit the available space. It could be used for the name of a person on a greeting card, for instance, or for the name of a product on a shelf talker. This feature is only available with Box and Div elements in Print sections.

For more information about this feature see "Copy Fit" on page 896.

Adding a Print section

The Print context can contain multiple sections: a covering letter and a policy, for example, or one section that is meant to be attached to an email as a PDF file and another one that is meant to be printed out on paper. When you create a Print template (see "Creating a Print template with a Wizard" on page 618 and "Print context" on page 624), you can select the number of print sections you want to start with. You can always add as many Print sections as you need later.

To add a section to a context:

- On the **Resources** pane, expand the **Contexts** folder, right-click the **Print** context , and then click **New section**.

  Or, right-click an existing Print section and select **Duplicate**.

Note that the new section automatically gets the same properties as the first section.

The first Master Page (see "Master Pages" on page 648) and Media (see "Media" on page 652) will automatically be applied to all pages in the new section, but this can be changed, see "Applying a Master Page to a page in a Print section" on page 651 and "Applying Media to a page in a Print section" on page 656.

Note that Print **sections** always start on a front page. To avoid empty pages between sections, consider combining them into one section, and use page breaks to push content to the next page (see "Page breaks" on page 647).

Tip: Editing PDF files in the Designer is not possible, but when they're used as a section's background, you can add text and other elements, such as a barcode, to them.
The quickest way to create a Print template based on a PDF file is to right-click the PDF file in the Windows Explorer and select **Enhance with Connect**. Alternatively, start creating a new Print template with a Wizard, using the PDF-based Print template (see "Creating a Print template with a Wizard" on page 618).
To use a PDF file as background image for an existing section, see "Using a PDF file or other image as background" on page 634.

Tip: If you need a whole Print section to be visible in the output only under certain conditions, consider using the Conditional Print Section script wizard; see "Conditional Print sections" on page 962.
You can use the Conditional Content script wizard to hide parts of the content of a section; see "Showing content conditionally" on page 958.

Cloning a Print section

If documents in the output need to have a **variable number of sections**, based on data, Print sections can be cloned (copied) on the fly. See "Cloning sections" on page 965.

Importing a Print section

To import a section from another template, click **File > Import > Connect Resources...** in the menu. See: "Import Resources dialog" on page 1155.
Remember also to add or import any related source files, such as images.

Note that when the imported Print section *replaces* a Print section in your template, its context's Color Output and Finishing settings get imported as well. (See "Print settings in the Print context and sections" on page 626.)

Importing a Word file into a section

To import a Word file into a Print section, open the Print section and then click **File > Import > Word** in the menu.

Note: When importing from Word we substitute the font “Aptos” (a proprietary cloud font that cannot be used by a browser) with the more general “Calibri”.

The following will be added to the template:

- Image files for each image in the original file. As with other templates, you can place images on the master page if you want them to appear in every document in the same place.
- A folder to contain the imported image files.
- A separate .css file in the Stylesheets for every stylesheet that is imported with the MS Word file. The css files are linked to the active section. They can be edited (see "Styling your templates with CSS files" on page 885).

Microsoft Word documents with mail merge fields

If the Microsoft Word document contains mail merge fields, these mail merge fields (or markers) are added to the Data Model of the OL Connect template.

Select **File > Data > Open Data File...** to import the corresponding data.

Or create a data mapping configuration to fill the Data Model with actual data.

In the **template**, the mail merge fields are replaced with *expressions*. (See: "Handlebars expressions" on page 1005.)

- The brackets from the mail merge fields are converted to double curly braces.

If expressions are not supported in the current section, *placeholders* and their associated scripts are added instead of expressions.

- The brackets from the mail merge fields are converted to the @ character.
- The variable is wrapped with a span element.
- A user script is created for each data field.

Note: To enable expressions in a section, right-click the section in the Resources pane, select **Properties** and check the option **Evaluate Handlebars expressions**. In templates made with OL Connect 2022.1 or older this option is unchecked by default.

Limitations

The conversion of a Word file to a template will most likely not be perfect. Here are some known limitations:

- If an image is cropped it will appear stretched in OL Connect.
- The position of an object is incorrect if it is positioned relative to something else (e.g. 100 points below a paragraph).

- Text wrapping is not supported in OL Connect. What you see in OL Connect generally corresponds to the layout option "In Line with Text" in MS Word.
- Lines and borders around objects are not supported.
- Headers and footers are not supported.

Deleting a Print section

To delete a Print section:

- On the **Resources** pane, expand the **Contexts** folder, expand the **Print** context, right-click the name of the section, and then click **Delete**.

Caution: If you don't have a backup of the template, the only way to recover a deleted section, is to click **Undo** on the **Edit** menu, until the deleted section is restored. After closing and reopening the template it is no longer possible to restore the deleted context this way.
In the Saving Preferences you can set whether a backup file should be created when you save the template; see "Save preferences" on page 1059.

Arranging Print sections

When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record. The sections are added to the output in the order in which they appear on the **Resources** pane, so changing the order of the sections in the Print context changes the order in which they are outputted to the final document.

To rearrange sections in a context:

- On the **Resources** pane, expand the **Print** context and drag and drop sections to change the order they are in.
- On the **Resources** pane, right-click a Print section and click **Arrange**. In the Arrange Sections dialog you can change the order of the sections by clicking the name of a section and moving it using the **Up** and **Down** buttons.

Styling and formatting a Print section

The contents of a Print section can be formatted directly, or styled with Cascading Style Sheets (CSS). See "Styling and formatting" on page 879.

In order for a style sheet to be applied to a specific section, it needs to be included in that section. There are two ways to do this.

Drag & drop a style sheet

1. Click and hold the mouse button on the style sheet on the **Resources** pane.
2. Move the mouse cursor within the **Resources** pane to the section to which the style sheet should be applied.
3. Release the mouse button.

Using the Includes dialog

1. On the Resources pane, right-click the section, then click **Includes**.
2. From the **File types** dropdown, select **Stylesheets**.
3. Choose which CSS files should be applied to this section. The available files are listed at the left. Use the arrow buttons to move the files that should be included to the list at the right.
4. You can also change the order in which the CSS files are read: click one of the included CSS files and use the **Up** and **Down** buttons. Note that moving a style sheet up in the list gives it **less** weight. In case of conflicting rules, style sheets read later will override previous ones.

Note: Style sheets are applied in the order in which they are included in a section. The styles in each following style sheet add up to the styles found in previously read style sheets. When style sheets have a conflicting rule for the same element, class or ID, the **last** style sheet 'wins' and overrides the rule found in the previous style sheet.

Note: Style sheets that are linked to (i.e. included in) a section show a chain icon in the Resources pane (see "Resources pane" on page 1250).

Using a PDF file or other image as background

In the Print context, a PDF file can be used as a section's background. It is different from the Media in that the section considers the PDF to be content, so the number of pages in the section will be the same as the number of pages taken from the PDF file.

With this feature it is possible to create a Print template from an arbitrary PDF fileor from a PDF file provided by the DataMapper. Of course, the PDF file itself can't be edited in a Designer template, but when it is used as a section's background, text and other elements, such as a barcode, can be added to it.

Encrypted PDF files are **not supported** in *PDF pass-through* mode.

Select a PDF file as background image

To use a PDF file as background image:

1. On the **Resources** pane, expand the **Print** context, right-click the print section and click **Background**.

   On the **Resources** pane, expand the **Sections** folder, right-click the print section and click **Background**.

2. Click the downward pointing arrow after **Image** and select either **From Datamapper input** or **From PDF resource**.
   From Datamapper input uses the active data mapping configuration to retrieve the PDF file that was used as input file, or another type of input file, converted to a PDF file. With this option you don't need to make any other settings; click OK to close the dialog.

3. For a PDF resource, you have to specify the **path**. Clicking the **Select Image** button opens the Select Image dialog (see "Select Image dialog" on page 1206).

   Click **Resources**, **Disk** or **Url**, depending on where the image is located.

   - **Resources** lists the images that are present in the **Images** folder on the **Resources** pane.
   - **Disk** lists image files that reside in a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder (or an image in a folder).
     As an alternative it is possible to enter the path manually. You can give a local path (e.g. C:\Images\Test.jpg) or use the "file" protocol.

     The complete syntax of a fully qualified URL with the "file" protocol is: `file://<host>/<path>`. Note: if the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: `file:///c:/resources/images/image.jpg`.
     If the file is located on another server in your network, the path must contain five slashes after "file:".

     > **Note:** Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file:////myserver/myfolder/file.txt).

   - **Url** lists image files from a specific web address. Select the protocol (**http** or **https**), and then enter a web address (for example, http://www.mysite.com/images/image.jpg).

   > **Note:** If a URL doesn't have a file extension, and the option Save with template is **not** selected, the Select Image dialog automatically adds the `filetype`

parameter with the file extension as its value (for example: `?filetype=pdf` (if it is the first parameter) or `&filetype=pdf`).
The `filetype`, `page` and `nopreview` parameters are not sent to the host; they are used internally. Therefore, URLs that rely on one of these parameters cannot be used.

With an external image, you can check the option **Save with template**. If this option is checked, the file will be inserted in the **Images** folder on the **Resources** pane at the top left.
If it isn't saved with the template, the image remains external. Note that external images need to be available when the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced. External images are updated (retrieved) at the time the output is generated.

Insertion Point group box:

Selects where the image is to be inserted.

- **Absolute** checkbox: Select to make the inserted image an Absolutely positioned one.
  Selecting **Absolute** prevents subsequent selection of **Location** and **Element**.
- **Location** pull down option: Select the Location at which to insert the image.
- **Elements** pull down option: Select the **Element** that the **Location** option is to apply to.
  Not available for **Location** selection of *At cursor position*.

Note: Any non-content features in the original PDF will get stripped out of the original PDF. These include document navigation (outline), local links, hyperlinks, annotations (markup annotations, comments, etc.), digital signatures, multimedia features, interactive forms, and tagged PDF features (marks in the page content will stay, but the logical structure is gone).

Settings for a background image

Once a background image has been selected (see above), you can make settings for it.

If you want to make settings for a previously added background, expand the **Sections** folder on the **Resources** pane, right-click the print section and click **Background**.

1. Select the PDF's **position**:
    - **Fit to page** stretches the PDF to fit the page size.
    - **Centered** centers the PDF on the page, vertically and horizontally.
    - **Absolute** places the PDF at a specific location on the page. Use the **Top** field to specify the distance between the top side of the page and the top side of the PDF, and the **Left** field to specify the distance between the left side of the page and the left side of the PDF.
      The Top and Left offset can be specified in the usual units of measurement or as a percentage of the page (for example: a Left value of 25% means it will be placed at 25% of the page width).
2. Set the **scale** of the width (x) and height (y) of the image as a percentage of the original image. To avoid stretching, check the option **Keep aspect ratio** and set the scale of the width.
3. Click one of the options next to **Rotation** to rotate the image.
4. Optionally, if the PDF has more than one page, you can set the range of **pages** that should be used.

    Note: The number of pages in the Print section is automatically adjusted to the number of pages in the PDF file that are being used as the section's background image.

5. Define a **clipped region** if the image should be **displayed** or **hidden** only in a certain region of the page.

    Tip: This setting is particularly useful when the background PDF has text that is meant to be extracted but should be invisible to the user. For an example, see this how-to: Section background image clipping.

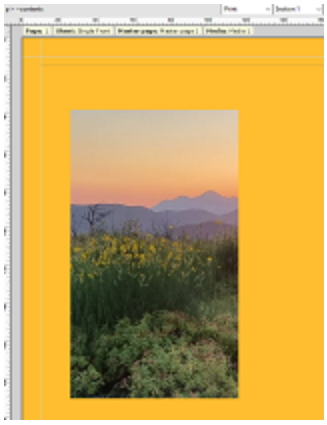
a. Check the option **Clipped region**.

b. Enter the **Top**, **Left**, **Width** and **Height** of the region. Valid measurement units are: cm, px, in, mm, pc, and pt.

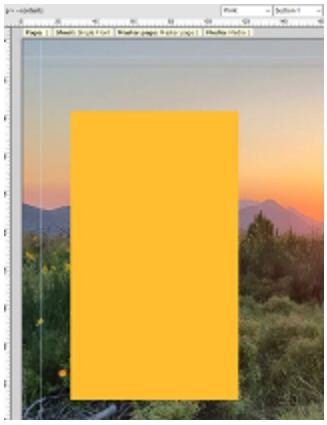
    Note that the Top, Left, Width and Height of the region are relative to the page. They are not affected by any offset, scale, or rotation that may be applied to the image.

c. Click **Invert region** to hide the image within the specified region.

Not inverted:



Inverted:



d. Select whether you want to apply this setting to the **first page** or to **all pages** in the selected background image.

> **Note:** This setting affects only what is visible of the image. The image itself is not changed.

6. Finally, click **OK**.

> **Tip:** An alternative to using a PDF as background inside the template is to layer the template (i.e. the PDF output of a Print section) over the background PDF via a Script task in a Workflow process. This is called 'stamping'. In the unusual case where extracting text from the PDF that is the output of a Print section with a PDF background doesn't work, it is recommended to use this method. For more information, see the online help of Workflow: Stamping one PDF file on another.

> **Note:** If the PDF doesn't look as expected, it may be missing one or more external fonts. See "Providing missing fonts for PDF" on page 354.

Dynamic backgrounds

To make the background change based on the value of a data field, you may use the Dynamic Background Script Wizard; see "Dynamic Print section backgrounds" on page 994.

Alternatively you could write your own Control Script to set the background; see "Control Script: Setting a Print section's background" on page 1106.

The settings in a script take precedence over the settings made in the Print Section Properties dialog.

Setting the binding style for a Print section

In printing, Finishing is the binding style, or the way pages are bound together. Each Print section can have its own Finishing settings, as well as the Print context as a whole; see "Setting the binding style for the Print context" on page 627.

To set the binding style of a Print **section**:

1. On the **Resources** pane, expand the **Contexts** folder, expand the **Print** context and right-click the Print section.
2. Click **Finishing**.
3. Choose a Binding style and, if applicable, the number of holes.

Overriding binding styles in a Job Preset

A *Job Preset* can override the binding styles set for the Print sections and for the Print context as a whole. To bind output in another way than defined in the template's settings:

1. Create a Job Preset that overrides the settings of one or more sections: select **File > New** and see "Job Preset" on page 1359 for more details.
2. Select that Job Preset in the Print wizard, or send it to Workflow and select it in the task that creates Print content; see "Generating Print output" on page 1639.

Enabling double-sided printing (Duplex, Mixplex)

To print a Print section on both sides of the paper, that Print section needs to have the Duplex printing option to be enabled. This is an option in the Sheet Configuration dialog. (See "Sheet Configuration dialog" on page 1208.)

> **Note:** Your printer must support Duplex for this option to work.

To enable Duplex or Mixplex printing:

1. On the **Resources** pane, expand the **Print** context, right-click the print section and click **Sheet configuration**.
2. Check **Duplex** to enable content to be printed on the back of each sheet.
3. When Duplex printing is enabled, further options become available.
    - Check **Omit empty back side for Last or Single sheet** to reset a page to Simplex if it has an empty back side. **This changes the Duplex job into a Mixplex job.** Thus changing a Duplex job into a **Mixplex** job may reduce volume printing costs as omitted back sides aren't included in the number of printed pages. On the other hand, depending on the printer type it may reduce the print speed as well. See also: "The consequences of empty back sides for printing and page numbering" below.

      Note: Use a print format that supports Duplex/Simplex switching to get a proper Mixplex job. PDF is not suitable as it has no device control.

      Print **sections** always start on a front page. To avoid empty pages between sections, consider combining them into one section, and use page breaks to push content to the next page (see "Page breaks" on page 647).
    - Check **Tumble** to duplex pages as in a calendar.
    - Check **Facing pages** to have the side margins switched alternately, so that after printing and binding the pages, they look like in a magazine or book. See "Pages" on the next page to find out how to set a left and right margin on a page.

Note: Master Pages, Media and Duplex printing options can also be set in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). This is especially useful when you need identical sections with different settings.

The consequences of empty back sides for printing and page numbering

In a Duplex job, the last page of a section may be empty since **each new section starts on a new sheet**. You may wonder what this means for the number of 'print clicks' and for the page numbering.

Note that an empty page is defined as a page that has no content **and** no Master Page. To suppress the Master Page on otherwise empty back sides, check the option **Omit Master Page Back in case of an empty back page** for last and single sheets, in the Sheet Configuration dialog. (See "Sheet Configuration dialog" on page 1208.)

As of version 2020.2, a page that only has a DataMapper PDF background is no longer seen as empty. This may affect the output of templates created with previous versions.

Print clicks

If a page is empty, but still sent to a printer, it may be counted as a 'click' on the printer. To avoid this, you could check the **Omit empty back side for Last or Single sheet** option in the Duplex printing settings. This resets a page to Simplex if it has an empty back side. The resulting print job can then be "mixed plex"; in that contains both simplex and duplex pages.
The omitted back side isn't sent to the printer, so it doesn't count as a print click.
If a page is empty, but not omitted (the Omit empty back side option is not checked) it will still be sent to the printer and may count as a print click.

Page numbers

An empty back side that is omitted from the output does not count in the page numbers either.

If a back side is empty, but not omitted (the Omit empty back side for Last or Single sheet option is not checked) it will be skipped from the page count *unless* the page numbers continue on the next section (see "Configuring page numbers" on page 645).

Pages

Unlike emails and web pages, Print sections can contain multiple *pages*. Pages are naturally limited by their size and margins. If the content of a section does not fit on one page, the overflow goes to the next page. This happens automatically, based on the section's page size and margins; see "Page settings: size, margins and bleed" on the facing page.
The minimum number of pages can be set via the Print section properties; see "Print section properties" on page 1200.

Although generally the same content elements can be used in all three contexts (see "Content elements" on page 761), the specific characteristics of pages make it possible to use special elements, such as page numbers; see "Page numbers " on page 644.

The widow/orphan setting lets you control how many lines of a paragraph stick together, when content has to move to another page; see "Preventing widows and orphans" on page 646. You can also avoid or force a page break before or after an entire element, see "Page breaks" on page 647.

Each page in a print section has a natural position: it is the first page, the last page, a 'middle' page (a page between the first and the last page) or a single page. For each of those positions, a different Master Page and Media can be set. A Master Page functions as a page's background, with for example a header and footer. A Media represents preprinted

paper that a page can be printed on. See "Master Pages" on page 648 and "Media" on page 652.

Page specific content elements

The specific characteristics of pages make it possible to use these special elements:

- **Page numbers** can only be used in a Print context. See "Page numbers " on page 644 to learn how to add and change them.
- Conditional content and dynamic tables, when used in a Print section, may or may not leave an empty space at the bottom of the last page. To fill that space, if there is any, an image or advert can be used as a **whitespace element**; see "Whitespace elements: using optional space at the end of the last page" on the next page.
- Dynamic Tables can be used in all contexts, but **transport lines** are only useful in a Print context; see "Dynamic Table" on page 974.

Positioning and aligning elements

Sometimes, in a Print template, you don't want content to move up or down with the text flow. To prevent that, put that content in a Positioned Box. See "Content elements" on page 761.

When it comes to positioning elements on a page, Guides can be useful, as well as Tables. See "How to position elements" on page 897.

> **Tip:** In Print sections, an Inline Box can be aligned to the bottom of the last page. See "Align a box to the bottom of the last page" on page 899 for instructions.

Page settings: size, margins and bleed

On paper, whether it is real or virtual, content is naturally limited by the page size and margins.

These, as well as the bleed, are set per Print section, as follows:

- On the **Resources** pane, right-click a section in the **Print** context and click **Properties**.

For the **page size**, click the drop-down to select a page size from a list of common paper sizes. Changing the width or height automatically sets the page size to Custom.

Margins define where your text flow will go. Static elements can go everywhere on a page, that is to say, within the printable space on a page that depends on the printer.

The **bleed** is the printable space around a page. It can be used on some printers to ensure that no unprinted edges occur in the final trimmed document. Note: Printers that can't print a bleed, will misinterpret this setting. Set the bleed to zero to avoid this.

> **Tip:** By default, measurements settings are in inches (in). You could also type measures in centimeters (add 'cm' to the measurement, for example: 20cm) or in millimeters (for example: 150mm).
> To change the default unit for measurement settings to centimeters or millimeters: on the menu, select **Window > Preferences > Print > Measurements**.

Whitespace elements: using optional space at the end of the last page

Print sections with conditional content and dynamic tables (see "Personalizing content" on page 923) can have a variable amount of space at the bottom of the last page. It is useful to fill the empty space at the bottom with transpromotional material, but of course you don't want extra pages created just for promotional data. 'Whitespace elements' are elements that will only appear on the page if there is enough space for them.

To convert an element into a whitespace element:

1. Import the promotional image or snippet; see "Images" on page 852 and "Snippets" on page 867.
2. Insert the promotional image or snippet in the content.

   > **Note:** Only a top-level element (for example, a paragraph that is not inside a table or div) can function as a whitespace element.
   >
   > Do not place the promotional image or snippet inside an absolute positioned box. Whitespacing only works for elements that are part of the text flow, not for absolute-positioned boxes.

3. Select the image or the element that holds the promotional content: click it, or use the breadcrumbs, or select it on the **Outline** tab; see "Selecting an element" on page 766.
4. On the **Attributes** pane, check the option **Whitespace element**.
5. (Optional.) Add extra space at the top of the element: on the menu **Format**, click the option relevant to the selected element (Image for an image, Paragraph for a paragraph, etc.) and adjust the spacing (padding and/or margins).
   Do not add an empty paragraph to provide space between the whitespace element and the variable content. The extra paragraph would be considered content and could end up on a separate page, together with the whitespace element.

Tip: To always align the whitespace element to the bottom of the page, select it, and type `alignbottom` in the **Class** field on the **Attributes** pane.

Page numbers

Inserting page numbers

Page numbers can be added to a Print section, but they are usually added to a Master Page, because headers and footers are designed on Master Pages; see also: "Master Pages" on page 648.

To insert a page number, select **Insert > Special character > Markers** on the menu, and then click one of the options to decide with what kind of page number the marker will be replaced:

- **Page number**: The current page number in the document. If a page is empty or does not display a page number, it is still added to the page count.
- **Page count**: The total number of pages in the document, including pages with no contents or without a page number.
- **Content page number**: The current page number in the document, counting only pages with contents that are supplied by the Print section. A page that has a Master Page (as set in the Sheet Configuration dialog, see "Applying a Master Page to a page in a Print section" on page 651) but no contents, is not included in the Content page count.
- **Content page count**: This is the total number of pages in the current document that have contents, supplied by the Print section. A page that has a Master Page but no contents, is not included in the Content page count.
- **Sheet number**: The current sheet number in the document. A sheet is a physical piece of paper, with two sides (or pages). This is equivalent to half the page number, for example if there are 10 pages, there will be 5 sheets.
- **Sheet count**: This marker is replaced by the total number of sheets in the document, whether or not they have contents.

Note: When a marker is inserted, a class is added to the element in which the marker is inserted. Do not delete that class. It enables the software to quickly find and replace the marker when generating output. The respective classes are: `pagenumber`, `pagecount`, `contentpagenumber`, `contentpagecount`, `sheetnumber`, and `sheetcount`.

Tip: Instead of page numbers, you might want to display the current **record index** in the document. This can be done with a Handlebars helper: "counter" on page 1010.

Creating a table of contents

A table of contents can only be created in a script.
If you are looking to create a short, simple table of contents in **one section**, you could add a Standard Script that uses the `pageRef()` function. For an example, see "pageref()" on page 1580.
For a **multi-page**, **cross-section** table of contents you must use a Post Pagination Script; see "Creating a Table Of Contents" on page 1116.

The basics of script-writing in the Designer are explained in the following topic: "Writing your own scripts" on page 1064.

Configuring page numbers

By default the page numbers are Arabic numerals (1, 2, 3, etc.) without leading zeros nor prefix, and page numbering starts with page 1 for each section. But this can be changed. To do that:

1. On the **Resources** pane, right-click a section in the **Print** context and click **Numbering**.
2. Uncheck **Restart Numbering** if you want the page numbers to get consecutive page numbers, instead of restarting the page numbering with this section.

   Note: Even if a section is disabled, so it doesn't produce any output, this setting is still taken into account for the other sections. This means that if Restart Numbering is checked on a disabled section, the page numbering will be restarted on the next section.
   Disabling a section can only be done in a Control Script (see "Control Scripts" on page 1098). Control Scripts can also change where page numbers restart.
3. Use the **Format** drop-down to select uppercase or lowercase letters or Roman numerals instead of Arabic numerals.
4. In **Leading Zeros**, type zeros to indicate how many digits the page numbers should have. Any page number that has fewer digits will be preceded by leading zeros.
5. Type the **Number prefix**. Optionally, check Add Prefix to Page Counts, to add the prefix to the total number of pages, too.
6. Close the dialog.

Preventing widows and orphans

Widows and orphans are lines at the beginning or at the end of a paragraph respectively, dangling at the bottom or at the top of a page, separated from the rest of the paragraph. By default, to prevent orphans and widows, lines are moved to the next page as soon as two lines get separated from the rest of the paragraph. The same applies to list items (in unordered, numbered and description lists).
The number of lines that should be considered a widow or orphan can be changed for the entire Print context, per paragraph and in tables.

> **Note:** Widows and orphans are ignored if the **page-break-inside** property of the paragraph is set to **avoid**; see "Preventing a page break" on page 648.

In the entire Print context

To prevent widows and orphans in the entire Print context:

1. On the menu, select **Edit > Stylesheets**.
2. Select the **Print** context.
3. Click **New** (or, when there are already CSS rules for paragraphs, click the selector **p** and click **Edit**).
4. Click **Format**.
5. After **Widows and Orphans**, type the minimum number of lines that should be kept together.

Alternatively, manually set the set the **widows** and **orphans** properties in a style sheet:

1. Open the style sheet for the Print context: on the **Resources** pane, expand the **Styles** folder and double-click context_print_styles.css.
2. Add a CSS rule, like the following:

```
p { widows: 4; orphans: 3 }
```

Per paragraph

To change the widow or orphan setting for one paragraph only:

1. Open the Formatting dialog. To do this, you can:
   - Select the paragraph using the breadcrumbs or the **Outline** pane (next to the **Resources** pane) and then select **Format > Paragraph** in the menu.

- Right-click the paragraph and select **Paragraph...** from the contextual menu.

2. After **Widows** and **Orphans**, type the minimum number of lines that should be kept together.

In tables

The CSS properties **widows** and **orphans** can be used in tables to prevent a number of rows from being separated from the rest of the table.

Dynamic Tables are automatically divided over several pages when needed. A Standard Table doesn't flow over multiple pages by default. Splitting a Standard Table over multiple pages requires setting the Connect-specific `data-breakable` attribute on all of its rows. You can either open the Source tab, or write a script to replace each `<tr>` with `<tr data-breakable="">`. Note that the effect will only be visible in Preview mode.

To set the number of widows and orphans for a table:

1. Open the Formatting dialog. To do this, you can:
   - Select the table using the breadcrumbs or the **Outline** pane (next to the **Resources** pane) and then select **Format > Table** in the menu.
   - Right-click the paragraph and select **Table...** from the contextual menu.
2. After **Widows** and **Orphans**, type the minimum number of table rows that should be kept together.

Page breaks

A page break occurs automatically when the contents of a section don't fit on one page.

> **Note:** Improved page breaking in Connect 2019.1 might impact upon templates made with earlier versions. See "Known Issues" on page 126.

Inserting a page break

To insert a page break before or after a certain element, set the `page-break-before` property or the `page-break-after` property of that element (a paragraph for example; see also "Styling text and paragraphs" on page 892):

1. Select the element (see "Selecting an element" on page 766).
2. On the **Format** menu select the respective element to open the Formatting dialog.
3. In the Breaks group, set the **before** or **after** property.

- **Before**: Sets whether a page break should occur **before** the element. This is equivalent to the **page-break-before** property in CSS; see MDN Web Docs - page-break-before for an explanation of the available options.
- **After**: Sets whether a page break should occur **after** the element. Equivalent to the page-break-after property in CSS; see MDN Web Docs - page-break-after for an explanation of the available options.

Click the button **Advanced** to add CSS properties and values to the inline style tag directly. Alternatively you could set this property on the Source tab in the HTML (for example: `<h1 style="page-break-before: always;">`), or add a rule to the style sheet; see "Styling your templates with CSS files" on page 885.

> **Note:** You cannot use these properties on an empty or on absolute-positioned elements.

Preventing a page break

To prevent a page break inside a certain element, set the **page-break-inside** property of that element to **avoid**:

- Select the element (see "Selecting an element" on page 766).
- On the **Format** menu, select the respective element to open the Formatting dialog.
- In the **Breaks** group, set the **inside** property to **avoid**, to prevent a page break inside the element. For an explanation of all available options of the **page-break-inside** property in CSS, see MDN Web Docs - page-break-inside.

Alternatively you could set this property on the Source tab in the HTML (for example: `<ul style="page-break-inside: avoid;">`), or add a rule to the style sheet; see "Styling your templates with CSS files" on page 885.

Master Pages

In Print sections, there are often elements that need to be repeated across pages, like headers, footers and logos. In addition, some elements should appear only on specific pages, such as only the first page, or the last page, or only on pages in-between. Examples are a different header on the first page, and a tear-off slip that shows up on the last page.

This is what Master Pages are used for. Master Pages can only be used in the Print context (see "Print context" on page 624).

Master Pages resemble Print sections, and they are edited in much the same way (see "Editing a Master Page" below) but they contain a single page and do not have any text flow. Only one Master Page can be applied per page in printed output. Then a Print template is created, one master page is added to it automatically. You can add more Master Pages; see "Adding a Master Page" below. Initially, the original Master Page will be applied to all pages, but different Master Pages can be applied to different pages; see "Applying a Master Page to a page in a Print section" on page 651.

Example

The following how-to demonstrates the use of Master Pages: Showing a Terms and Conditions on the back of the first page only.

Adding a Master Page

When a Print template is created, one master page is added to it automatically. Adding more Master Pages can be done as follows:

- On the **Resources** pane, right-click the **Master pages** folder and click **New Master Page**.
- Type a name for the master page.
- Optionally, set the margin for the header and footer. See "Adding a header and footer" on the facing page.
- Click **OK**.

Initially, the master page that has been created together with the Print context will be applied to all pages in the Print section. After adding more Master Pages, different Master Pages can be applied to different pages; see "Applying a Master Page to a page in a Print section" on page 651.

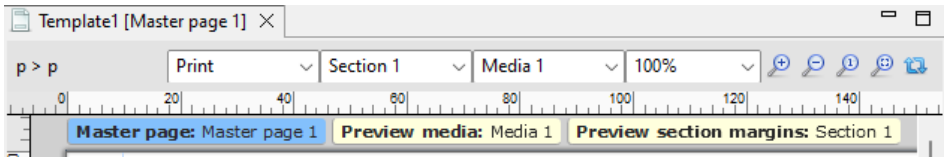
Importing a Master Page

To import one or more Master Pages from another template, click **File > Import Resources...** in the menu. See: "Import Resources dialog" on page 1155.

Editing a Master Page

Master Pages are edited just like sections, in the workspace. To open a Master Page, expand the **Master pages** folder on the **Resources** pane, and double-click the Master Page to open it.

The drop-downs at the top of the Workspace let you select a Section and a Media (front or back) that will serve as a background to your Master Page design.



The blue box below the drop-downs shows which Master Page is being edited.

A Master Page can contain text, images and other elements (see "Content elements" on page 761), including variable data and dynamic images (see "Personalizing content" on page 923). All elements on a Master Page should have an absolute position or be inside an element that has an absolute position. It is good practice to position elements on a Master Page by placing them in a Positioned Box (see "Content elements" on page 761).

Keep in mind that a Master Page always remains a single page. Its content cannot overflow to a next page. Content that doesn't fit, will not be displayed.

> **Note:** Editing the Master Page is optional. One Master Page must always exist in a Print template, but if you don't need it, you can leave it empty.

Adding a header and footer

Headers and footers are not designed as part of the contents of a Print section, but as part of a Master Page, which is then applied to a page in a print section.

To create a header and footer:

1. First insert elements that form the header or footer, such as the company logo and address, on the Master Page; see "Editing a Master Page" on the previous page.
2. Next, define the margins for the header and footer. The margins for a header and footer are set in the Master Page properties. This does not change the content placement within the Master Page itself; in Master Pages, elements can go everywhere on the page. Instead, the header and footer of the Master Page limit the text flow on pages in the Print sections to which this Master Page is applied. Pages in a Print section that use this Master Page cannot display content in the space that is reserved by the Master Page for the header and footer, so that content in the Print section does not collide with the content of the header and footer. To set a margin for the header and/or footer:

a. On the **Resources** pane, expand the **Master pages** folder, right-click the master page, and click **Properties**.

b. Fill out the height of the header and/or the footer. The contents of a print section will

not appear in the space reserved for the header and/or footer on the corresponding master page.

3. Finally, apply the master page to a specific page in a print section. See "Applying a Master Page to a page in a Print section" below.

Applying a Master Page to a page in a Print section

Every sheet in a Print section has a natural position: it can be the **first**, the **last**, one of the sheets in between ('**middle**'), or a **single** sheet. For each of these positions, you can set a different Master Page and Media (see "Media" on the facing page). It can even have two master pages, if printing is done on both sides (called duplex printing).

To apply Master Pages to specific page positions in a Print section:

1. On the **Resources** pane, expand the **Print** context; right-click the Print section, and click **Sheet configuration**.
2. Optionally, check **Duplex** to enable content to be printed on the back of each sheet. Your printer must support duplex for this option to work. If Duplex is enabled, you can also check **Tumble** to duplex pages as in a calendar, and **Facing pages** to have the margins of the section switch alternately, so that pages are printed as if in a magazine or book.
3. If the option **Same for all positions** is checked, the same Master Page will be applied to every page in the print section (and to both the front and the back side of the page if duplex printing is enabled). Uncheck this option.
4. Decide which Master Page should be linked to which sheet (position): click the downward pointing arrow after **Master Page Front** and select a Master Page.
   If Duplex is enabled, you can also select a Master Page for the back of the sheet and consequently, check **Omit Master Page Back in case of an empty back page** to omit the specified Master Page on the last backside of a section if that page is empty. That page will then also be skipped from the page count *unless* the page numbers continue on the next section (see "Configuring page numbers" on page 645).
   Note that if the **Omit empty back side for Last or Single sheet** option (see "General options" on page 1208) is checked as well, the empty backside will not appear in the output at all and will not be counted in any case.
5. Optionally, decide which Media should be linked to each sheet.
6. If output documents can be so long that they cannot fit in one envelope, you may check the **Repeat sheet configuration** option to have the sheet configuration repeat every *n* number of pages.
7. Click OK to save the settings and close the dialog.

Note: Master Pages, Media and Duplex printing options can also be set in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). This is especially useful when you need identical sections with different settings.

Deleting a Master Page

To delete a Master Page, expand the **Master pages** folder on the **Resources** pane, right-click the master page, and click **Delete**.

Note that one Master Page as well as one Media must always exist in a Print template. Just leave it empty if you don't need it.

Media

When the output of a Print context is meant to be printed on paper that already has graphical and text elements on it (called stationery, or preprinted sheets), you can add a copy of this media, in the form of a PDF file, to the Media folder.

Media can be applied to pages in a Print section, to make them appear as a background to those pages. This ensures that elements added to the Print context will correspond to their correct location on the preprinted media.

For further explanation about how to apply Media to different pages, see "Applying Media to a page in a Print section" on page 656.

Media will not be printed, unless you want them to; see below.

Per Media, a front and back can be specified and you can specify on what kind of paper the output is meant to be printed on. This includes paper weight, quality, coating and finishing; see "Setting Media properties" on the next page.

Adding Media

To add a Media, right-click the **Media** folder on the **Resources** pane and select **New Media**.

The new Media is of course empty. You can specify two PDF files for the Media: one for the front, and, optionally, another for the back.

Tip: Right-click an existing Media entry and select **Duplicate** to copy and paste it into the same folder.

Importing Media

To import Media from another template, click **File > Import Resources...** in the menu. See: "Import Resources dialog" on page 1155.

Specifying and positioning Media

Specifying a PDF for the front: the fast way

To quickly select a PDF file for the front of a Media, drag the PDF file from the Windows Explorer to one of the Media. The Select Image dialog opens; select an image and check the option **Save with template** if you want to insert the image into the **Images** folder on the **Resources** pane. (For PDF files selected by URL this option is always checked.)
Alternatively you could first import the PDF file to the **Images** folder on the **Resources** pane (using drag & drop) and drag it from there on one of the Media in the **Media** folder.
Either way, you cannot set any options.

To be able to specify a PDF file for both the front and the back of the Media, and to specify a position for the Media's PDF files, you have to edit the properties of the Media.

Setting Media properties

Media have a number of properties that you can set. You can change the Media's page size and margins (as long as it isn't applied to a section), you can specify a PDF file (or any other type of image file) for both the front and the back of the Media, and you can determine how the virtual stationery should be positioned on the page. This is done as follows:

1. On the **Resources** pane, expand the **Contexts** folder, expand the **Media** folder, right-click the Media and click **Properties**.
2. Now you can change the name and page size of the Media. Note that it isn't possible to change the page size once the Media is applied to a section. Media can only be applied to sections that have the same size.

   Note: Naming this element with a purely numeric name will cause issues with scripting. We recommend using a Alphanumeric name.

3. On the **Virtual Stationery** tab, you can click the **Select Image** button to select a PDF image file.

   Note: Encrypted PDF files are **not supported** in *PDF pass-through* mode. See "PDF Options" on page 1390.

4. Click **Resources**, **Disk** or **Url**, depending on where the image is located.
   - **Resources** lists the images that are present in the **Images** folder on the **Resources** pane.

- **Disk** lists image files that reside in a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder (or an image in a folder).
  As an alternative it is possible to enter the path manually. You can give a local path (e.g. C:\Images\Test.jpg) or use the "file" protocol.

  The complete syntax of a fully qualified URL with the "file" protocol is: `file://<host>/<path>`. Note: if the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: `file:///c:/resources/images/image.jpg`.
  If the file is located on another server in your network, the path must contain five slashes after "file:".

  > **Note:** Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file:////myserver/myfolder/file.txt).

- **Url** lists image files from a specific web address. Select the protocol (**http** or **https**), and then enter a web address (for example, http://www.mysite.com/images/image.jpg).

> **Note:** If a URL doesn't have a file extension, and the option Save with template is **not** selected, the Select Image dialog automatically adds the `filetype` parameter with the file extension as its value (for example: `?filetype=pdf` (if it is the first parameter) or `&filetype=pdf`).
> The `filetype`, `page` and `nopreview` parameters are not sent to the host; they are used internally. Therefore, URLs that rely on one of these parameters cannot be used.

With an external image, you can check the option **Save with template**. If this option is checked, the file will be inserted in the **Images** folder on the **Resources** pane at the top left.
If it isn't saved with the template, the image remains external. Note that external images need to be available when the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced. External images are updated (retrieved) at the time the output is generated.

Insertion Point group box:

Selects where the image is to be inserted.

- **Absolute** checkbox: Select to make the inserted image an Absolutely positioned one.
  Selecting **Absolute** prevents subsequent selection of **Location** and **Element**.
- **Location** pull down option: Select the Location at which to insert the image.
- **Elements** pull down option: Select the **Element** that the **Location** option is to apply to.
  Not available for **Location** selection of *At cursor position*.

5. If the PDF file consists of more than one page, select the desired page.

   Note: The number of pages in a PDF file can not be determined via the HTTP and HTTPS protocols. If you wish to use a page other than page 1 in a remote PDF, check the option **Save with template**; then click OK and reopen the dialog. Next, on the Resources tab, select the PDF, and select a page.

6. Click **Finish**.
7. For each of the PDF files, select a **position**:
   - **Fit to page** stretches the PDF to fit the page size.
   - **Centered** centers the PDF on the page, vertically and horizontally.
   - **Absolute** places the PDF at a specific location on the page. Use the **Top** field to specify the distance between the top side of the page and the top side of the PDF, and the **Left** field to specify the distance between the left side of the page and the left side of the PDF.
8. Finally, click **OK**.

Note: Any non-content features in the original PDF will get stripped out of the original PDF. These include document navigation (outline), local links, hyperlinks, annotations (markup annotations, comments, etc.), digital signatures, multimedia features, interactive forms, and tagged PDF features (marks in the page content will stay, but the logical structure is gone).

Setting the paper's characteristics

To set a Media's paper characteristics:

1. On the **Resources** pane, expand the **Contexts** folder, expand the **Media** folder, and right-click the Media. Click **Characteristics**.
2. Specify the paper's characteristics:

- **Media Type**: The type of paper, such as Plain, Continuous, Envelope, Labels, Stationery, etc.
- **Weight**: The intended weight of the media in grammage (g/m2).
- **Front Coating**: The pre-process coating applied to the front surface of the media, such as Glossy, High Gloss, Matte, Satin, etc.
- **Back Coating**: The pre-process coating applied to the back surface of the media.
- **Texture**: The intended texture of the media, such as Antique, Calenared, Linen, Stipple or Vellum.
- **Grade**: The intended grade of the media, such as Gloss-coated paper, Uncoated white paper, etc.
- **Hole Name**: A predefined hole pattern that specifies the pre-punched holes in the media, such as R2-generic, R2m-MIB, R4i-US, etc.

3. Click **OK**.

Rename Media

To rename Media:

- On the **Resources** pane, expand the **Contexts** folder, expand the **Media** folder, right-click the Media and click **Rename**. Type the new name and click **OK**.

  > **Note:** Naming this element with a purely numeric name will cause issues with scripting. We recommend using a Alphanumeric name.

- Alternatively, on the **Resources** pane, expand the **Contexts** folder, expand the **Media** folder, right-click the Media and click **Properties**. Type the new name in the **Name** field and click **OK**.

Applying Media to a page in a Print section

Every page in a print section has a natural position: it can be the first page, the last page, one of the pages in between (a 'middle page'), or a single page. For each of those positions, you can set different Media.

To apply Media to specific page positions in a Print section:

1. On the **Resources** pane, expand the **Print** context; right-click the Print section, and click **Sheet configuration**.
2. Optionally, check **Duplex** to enable content to be printed on the back of each sheet. Your printer must support duplex for this option to work. If Duplex is enabled, you can

also check **Tumble** to duplex pages as in a calendar, and **Facing pages** to have the margins of the section switch alternately, so that pages are printed as if in a magazine or book.

3. If the option **Same for all positions** is checked, the same Media will be applied to every page in the print section. Uncheck this option.
4. Decide which Media should be linked to each sheet position: click the downward pointing arrow after **Media** and select a Media.
5. Optionally, decide which Master Page should be linked to each sheet; see "Master Pages" on page 648.

Note: When both Media and a Master Page are used on a certain page, they will both be displayed on the Preview tab of the workspace, the Master Page being 'in front' of the Media and the Print section on top. To open the Preview tab, click it at the bottom of the Workspace or select **View > Preview View** on the menu.

Note: Master Pages, Media and Duplex printing options can also be set in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). This is especially useful when you need identical sections with different settings.

Dynamically switching the Media

In addition to applying Media to sheets via the settings, it is possible to change Media dynamically, based on a value in a data field, in a script. The script has already been made; you only have to change the name of the Media and the section in the script, and write the condition on which the Media has to be replaced.

1. On the **Resources** pane, expand the **Contexts** folder, expand the **Print** context, right-click the print section and click **Sheet configuration**.
2. Decide which pages should have dynamically switching media: every first page in the Print section, every last page, one of the pages in between (a 'middle page'), or a single page. (Uncheck the option **Same for all positions**, to see all page positions.)
3. In the area for the respective sheet position, click the **Edit script** button next to **Media**. The Script Wizard appears with a Control Script:

```
var section = merge.template.contexts.PRINT.sections["Section 1"];
section.sheetConfig.positions.first.media = "Media 1";
```

"Media 1" will have been replaced with the name of the media selected for the chosen sheet position (see "Position" on page 1610).

4. Change the script so that on a certain condition, another media will be selected for the content. For instance:

```
if(record.fields.GENDER === 'M') {
    section.sheetConfig.positions.single.media = "Media 1";
}
```

This script changes the media to Media 2 for male customers.

See "Writing your own scripts" on page 1064 if you are not familiar with how scripts are written.

5. Click **Apply**, open the tab **Preview** and browse through the records to see if the script functions as expected.
6. When you click **OK**, the script will be added to the **Scripts** pane.

> **Note:** If you rename the script, the Sheet Configuration dialog will not be able to open it when the Edit Script button for the respective sheet position is clicked.

Rotating the Media in a Print section

The actual orientation of the Media and that of a section to which the Media is applied may not match.
The Media (to be more accurate: the Virtual Stationery images specified for this Media) can therefore be rotated per Print section:

- On the **Resources** pane, expand the **Print** context; right-click the Print section, and click **Sheet configuration**.
- Click one of the options next to **Media rotation**.

The Media will be rotated accordingly in the entire section.

> **Note:**
>
> - Any Virtual Stationery settings made for the Media also influence how the Media is displayed in each section (see "Setting Media properties" on page 653).
> - Section backgrounds are rotated separately (see "Using a PDF file or other image as background" on page 634).

If in the Media properties, the Virtual Stationery position is set to Absolute, any offset given by the Top and Left values will be applied **after** rotation. A Virtual Stationery image located absolutely at the top left (Top: 0, Left: 0) will still appear at the top left of the page after rotating the Media.

Printing virtual stationery

Media are not printed, unless you want them to. Printing the virtual stationery is one of the settings in a Job Preset. To have the virtual stationery printed as part of the Print output:

1. Create a Job Preset that indicates that Media has to be printed: select **File > New** and see "Job Preset" on page 1359 for more details.
2. Select that Job Preset in the Print Wizard; see "Generating Print output" on page 1639.

Email

With the Designer you can create one or more Email templates and merge the template with a data set to generate personalized emails.

The Email **context** is the folder in the Designer that can contain one or more Email templates, also called Email **sections**. The HTML generated by this context is meant to be compatible with as many clients and as many devices as possible.

Email template

It is strongly recommended to start creating an Email template with a Wizard; see "Creating an Email template with a Wizard" on page 664.
Also see "Designing an Email template" on the facing page for guidelines on the design.
Designing HTML email that displays properly on a variety of devices and screen sizes is challenging. Building an email is not like building for the web. While web browsers comply with standards (to a significant extent), email clients do not. Different email clients interpret the same HTML and CSS styles in totally different ways.

When an Email template is created, either with a Wizard or by adding an Email context to an existing template (see "Adding a context" on page 611), the Email context folder is created along with other files that are specific to an Email context; see "Email context" on page 667.

Only one Email section is created at the start, but you can add as many Email sections as you need; see "Email templates" on page 669. However, when the Designer merges a data set to generate output from the Email context, it can merge only one of the sections with each record; see "Generating Email output" on page 1673.

Email templates are personalized just like any other template; see "Personalizing content" on page 923.

Sending email

When the template is ready, you can change the email settings (see "Email header settings" on page 672) and send the email, directly from the Designer, or via Workflow (see "Email processes with OL Connect tasks" on page 305). To test a template, you can send a test email first.

Output generated from an Email template, can have the following attachments (and they will be attached in this order):

- Inline image attachments (in the order in which they are encountered in the email).
- The contents of the Print context, in the form of a single PDF attachment. (Compression options for PDF attachments can be specified in the Email context's properties; see "Compressing PDF attachments" on page 668.)
- Attachments specified in the head of the email; an image or a PDF leaflet for example. These can be dynamic. (See "Dynamic attachments: creating file names based on data fields" on page 680.)
- The output of the Web context, as a self-contained HTML file.

Attaching the Print context and/or the Web context is one of the options in the "Send (Test) Email" on page 1204 dialog.

These options are also available in the Create Email Content task in Workflow.

See "Email attachments" on page 679 and "Generating Email output" on page 1673.

Designing an Email template

With the Designer you can design Email templates. It is strongly recommended to start creating an Email template with an Email Template Wizard, because it is challenging to design HTML email that looks good on all email clients, devices and screen sizes that customers use when they are reading their email.

This topic explains why designing HTML email design is as challenging as it is, which solutions are used in the Email Template Wizards and it lists good practices, for example regarding the use of images in HTML email. It will help you to create the best possible Email templates in the Designer.

HTML email challenges

Creating HTML email isn't like designing for the Web. That's because email clients aren't like web browsers. Email clients pass HTML email through a preprocessor to remove anything that could be dangerous, introduce privacy concerns or cause the email client to behave unexpectedly. This includes removing javascript, object and embed tags, and unrecognized tags. Most preprocessors are overly restrictive and remove anything with the slightest potential to affect the layout of their email client. Next, the HTML has to be rendered so that it is safe to show within the email client. Unfortunately, desktop, webmail, and mobile clients all use different rendering engines, which support different subsets of HTML and CSS. More often than not, the result of these operations is that they completely break the HTML email's layout.

Designing HTML email in OL Connect Designer

The problem of HTML email is that preprocessing and rendering engines break the HTML email's layout. HTML tables, however, are mostly left untroubled. As they are supported by every major email client, they are pretty much the only way to design HTML emails that are universally supported. That's why Tables are heavily used to position text and images in HTML email.

Nesting tables (putting tables in table cells) and applying CSS styles to each table cell to make the email look good on all screen sizes is a precision work that can be a tedious and demanding. Connect's Designer offers the following tools to make designing HTML email easier.

Email templates: Slate and others

The most obvious solution offered in the Designer is to use one of the templates provided with the Designer; see "Creating an Email template with a Wizard" on page 664. The layout of these templates has been tested and proven to look good in any email client, on any device and screen size. The Tables in these templates are nested (put inside another table) and they have no visible borders, so readers won't notice them.

> **Tip:** Click the **Edges** button on the toolbar to make borders of elements visible on the Design tab. The borders will not be visible on the Preview tab or in the output.

Litmus

There are several tools to preview how email will be rendered on a variety of clients. We recommend using **Litmus**. Support for Litmus is integrated into the Designer; the Send Test Email dialog has an option to "Send to Litmus".

Emmet

Emmet is a plugin that enables the lightning-fast creation of HTML code though the use of a simple and effective shortcut language. The Emmet functionality is available in the HTML and CSS source editors of Connect Designer. Emmet transforms abbreviations for HTML elements and CSS properties to the respective source code. The expansion of abbreviations is invoked with the **Tab** key.

In the Source tab of the Workspace, you could for example type `div.row`. This is the abbreviation for a element with the class row.
On pressing the Tab key, this abbreviation is transformed to:

```
<div class="row"></div>
```

To quickly enter a table with the ID 'green', one row, and two cells in that row, type:

```
table#green>tr>td*2
```

On pressing the Tab key, this is transformed to:

```
<table id="green">
	<tr>
		<td></td>
		<td></td>
	</tr>
</table>
```

> **Tip:** For more examples, see this how-to: Streamline coding with emmet and custom abbreviations.

All standard abbreviations can be found in Emmet's documentation: Abbreviations.

To learn more about Emmet, please see their website: Emmet.io and the Emmet.io documentation: http://docs.emmet.io/.

Preferences

To change the way Emmet works in the Designer, select **Window > Preferences**, and in the Preferences dialog, select **Emmet**; see "Emmet preferences" on page 1049.

Using CSS files with HTML email

Email clients do not read CSS files and some even remove a <style> tag when it is present in the email's header. Nevertheless, CSS files can be used with the Email context in the Designer.
When generating output from the Email context, all CSS rules that apply to the content of the email can be processed and added either to the header of the email or to inline style

properties as if local formatting was applied, depending on the Email section properties.
See "Email section properties" on page 1199.

Using images in email campaigns: tips

Host images on a public server

In the Designer you can add images as resource to the template document. When used in email messages these images are automatically embedded on sending the email. These embedded images appear instantly when viewing the message in your email client.
There is, however, a downside to this method: embedded images can't be used to track email open rates. Email services like mandrillapp.com embed a tiny tracer image at the bottom of your message. Each time a recipient opens the email the tracer image (aka beacon image) is downloaded and yet another 'open' is registered. On mobile devices this happens when the user clicks the Display Images button.
So, when tracking open rates in your email campaigns, store your images on a publicly-accessible server (preferably your own server- you could set up a process in Workflow to serve images and track open rates) or a reputable image hosting service, like photobucket.com. Don't forget to set the Alternate Text for your images on the Attributes pane.

Do not capture your email in one big image

Most e-mail clients do not automatically download images, so do not capture your email in one big image. The recipient initially sees a blank message and probably deletes it right away.

Do not resize images in your email

Many mail clients do not support image resizing and will show the image in its original dimensions. Resize the images before you link to or embed them.

Use background images wisely

Most mail clients do not support background images: a very good reason to stay away from them in your mainstream email campaign. There is one situation in which they do come in handy. Both iPhone and Android default mail have solid CSS support and cover most of the mobile marketspace. You could use background images to substitute images when viewed on these devices. This is done by hiding the actual image and showing a mobile-friendly image as background image instead. This is a technique used in Responsive Email Design.

Creating an Email template with a Wizard

With the Designer you can design Email templates as well as PDF attachments. PDF attachments are designed in the Print context; see "Print context" on page 624.

It is strongly recommended to start creating an Email template with a Wizard, because designing HTML email that displays properly on a variety of devices and screen sizes is challenging. Email clients can, and will, interpret the same HTML and (inline) CSS in totally different ways (see "Designing an Email template" on page 660).

With an Email Template Wizard you can easily create an Email template that outputs emails that look good on virtually any email client, device and screen size.

After creating an Email template, the other contexts can be added to it, as well as other sections (see "Context" on page 610 and "Email templates" on page 669).

To create an Email template with a Template Wizard:

1. In the **Welcome** screen that appears after startup:

a. Choose **New** at the left, then **Email** at the right.

b. Select **Blank template**, **Basic message** (with a call-to-action button) or **Invoice** (with a detail table); or scroll down and select one of the Email templates from the **online resources**.

> **Tip:** Click the Home icon at the top right to reopen the Welcome screen.

Alternatively, on the menu select **File > New**, and expand the **Template** folder. Now you can select **Email Template**; this starts the Basic Action Email wizard. Or scroll further down and expand the **Basic Email templates** folder, the **Banded Email templates** folder, or the **Slate: Responsive Email Templates by Litmus** folder, and select one of those templates.

See "Email Template Wizards" on the next page for information about the various types of Template Wizards.

2. Make adjustments to the initial (or last used) settings. The options for each type of template are listed below.
   Click **Next** to go to the next settings page if there is one.
3. Click **Finish** to create the template.

The Wizard creates:

- An Email context with one section in it. The section contains dummy text and one or more **expressions** for variable data, for example: "Hello {{first}}". You will want to replace those by the names of fields in your data. See "Variable data in the text" on page 924.

  The Invoice email template also contains a Dynamic Table; see "Dynamic Table" on page 974.
  Note that this table does not use one of the default table styles, and that the style sheet with the default table styles is not present in the template. To add that style sheet to the template, insert a table using the Dynamic Table wizard.
- A style sheet, named context_htmlemail_styles.css, and another style sheet depending on which Template Wizard was used. The style sheets can be found in the **Stylesheets** folder on the **Resources** pane.

The Wizard opens the Email section, so that you can fill it with text and other elements; see "Content elements" on page 761, "Email context" on page 667, "Email templates" on page 669 and "Email header settings" on page 672.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Note that the contents of the email are arranged in tables. The many tables in an Email template ensure that the email looks good on virtually any email client, device and screen size. As the tables have no borders, they are initially invisible.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

Tip: To get started quickly, you can import various resources (master pages, media, sections, images, style sheets, scripts, and the data model) from an existing template. See "Import Resources dialog" on page 1155.

Email Template Wizards

There are Wizards for three kinds of Email templates: for **Basic Email**, for **Banded Email**, and **Slate** templates for responsive email designed by Litmus.

Slate: Responsive Email Templates by Litmus

Scroll past the Web Template Wizards to see the Slate: Responsive Email templates, created by Litmus (see https://litmus.com/resources/free-responsive-email-templates).

More than 50% of emails are opened on mobile. These five responsive HTML email templates are optimized for small screens and they look great in any inbox. They've been tested in Litmus and are completely bulletproof.

Tip: After creating the email template, click the Responsive Design View icon at the top of the workspace to see how the email looks on different screen sizes.

The only thing you can set in advance for a Slate template is the color of the call-to-action button. Click the small colored square, right next to the field that holds the default color value, to open the Color dialog and pick a color (see "Color Picker" on page 1143).
The color can be changed later; see "Colors" on page 913.

Basic Email and Banded Email

The difference between Basic and Banded email is that the contents of a Basic email extend to the email's margin, rather than to the edge of the window in which it is read, as the contents of Banded emails do.
The Banded Email **Action** Template is a simple call-to-action email with a message, header and a button linking to a website, such as an informational or landing page.
The Banded Email **Invoice** Template is an invoice with an optional Welcome message and Pay Now button.

Settings

For a **Blank** email you can not specify any settings in the Wizard.

For an **Action** or **Invoice** email, the Email Template Wizard lets you choose:

- The subject. You can change and personalize the subject later, see "Email header settings" on page 672.
- The text for the header. The header is the colored part at the top. The text can be edited later.
- The color of the header and the color of the button. Click the small colored square, right next to the field that holds the default color value, to open the Color dialog and pick a color (see "Color Picker" on page 1143). The color can be changed later; see "Colors" on page 913.

- A logo. Use the Browse button to select an image from disk, or enter a URL to use a remote image.
- The web address where the recipient of the email will be taken after clicking the button in the email. Type the URL in the **Link** field.

In addition, for an **Invoice** email you can change the following content settings:

- **Show Welcome Message**. Check this option to insert a salutation and one paragraph with dummy text in the email.
- **Detail Table Name**. Type the name of a detail table to fill the lines of the invoice with data. In the Designer, a detail table is a field in the Data Model that contains a variable number of items (usually transactional data).

Email context

In the Designer the Email context is the folder that contains Email templates. From the Email context, output can be generated in the form of email (see below).

When an Email template is created (see "Creating an Email template with a Wizard" on page 664) or when an Email context is added to a template (see "Adding a context" on page 611) the following happens:

- The Email context is created and one **Email section** is added to it. You can see this on the **Resources** pane: expand the **Contexts** folder, and then expand the **Email** folder. See "Email templates" on page 669 to learn how to fill an Email section.
  Although only one email can be sent per record when generating Email output, the Email context can contain multiple sections. One Email section is created at the start, but you can add more; see "Adding an Email template" on page 670 and "Importing an Email template" on page 670.
- A style sheet, named context_htmlemail_styles.css, is added to the template. Depending on which Template Wizard was used to create the template, another style sheet can be added as well. Style sheets are located in the folder **Stylesheets** on the **Resources** pane. These style sheets are meant to be used for styles that are only applied to elements in the Email context.

The Wizard opens the Email section, so that you can fill it with text and other elements; see "Content elements" on page 761 and "Email templates" on page 669.

Sending email

When the template is ready, you can generate Email output; See "Generating Email output" on page 1673.
To test a template, you can send a test email first. This allows you to override the recipient address. Test email is only sent to the address specified in the Send Test Email dialog.

Output generated from an Email template, can have the following attachments (and they will be attached in this order):

- Inline image attachments (in the order in which they are encountered in the email).
- The contents of the Print context, in the form of a single PDF attachment. (Compression options for PDF attachments can be specified in the Email context's properties; see "Compressing PDF attachments" below.)
- Attachments specified in the head of the email; an image or a PDF leaflet for example. These can be dynamic. (See "Dynamic attachments: creating file names based on data fields" on page 680.)
- The output of the Web context, as a self-contained HTML file.

Attaching the Print context and/or the Web context is one of the options in the "Send (Test) Email" on page 1204 dialog.

These options are also available in the Create Email Content task in Workflow.

> **Note:** To split the Print context into multiple attachments, or to attach multiple Web sections as separate attachments, and to rename this type of attachment, you need to create a Control Script that specifies **parts**; see "Parts: splitting and renaming email attachments" on page 1104.

See "Email attachments" on page 679.

Email output settings

The following settings in an Email context influence how the Email output is generated.

Compressing PDF attachments

For PDF attachments, generated from the Print context, you can set the Print Context Image Compression to determine the quality of the files, and with that, the size of the files.

To set the Print Context Image Compression:

1. On the **Resources** pane, expand the **Contexts** folder; then right-click the **Email** context and select **PDF Attachments**.
   Alternatively, select **Context > PDF Attachments** on the main menu. This option is only available when editing an Email section in the Workspace.
2. Change the properties of the PDF file that will be attached when the Print context is attached to the email.

   Lossless is the maximum quality. Note that this will produce a larger PDF file. Uncheck this option to be able to set a lower quality.

   The **quality** is set in a percentage of the maximum quality.

   Tile Size is the size of the files in which the image that is being compressed is divided. (If the image height or width is not an even multiple of the tile size, partial tiles are used on the edges.) Image data for each tile is individually compressed and can be individually decompressed. When low Quality values are used to optimize images smaller than 1024 x 1024 pixels, using the largest tile size will produce better results.

Setting a default section for output

When generating output from the Email context, only one of the Email templates can be merged with each record. One of the Email sections is the 'default'; see "Setting a default Email template for output" on page 672.

Email templates

Email templates (also called Email **sections**) are part of the Email context in a template. The Email context outputs HTML email with embedded formatting to an email client through the use of an email server. Since email clients are numerous and do not support same features, the HTML generated by this context is not optimized for any specific client - rather, it's meant to be compatible with as many clients and as many devices as possible.

In Email templates, many **content** elements can be used; see "Content elements" on page 761. However, special attention must be paid to the way elements are positioned. In Email sections, it is advisable to position elements using Tables and to put text in table cells (see "Designing an Email template" on page 660).

Email templates are **personalized** just like any other template; see "Personalizing content" on page 923.

The **subject**, **recipients** (To, CC and BCC), **sender** and **reply-to** address are specified with either Handlebars expressions or Email Script Wizards; see "Email header settings" on page 672.

An Email context can contain multiple templates. When generating **output** from the Email context, however, only one of the Email templates can be merged with each record. Set the 'default' Email section (see below) before generating Email output; see also "Generating Email output" on page 1673.

For information about **attachments** see "Email attachments" on page 679.
A plain-text version of the HTML is added to each email if the option is checked in the Email section's properties (see "Properties tab" on page 1199). With new templates this is always the case.

Adding an Email template

When an Email template is created (see "Creating an Email template with a Wizard" on page 664), only one Email section is added to it. An Email context may contain various templates, but per record only one of those can be sent when you generate Email output.

It is not possible to add an Email section to an existing Email context with the help of a Template Wizard.

To provide alternative content for your email, you could use Conditional Content (see "Showing content conditionally" on page 958), or Snippets and a script (see "Snippets" on page 867 and "Loading a snippet via a script" on page 1091).

If you would like to start with a template that is identical to the one you already have, consider copying it (see "Copying a section" on page 613). If it's inside another template you can import it (see below).

To add a section to the Email context:

- On the **Resources** pane, expand the **Contexts** folder, right-click the **Email** folder, and then click **New Email**.
- Or, right-click an Email section already in the Resources and select **Duplicate**.

Importing an Email template

To import an Email section from another template, click **File > Import Resources...** in the menu. See: "Import Resources dialog" on page 1155.
Remember also to add or import any related source files, such as images.

Note that when the imported Email section *replaces* an Email section in your template, the PDF attachments settings are imported as well. (See: "Compressing PDF attachments" on page 668.)

Deleting an Email template

To delete an Email section:

- On the **Resources** pane, expand the **Contexts** folder, expand the **Email** context, right-click the name of the section, and then click **Delete**.

Caution: If you don't have a backup of the template, the only way to recover a deleted section, is to click **Undo** on the **Edit** menu, until the deleted section is restored. After closing and reopening the template it is no longer possible to restore the deleted context this way.
In the Saving Preferences you can set whether a backup file should be created when you save the template; see "Save preferences" on page 1059.

Styling and formatting an Email template

The contents of an Email section can be formatted directly, or styled with Cascading Style Sheets (CSS). See "Styling and formatting" on page 879.

Email clients do not read CSS files and some even remove a <style> tag when it is present in the email's header. Nevertheless, CSS files can be used with the Email context in the Designer.
When generating output from the Email context, all CSS rules that apply to the content of the email can be processed and added either to the header of the email or to inline style properties as if local formatting was applied, depending on the Email section properties. See "Email section properties" on page 1199.

Tip: Before you can style an element, you have to select it. In an Email context it can be difficult to select an element by clicking on it. Use the **breadcrumbs** at the top and the **Outline** pane at the left, to select an element. See "Selecting an element" on page 766.

Including a style sheet in a section

In order for a style sheet to be applied to a specific section, it needs to be included in that section. There are two ways to do this.

Drag & drop a style sheet

1. Click and hold the mouse button on the style sheet on the **Resources** pane.
2. Move the mouse cursor within the **Resources** pane to the section to which the style sheet should be applied.
3. Release the mouse button.

Using the Includes dialog

1. On the Resources pane, right-click the section, then click **Includes**.
2. From the **File types** dropdown, select **Stylesheets**.
3. Choose which CSS files should be applied to this section. The available files are listed at the left. Use the arrow buttons to move the files that should be included to the list at the right.
4. You can also change the order in which the CSS files are read: click one of the included CSS files and use the **Up** and **Down** buttons. Note that moving a style sheet up in the list gives it **less** weight. In case of conflicting rules, style sheets read later will override previous ones.

Note: Style sheets are applied in the order in which they are included in a section. The styles in each following style sheet add up to the styles found in previously read style sheets. When style sheets have a conflicting rule for the same element, class or ID, the **last** style sheet 'wins' and overrides the rule found in the previous style sheet.

Note: Style sheets that are linked to (i.e. included in) a section show a chain icon in the Resources pane (see "Resources pane" on page 1250).

Setting a default Email template for output

An Email context can contain multiple templates. When generating output from the Email context, however, only one of the Email templates can be merged with each record.

To select the Email section that will be output by default:

- On the **Resources** pane, expand the **Email** context, right-click a section and click **Set as Default**.

Tip: Use a Control Script to dynamically select an Email section for output depending on the value of a data field.

Email header settings

An email header contains routing information, such as the **sender**, **recipient**/s and **subject** of the message. This topic explains how to customize the header of an email that is generated from an Email template.

The default Email SMTP settings and the sender's name and address are defined in the Connect Designer preferences (see "Email preferences" on page 1048). They can be adjusted per run in the Send Email and Send Test Email dialogs.

Email Fields

The **subject**, the **recipients** (To, Cc and Bcc), the **sender** and the **Reply to** address can be entered in the **Email Fields** at the top of the workspace. If the fields are not visible, click the words 'Email Fields' (or the small plus before them) to expand the Email Fields area.
To use a variable email address in any of the fields, simply **drag and drop** a data field into the email field. This adds an **expression** to the field. For example, if the data field name is *customer_email*, then the expression is *{{customer_email}}*.

If the section is not set to evaluate Handlebars expressions, drag-and-drop adds a placeholder and a **script** that replaces the placeholder.
To make the software insert expressions instead of placeholders, right-click the section in the **Resources** pane, select **Properties** and check the option **Evaluate Handlebars expressions**.

For more information about Handlebars expressions, see "Variable data in text: expressions" on page 999 and "Handlebars in OL Connect" on page 998.

The specified subject and addresses will be visible when viewing the email in the workspace in Preview mode.
The To address must always be variable. This field is not used when you send a test email (see "Generating Email output" on page 1673).

> **Note:** Using a variable email address requires you to load data or a data mapping configuration first; see "Loading data" on page 926.

The Email Script Wizard

If you want to use scripts to add data to an email header field, you can use the Email Script Wizard. It lets you choose one or more data fields and enter a prefix and/or suffix (per data field).
There are two ways to open the Email Script Wizard:

- Via the **Email Fields**. Open the email section and expand the Email Fields at the top by clicking **Email Fields**. Click the word before the email field that you want to set. If there already is a script for that field, that script will be opened. Otherwise, a new script will be created and opened.
- Via the **Scripts** pane. Click the black triangle on the **New** button and select the respective email script. A new script will be added to the Scripts pane. Double-click the new script to open it. See "Script wizards" on page 1192 for an explanation of the options in the script wizard.

The default script adds the content of the selected data field to the header field.
If you want to write a more complex script, click the **Expand** button. The result of the script should be a valid, fully-formed email address.

The language in which the script has to be written is JavaScript. For more information on writing scripts, see "Writing your own scripts" on page 1064.

Other header fields

At some point you may need to define a header field that isn't available in the Preferences or in the Email Fields. This can be done in a Control Script. For a few examples of such scripts, see "Adding custom ESP handling instructions" on page 1679. To get started with Control Scripts, refer to "Control Scripts" on page 1098.

Email SMTP settings

Simple Mail Transfer Protocol (SMTP) is the standard protocol for sending emails across the Internet.

Default SMTP settings can be specified in the Preferences dialog: select **Window > Preferences**, expand the **Email** preferences and click **SMTP**.

You can add as many presets as needed, for example for different Email Service Providers (see "Using an ESP with OL Connect" on page 1678). To do this, click the Add button at the right. Then fill out the following settings:

- **Name**: The name of the preset. This will show up in the Send Email dialog.
- **Host**: The SMTP server through which the emails are to be sent. This can be a host (mail.domain.com) or an IP address.

  Tip: Gmail only allows Connect to be used as an SMTP client when "Access for less secure apps" is enabled in the Google account settings.

- **Port**: You can specify a port number. This will be added to the host name, for example: smtp.mandrillapp.com:465.

  Tip: If the mail server supports it, the connection will be encrypted without the need to send the server a STARTTLS instruction when port 465 is used.

- **Use authentication**: Check this option and fill in the user name if a user name and password are needed to send emails through the host. (The password has to be specified in the Send Email or Send Test Email dialog.)
- **Send STARTTLS**: This option is enabled if authentication is checked. With STARTTLS the client negotiates with the mail server to use some form of encryption, usually a

version of Transport Layer Security (TLS). Since this improves security it is recommend to enable this option if you use port 25 (the default port), 2525, or 587.
Note that the email will not be sent if the SMTP server does not support TLS or SSL (an older encryption type).
This option is ignored when port 465 is used.

When you click the **Restore** button, the presets for a number of Email Service Providers will appear.

Note: When updating the software from a version prior to version 1.5, pre-existing presets will be maintained in the new version.

In the Send Test Email dialog and Send Email dialog (see "Send (Test) Email" on page 1204) you will be able to choose one of the presets and adjust the settings to your needs.

Subject

To specify a subject for an email template:

1. Open the email section and expand the Email Fields by clicking **Email Fields** at the top of the section.
2. Type the subject in the **Subject** field.

To add **variable data** to the subject of an email section, **drag and drop** a data field into the Subject field at the top of the workspace. Two things will happen:

- A placeholder for the data field appears in the subject line (for example: @email@).
- A new script, named Subject, is added to the Scripts pane.

You can add as many data fields to the subject as you like. When you do add more than one data field, the existing Subject script will be modified to include all data fields that are added to the subject.

The result of the script will be visible in the Subject field in Preview mode: click the Preview tab at the bottom of the workspace.

Note: By default, the Subject script targets one email section specifically. You can see this when you double-click the script on the Scripts pane. The selector of the Subject script contains the name of a particular email section, for example: html.HTML_EMAIL [section="Content"] (in this case, Content is the name of the email section). If you remove the [section=...] part from the selector, the script will work for all email sections.

Subject scripts made with versions of the software prior to version 1.7 are **not** specific to one email section.

Writing a custom Subject script

The default script replaces all @field@ placeholders in the subject line with field values. This script can be modified, for example to create a subject that depends on the value of a data field. Open the Script Wizard (see "The Email Script Wizard" on page 673), click the **Expand** button and modify the script.
If you don't know how to write a script, see "Writing your own scripts" on page 1064 first.

Note: A Subject script created by clicking Subject in the Email Fields always targets one email section specifically, for example: html[section="Content"] (in this case, Content is the name of the email section). Remove the `[section=...]` part from the **selector** of the script to make the script work for all email sections.

Recipients: To, CC and BCC

To specify recipients for Email output, you can simply **drag and drop** a data field that contains an email address into the **To** field at the top of the workspace. A new script, named To, will be added to the Scripts pane.
Note that you can add only one data field to the email field this way. When you drag another data field into the email field the existing script will be replaced.

Alternatively, you could use the Script Wizard to create the scripts; see "The Email Script Wizard" on page 673.

Email addresses can be added to the **Cc** and **Bcc** fields in the same manner, but it is also possible to **type** an email address directly in the Cc or Bcc field (as long as no script is present for that field). Email addresses in the Bcc ('blind carbon copy') field will not be visible to any other recipient of the email.

For each recipient, a name and email address can be specified. In that case the email address itself must be enclosed in <>, for example: Tester <tester@example.com>. The name can be enclosed in quotes: "Tester" <tester@example.com>. If the recipient's name has a comma, it should always be enclosed in quotes: "Tester, QA" <tester@example.com>.

Separate **multiple** email addresses using a semicolon or a comma.

Sender

From address

A default **From** name and email address can be specified in the Preferences dialog: select **Window > Preferences**, expand the **Email** preferences and click **General**.
This name and email address will appear as the default in the Send Email and Send Test Email dialogs (see "Send (Test) Email" on page 1204).

The default can be overwritten by typing an email address directly in the From field (as long as no script is present for this field).

Using the Script Wizard you can create a **dynamic** From address; see "The Email Script Wizard" on page 673. It is also possible to drag and drop one data field into the From field directly.

Tip: A dynamic From address is often used when sending email campaigns and to do tracking of email replies. Include the recipient's email address in a dynamic From address to enable automatic detection and removal of undeliverable e-mail addresses. (This technique is called VERP; see Wikipedia.)

Note: When sending emails with a variable From address through OL Connect Workflow, check the option **Precedence to template address** in the Create Email Content task properties to make sure that the dynamic address gets precedence over the email address specified in the task properties (see Create Email Content task).

Reply To address

The Reply To address is used by mail clients, when the recipient clicks the Reply To (or Reply All) button.

You can type an email address directly in the Reply To field (as long as no script is present for this field).
Alternatively, you can drag and drop one data field into the field, or use the Email Script Wizard (see "The Email Script Wizard" on page 673), to specify the Reply To address in a script.

Meta information

Meta information in an email is not visible to the receiver, but might have an effect on the representation of the email in the e-mail client.

To add custom meta information to an email:

1. Right-click the email section on the Resources pane and select **Properties...**.
2. The **Meta Information** group lists all <meta> tags that will be added to the email. Click the **Add** button to add a new <meta> tag to the list.
3. Select the **type** of <meta> tag, which is either name or http-equiv.
4. For a name-type meta tag, enter the **value**.
5. Enter the **content**.

> **Example:** When you add a *name* meta tag with the value *viewport* and content *width=320*, the following will be added to the email:
> <meta name="viewport" content="width=320">.

For more information on <meta> tags, see MDN Web Docs - meta.

Email PDF password

The Email PDF Password Script Wizard defines a password with which to protect the PDF generated when using the Print context as PDF Attachment option in the Send Email or Send Test Email dialogs (see "Generating Email output" on page 1673). The result of the script will be the password necessary to open the PDF when it is received by email.

To define a password to protect the generated PDF attachment:

1. On the Scripts pane, click the black triangle on the **New** button and click **Email PDF password Script**. A new script is added to the Scripts pane.
2. Double-click the new script to open it.
3. Select a data field and optionally, type a prefix and/or suffix.

Password types

PDF allows for two types of passwords to be set on a secured PDF file: a user password and owner password. The user password allows a limited access to the file (e.g. printing or copying text from the PDF is not allowed). The owner password allows normal access to the file. The Email PDF password script sets both the user and owner password to the same value, so that when the recipient provides the password, he can manipulate the file without limitations.

> **Note:** If a template has a Control Script that creates multiple PDF attachments, all the attachments are secured by the same password.

Note: Via a Control Script it is possible to set a different user password and owner password, see "Control Script: Securing PDF attachments" on page 1112, "Control Scripts" on page 1098 and "Control Script API" on page 1591.

Email attachments

Output generated from an Email template, can have the following attachments (and they will be attached in this order):

- Inline image attachments (in the order in which they are encountered in the email).
- The contents of the Print context, in the form of a single PDF attachment. (Compression options for PDF attachments can be specified in the Email context's properties; see "Compressing PDF attachments" on page 668.)
- Attachments specified in the head of the email; an image or a PDF leaflet for example. These can be dynamic. (See "Dynamic attachments: creating file names based on data fields" on the facing page.)
- The output of the Web context, as a self-contained HTML file.

Attaching the Print context and/or the Web context is one of the options in the "Send (Test) Email" on page 1204 dialog.

These options are also available in the Create Email Content task in Workflow.

Caution: If an attachment cannot be found, the email will not be sent.

Attaching a Context

By default, when adding the Print context to an email, all Print sections are output to a single PDF file, named after the email subject, which is then attached to the email.
The PDF can be protected with a password (see "Email PDF password" on the previous page).
Compression options for PDF attachments can be specified in the Email Context Properties dialog; see "Compressing PDF attachments" on page 668.

When adding the Web context to an email, only the default Web section is generated and added to the email as an HTML file that is named after the email subject.

Note: To split the Print context into multiple attachments, or to attach multiple Web sections as separate attachments, and to rename this type of attachment, you need

to create a Control Script that specifies **parts**; see "Parts: splitting and renaming email attachments" on page 1104.

Note: A plain-text version of the HTML is added to each email if the option is checked in the Email section's properties (see "Properties tab" on page 1199). With new templates this is always the case.

Attaching files

Selecting and adding files as attachments

If you want all recipients to get the same attachments with their email, you can add the attachments to the Email section(s).

The easiest way is to **drag and drop** the desired file on the Email section.
If the file is an image, you will be presented with the option to import it into the template's Resources folder. Any other file will be added to the list of attachments directly.

The **Attachments** dialog also lets you select files and delete attachments.
To open the Attachments dialog, **right-click** the Email section in the Resources pane and select **Attachments**.
Alternatively, select **Section > Attachments** from the menu. This menu item is only available when an Email section is opened in the workspace.

For further information about this dialog, see: "Attachments dialog" on page 1129.

Dynamic attachments: creating file names based on data fields

The **Dynamic Attachment Script Wizard** lets you add a different attachment for each email recipient. It composes **one** file name (including the path) based on the value of one or more data fields.

The Dynamic Attachment wizard lets you concatenate this value with the base location and/or file extension to construct the path. Dynamic Attachment scripts are created via the New option on the toolbar of the Scripts panel.

1. On the **Scripts** pane at the bottom left, click the downward pointing arrow next to the **New** button; then select **Email Scripts > Dynamic Attachment**.

   Note: An Attachments script creates **one single attachment**. To add more attachments, you could either add Attachments scripts, or click the Expand button and edit the script.

2. A new script called **Attachments** has appeared in the list. Double-click to open it.

3. Choose an Email section from the **Section** drop-down list.
4. Fill in the different parts of which the file name is composed:
    - **Prefix**. The first prefix contains the base path (or at least the first, static part of the path). For example:`C:\Attachments\`, `C:/Attachments/`, or `file:///C:/Attachments/`.
    - **Data field/s**. The selected data field/s will be evaluated. If a data field is empty, the entire row is skipped. Otherwise the prefix, data field value and suffix are added to the path/file name.
    - **Suffix**. The suffix on the last used row should contain the file extension, including the dot (for example `.pdf`).

    For resources **inside** the template, refer to the folder in the Resources, e.g. 'images/file.extension' , or 'fonts/myfont.otf', etc.
    For any other file, give a valid **URL**.
    - Use the file protocol for a file on **disk**, for example: file:///c:/somefolder/attachments/INV2018.246.pdf (which equals file://localhost/c:/somefolder/attachments/INV2018.246.pdf; if the host is "localhost", it can be omitted).
    - For a **remote** file, you can use the http protocol e.g. http://www.mysite.com/somefolder/attachments/INV2019-246.pdf or http://localhost:8080/pod/v1/deliverynotes/{8FCEC8BC-72E8-486B-A206-516BF10E21F6}.

    Note: For attachment names, it is recommended to use only US-ASCII characters. Other characters may not be supported by all email servers and clients.

    Note: Certain characters are invalid in a URL (for example, '$', '%', and '&') and must be percent-encoded. The same applies to a file path, since that actually is a URL that starts with the file protocol.

    Note that even a space character is invalid in a URL. Spaces in a URL are supported for backward compatibility, but it is recommended to percent-encode a space character as %20.
5. The attachment's name in the email will be the part of the path that comes after the last '/'. When there are no forward slashes in the path, the full path is used.
    You may want to use a custom attachment name. To learn how to do that, see

"Renaming attachments" on the facing page.

6. Click **OK** or **Apply** to save your changes.

Renaming attachments

Sections that are attached to an email can only be renamed via a Control Script; see "Parts: splitting and renaming email attachments" on page 1104.

Renaming dynamic attachments

Dynamic attachments can be renamed via the script that attaches them to the email. Double-click the script to open it and click the **Expand** button.

Dynamic attachment scripts add a <link> element to the <head> of an Email section. The title attribute of that element specifies the attachment name that will show up in the email. Take a look at the last line of the script:
results.append(query("<link rel=related>").**attr("title", result.split('/').pop())**.attr("href", result));
To give the attachment another name, you have to replace the bold part of the code by that new name. For example:
results.append(query("<link rel=related>").attr("title", **"Invoice.pdf")**.attr("href", result));
Of course, you can also use data field values here, for example: results.append(query("<link rel=related>").attr("title", **record.fields.invoice_number + ".pdf")**.attr("href", result));

Note that the Wizard can no longer be used once you have edited and saved the script.

> **Note:** For attachment names, it is recommended to use only US-ASCII characters. Other characters may not be supported by all email servers and clients.

Web

With the Designer you can create one or more Web templates and merge the template with a data set to generate personal web pages.

The **Web** context is the Web output channel and the folder in the Designer that can contain one or more Web templates. Capture OnTheGo templates are Web templates designed for the Capture OnTheGo app (see "Capture OnTheGo" on page 707). They are stored in the Web folder as well.

The Web context outputs one HTML web page. In addition to the HTML text it contains either the resources or references to the resources necessary to display it.
JavaScript files are added to the <head> in the generated HTML file. JavaScript toolboxes like jQuery and its plugins, or MooTools, are useful when you want to implement special features in the web page. (See "Using JavaScript" on page 703)

Style sheets are also added to the <head> and are used just as they would be used in a regular web page. (Also see: "Styling templates with CSS files" on page 881)

It is advisable to follow design guidelines for web pages, so that they are likely to look good in different browsers and on different devices and screen sizes. When you start with a Foundation or COTG Web Template Wizard, the Foundation framework is added to the template, to guarantee just that; see "Creating a Web template with a Wizard" below and "Capture OnTheGo template wizards" on page 717.

> **Tip:** You can use the Responsive Design View drop-down at the top of the workspace to see how your web page looks and behaves with different screen sizes.

When a Web template is created, either with a Wizard or by adding the Web context to an existing template (see "Adding a context" on page 611), the Web context folder is created along with other files that are specific to a Web context; see "Web Context" on page 687.

Many of the content elements that are available for all three contexts are particularly suitable for web pages; see "Content elements" on page 761. Web templates are personalized just like any other template; see "Personalizing content" on page 923.

You can add as many Web sections as you need; see "Web pages" on page 688. However, when output is created from the Web context, only one of the Web sections can be merged with a record; see "Generating Web output" on page 1682.

Creating a Web template with a Wizard

With the Designer you can design Web templates and output them through Workflow or as an attachment to an email when generating Email output.

Capture On The Go templates are a special kind of Web templates; see "Capture OnTheGo template wizards" on page 717.

A Web Template Wizard helps you create a Web page that looks good on virtually any browser, device and screen size.

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop computers, tablets, and mobile phones. Foundation is tested across many browsers and

devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

For more information about the use of Foundation in the Designer, see "Using Foundation" on page 721.

After creating a Web template, the other contexts can be added, as well as other sections (see "Adding a context" on page 611 and "Adding a Web page" on page 689).

To create a Web template with a Template Wizard:

1. In the **Welcome** screen that appears after startup:

a. Choose **New** at the left, then **Web** at the right.

b. Select **Blank template** or scroll down and select one of the Web templates from the **online resources**.

> **Tip:** Click the Home icon at the top right to reopen the Welcome screen.

Alternatively, on the menu select **File > New**, and expand the **Template** folder. Now you can select **Web Page Template**.
Or scroll further down, expand the **Foundation Web Page Starter** folder and select one of those templates.

There are 4 types of Foundation Web Template Wizards:

- Blank
- Contact Us
- Jumbotron
- Thank You

If you don't know what template to choose, see "Web Template Wizards" on page 686 further down in this topic, where the characteristics of each kind of template are described.

2. Click **Next** and make adjustments to the settings. The wizard remembers the settings that were last used for a Foundation Web template.

- **Section**:
  - **Name**: Enter the name of the Section in the Web context. This has no effect on output.

- **Description**: Enter the description of the page. This is the contents of a <meta name="description"> HTML tag.

- **Top bar** group:
  - **Set width to Grid**: Check this option to limit the width of the top bar contents to the Foundation Grid, instead of using the full width of the page.
  - **Stick to the top of the browser window**: Check to lock the top menu bar to the top of the page, even if the page has scroll bars. This means the menu bar will always be visible in the browser.
  - **Background color**: Enter a valid hexadecimal color code for the page background color (see w3school's color picker), or click the colored circle to the right to open the Color Picker.
- **Colors** group: Enter a valid hexadecimal color code (see w3school's color picker) or click the colored square to open the Color Picker dialog (see "Color Picker" on page 1143), and pick a color for the following elements:
  - **Primary**: links on the page.
  - **Secondary**: secondary links on the page.
  - **Text**: text on the page contained in paragraphs (<p>).
  - **Headings**: all headings (<h1> through <h6>) including the heading section's subhead.

3. Click **Finish** to create the template.

The Wizard creates:

- A Web context with one web page template (also called a **section**) in it. The web page contains a Header, a Section and a Footer element with dummy text, and depending on the type of web page, a navigation bar, button and/or Form elements.
- Resources related to the Foundation framework (see "Web Template Wizards" on the facing page): style sheets and JavaScript files. The style sheets can be found in the **Stylesheets** folder on the **Resources** pane. The JavaScript files are located in the **JavaScript** folder on the **Resources** pane, in a **Foundation** folder.
- A collection of Snippets in the **Snippets** folder on the Resources pane. The Snippets contain ready-to-use parts to build the web page. Double-click to open them. See "Snippets" on page 867 for information about using Snippets.
- Images: icons, one picture and one thumbnail picture. Hover your mouse over the names of the images in the **Images** folder on the **Resources** pane to get a preview.

The Wizard opens the Web section, so that you can fill it with text and other elements; see "Content elements" on page 761, "Web Context" on page 687 and "Web pages" on page 688.

Web pages can be personalized just like any other type of template; see "Personalizing content" on page 923.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

Tip: To get started quickly, you can import various resources (master pages, media, sections, images, style sheets, scripts, and the data model) from an existing template. See "Import Resources dialog" on page 1155.

Web Template Wizards

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop computers, tablets, and mobile phones. Foundation is tested across many browsers and devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

Jumbotron

The name of the Jumbotron template is derived from the large screens in sports stadiums. It is most useful for informative or marketing-based websites. Its large banner at the top can display important text and its "call to action" button invites a visitor to click on to more information or an order form.

Contact Us

The Contact Us template is a contact form that can be used on a website to receive user feedback or requests. It's great to use in conjunction with the Thank You template, which can recap the form information and thank the user for feedback.

Thank You

The Thank You template displays a thank you message with some text and media links.

Blank web page

The Blank Web Page template is a very simple Foundation template that contains a top bar menu and some basic contents to get you started.

Web Context

In the Designer the Web context is the folder that contains Web page templates.

Creating the Web context

You can start creating a Web template with a Wizard (see "Creating a Web template with a Wizard" on page 683), or add the Web context to an existing template (see "Adding a context" on page 611).

When a Web template is created the following happens:

- The Web context is created and one Web page or **section** is added to it. You can see this on the **Resources** pane: expand the **Contexts** folder, and then expand the **Web** folder.
  See "Web pages" on the facing page to learn how to fill a web page template in the Designer.
  Although only one web page can be generated per record when generating Web output, the Web context can contain multiple sections. One section is created at the start, but you can add more; see "Adding a Web page" on page 689 and "Importing a Web page" on page 689.
- A style sheet, named context_web_styles.css, is added to the template. If a Template Wizard was used to create the template, Foundation style sheets are added as well. Style sheets are located in the folder **Stylesheets** on the **Resources** pane. These style sheets are meant to be used for styles that are only applied to elements in the Web context; see "Styling and formatting" on page 879.

Generating Web output

When the template is ready, you can:

- Output the web page as an as an integral HTML file attached to an Email context in the same template.
- Output the Web context in an automated Workflow using the Create Web Content task (see Workflow Help: Create Web Content).

See "Generating Web output" on page 1682.

Includes

The Web context outputs one HTML web page. In addition to the HTML text it contains either the resources or references to the resources necessary to display it.
JavaScript files are added to the <head> in the generated HTML file. JavaScript toolboxes like jQuery and its plugins, or MooTools, are useful when you want to implement special features in the web page. (See "Using JavaScript" on page 703)
Style sheets are also added to the <head> and are used just as they would be used in a regular web page. (Also see: "Styling templates with CSS files" on page 881)

Which style sheets and JavaScript files are included, and in which order, can be decided for the **Web context** as a whole, as well as per Web section.

To make this setting for the Web context as a whole, right-click the context on the **Resources** pane and select **Includes**. Alternatively, select **Context > Includes** on the main menu. (Note that this option is only available when editing a Web section in the Workspace.)
For further explanation see: "Includes dialog" on page 1157.
To make this setting for a particular Web section, right-click the section on the **Resources** pane and select **Includes**.

Web pages

Web pages (also called Web **sections**) are part of the Web context (see "Web Context" on the previous page) in a template.

The Web context outputs one HTML web page. In addition to the HTML text it contains either the resources or references to the resources necessary to display it.
JavaScript files are added to the <head> in the generated HTML file. JavaScript toolboxes like jQuery and its plugins, or MooTools, are useful when you want to implement special features in the web page. (See "Using JavaScript" on page 703)
Style sheets are also added to the <head> and are used just as they would be used in a regular web page. (Also see: "Styling templates with CSS files" on page 881)

A Web context can contain multiple templates. When generating output from the Web context, however, only one of the Web templates can be merged with each record. Set the

'default' Web section (see "Setting a default Web page for output" on page 691) before generating Web output; also see "Generating Web output" on page 1682.

Creating a Web page

When creating a Web page, it is advisable to follow design guidelines for web pages, so that they are likely to look good in different browsers and on different devices and screen sizes. When you start with a Web Template Wizard, the Foundation framework is added to the template, to guarantee just that; see "Creating a Web template with a Wizard" on page 683. Other approaches are described below, in "Adding a Web page" below.

Adding a Web page

When a Web template is created (see "Creating a Web template with a Wizard" on page 683), only one Web section is added to it. A Web context may contain various templates, but per record only one of those can be used to generate output.

It is not possible to add a Web section to an existing Web context with the help of a Template Wizard.

To provide alternative content for the web page, you could use Conditional Content (see "Showing content conditionally" on page 958), or Snippets and a script (see the Help topics "Snippets" on page 867 and "Loading a snippet via a script" on page 1091.)

If you would like to start with a Web template that is identical to one you already have, right-click that section and select **Duplicate**. Via the Using the Source

To add a blank section to the Web context:

- On the **Resources** pane, expand the **Contexts** folder, right-click the **Web** folder, and then click **New Web page**.

Importing a Web page

To import a Web page from another template, click **File > Import Resources...** in the menu.
See: "Import Resources dialog" on page 1155.
The Web page's Includes settings get imported automatically (see "Includes" on the previous page).
Remember to add or import any related source files, such as style sheets and images.

Deleting a Web page

To delete a Web section:

- On the **Resources** pane, expand the **Contexts** folder, expand the **Web** context, right-click the name of the section, and then click **Delete**.

Caution: If you don't have a backup of the template, the only way to recover a deleted section, is to click **Undo** on the **Edit** menu, until the deleted section is restored. After closing and reopening the template it is no longer possible to restore the deleted context this way.
In the Saving Preferences you can set whether a backup file should be created when you save the template; see "Save preferences" on page 1059.

Filling a Web page

Many of the content elements that are available for all three contexts are particularly suitable for web pages; see "Content elements" on page 761. Do not use Positioned Boxes and Tables to position elements, however; use Inline Boxes instead.

Forms and Form elements are only available in a Web context; see "Forms" on page 842 and "Form Elements" on page 847.

Using variable data on a Web page

Web templates are personalized just like any other template; see "Variable data in the text" on page 924. There are a few extra possibilities, though: variable data can be used in Form elements and they can be passed to client-side JavaScript.

Passing Variable Data to client-side JavaScript

When serving Web pages using Workflow, the HTML is first personalized and then served to the web browser by a Workflow process. At that stage custom JavaScripts do not have access to the information stored in the Data Model. To enable a client-side script to use variable data, you need to create a Text Script that produces a JSON string and stores that in the attribute of an HTML element, the value attribute of a hidden field for example. The custom JavaScript can then retrieve that information and use it to create dynamic page elements. Producing a JSON string and storing the results in the attribute of an HTML element are both options in the Text Script wizard; see "Using the Text Script Wizard" on page 948.

Styling and formatting a Web page

The contents of a Web section can be formatted directly, or styled with Cascading Style Sheets (CSS). See "Styling and formatting" on page 879 and "Styles pane" on page 1260.

In order for a style sheet to be applied to a specific section, it needs to be included in that section. There are two ways to do this.

Drag & drop a style sheet

1. Click and hold the mouse button on the style sheet on the **Resources** pane.
2. Move the mouse cursor within the **Resources** pane to the section to which the style sheet should be applied.
3. Release the mouse button.

Using the Includes dialog

1. On the Resources pane, right-click the section, then click **Includes**.
2. From the **File types** dropdown, select **Stylesheets**.
3. Choose which CSS files should be applied to this section. The available files are listed at the left. Use the arrow buttons to move the files that should be included to the list at the right.
4. You can also change the order in which the CSS files are read: click one of the included CSS files and use the **Up** and **Down** buttons. Note that moving a style sheet up in the list gives it **less** weight. In case of conflicting rules, style sheets read later will override previous ones.

Note: Style sheets are applied in the order in which they are included in a section. The styles in each following style sheet add up to the styles found in previously read style sheets. When style sheets have a conflicting rule for the same element, class or ID, the **last** style sheet 'wins' and overrides the rule found in the previous style sheet.

Note: Style sheets that are linked to (i.e. included in) a section show a chain icon in the Resources pane (see "Resources pane" on page 1250).

Note: Which style sheets are included can also be set for the **Web context** as a whole: in step 1, right-click the Web context, instead of a section.

Setting a default Web page for output

When generating output from the Web context, only one of the Web templates can be merged with each record.

To select the Web section that will be output by default:

- On the **Resources** pane, expand the **Web** context, right-click a section and click **Set as Default**.

Tip: Use a Control Script to dynamically select a Web section for output depending on the value of a data field. See "Control Scripts" on page 1098.

Including JavaScript files

Which JavaScript files are included in the a Web section, depends on a setting for that section. To change this:

1. On the **Resources** pane, right-click a section in the **Web** context and click **Includes**.
2. From the **File types** dropdown, select **JavaScripts**.
3. Choose which JavaScript files should be included in this section. The list at the left displays the JavaScript files that are present in the template's resources. The list at the right shows the style sheets and or JavaScript files that will be included in the output of the current section (or Web sections, if you have selected the Web context). Use the **Include** and **Exclude** buttons (or double-click) to move files from one list to the other.
4. Click one of the included files and use the **Up** and **Down** buttons (or drag-and-drop) to change the order in which the files are included.

For more information about using JavaScript files, see "Using JavaScript" on page 703.

Setting the title, meta data and a shortcut icon

Each Web section has a set of properties to define the title of the web page, the shortcut icon and the meta tags appearing in the web page's head (with the HTML tag: <head>, see MDN Web Docs - head).

To change these properties:

1. On the **Resources** pane, expand the **Web** context, right-click the section and click **Properties**.
2. Enter the **Page Title**. The contents of this field will go in the <title> HTML tag. (**Name** is the name of the section in the Web context; this has no effect on output.)
3. Add a **Shortcut Icon** by entering the path to the favicon.ico file, for instance images/favicon.ico.

   > **Tip:** If a valid favicon image is dragged to the Web section, it will automatically be set as a shortcut icon.

4. The **Meta Information Group** lists all <meta> tags that will be added to the header of the HTML file generated in the output. Click the **Add** button to add a new <meta> tag to the list. Then you can select the type of <meta> tag, which is either name or http-equiv, and enter the value (for a name-type meta tag) or the content. For more information on <meta> tags, see MDN Web Docs - HTML meta tag.

Adding information to the <head> via a script

When generating Web output, the Designer automatically adds the included resources to the <head>. To add other tags to the <head>, such as a <base> tag to set a default base URL/target for all relative URLs in a document, you need to write a script. If you are not familiar with scripts, see "Writing your own scripts" on page 1064 for an explanation of how scripts work.

1. Create a script: on the **Scripts** pane at the bottom left, click **New**. A new script appears in the list. Double-click on it to open it.
2. Change the name of the script, so that it reflects what the script does.

   Note: Scripts can only have the same name when they are not in the same folder.
3. Choose the option **Selector** and in the **Selector** field, type `head`.
4. Write a script that appends an element to the <head> of the web page.

Example: The following script adds a <base> element to the head of a web page.

```
results.append("<base href='https://www.w3schools.com/images/' target='_blank'>");
```

Forms

Web templates can contain Forms. Capture OnTheGo templates always contain a Form.

Tip: To create a Capture OnTheGo template, preferably use a Template Wizard (see "Capture OnTheGo template wizards" on page 717). The Wizard doesn't just add the form, it also adds the necessary Capture OnTheGo form elements (see "COTG Elements" on page 835), style sheets and JavaScript files, and extra pre-made elements.

Adding a Form

This procedure describes how to add a Form element to an existing Web context.

1. On the **Resources** pane, expand the **Web** context and double-click a Web page to open it.
2. To use the Form Wizard, select the **Insert > Form Wizard** menu option. The Form Wizard adds a Form to the Web page including the specified fields.
   Alternatively, you can select **Insert > Form** on the menu to open a dialog that lets you

set the Form's properties, validation method and location, but doesn't allow you to specify fields. If you choose this method, skip step 8 and 9 of this procedure and add fields after inserting the Form (see "Adding elements to a Form" on page 698).

3. Add an **ID** and/or a **class**. ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).
4. In the **Action** field, enter the URL where the form data should be sent. The URL should be a server-side script that can accept form data. The **action** of a Capture OnTheGo form should specify the Workflow HTTP Server Input task that receives and handles the submitted data. The action will look like this: **http://127.0.0.1:8080/action** (8080 is Workflow's default port number; 'action' should be replaced by the HTTP action of that particular HTTP Server Input task).The **method** of a Capture OnTheGo form should be **POST** to ensure that it doesn't hit a data limit when submitting the form. The GET method adds the data to the URL, and the length of a URL is limited to 2048 characters. Especially forms containing one or more Camera inputs may produce a voluminous data stream that doesn't fit in the URL. GET also leaves data trails in log files, which raises privacy concerns. Therefore POST is the preferred method to use.
5. Using the the **Method** drop-down, select whether the form should be sent using the GET or POST method.
6. Using the next drop-down, select the form's Encryption Type (**enctype**):
    - **application/x-www-form-urlencoded**: Default. All characters are encoded before they are sent. Spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values.
    - **multipart/form-data**: No characters are encoded. This value is required when you are using forms that have a file upload control.
    - **text/plain**: Spaces are converted to "+" symbols, but no special characters are encoded.
7. Select a **validation** method:
    - The **Browser** validation method leaves it up to the browser to validate the user input. When adding fields to the Form (see the next step) you can only make fields required and set the maximum length as an additional requirement for some fields.
    - Select **JQuery Validation** to validate using JQuery scripts. This allows you to specify stricter requirements per field and type a different message for each

field to display to the user if the input is not valid. This method ensures a more consistent validation as it is browser independent. The necessary JQuery files will be added to the JavaScript folder on the Resources pane when the form is inserted.

8. Under **Fields**, click the **Add** button and click on the desired field type to add a field of that type; see "Form Elements" on page 847.

   Note: A Fieldset is not available in the Form Wizard, because a Fieldset itself can contain multiple different fields. Add the Fieldset after inserting the Form; see "Adding elements to a Form" on page 698.

9. Double-click each field in the Fields list and change its settings. For an explanation of the settings, see "Adding elements to a Form" on page 698.

10. The order of the elements in the list under **Fields** determines in which order the elements will be added to the Form. Use the **Move Up** and **Move Down** buttons to change the order of the elements in the list.

11. Use the **Location** drop-down to select where to insert the element.

    - **At cursor position** inserts it where the cursor is located in the template.
    - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the tag.*
    - **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*
    - **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
    - **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph ( ).*
    - **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

    * If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

> **Note:** HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph, the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

12. Close the dialog. Now you can start adding elements to the Form (see "Using Form elements" on page 698, "Form Elements" on page 847, and "COTG Elements" on page 835).

Changing a Form's properties and validation method

Once a Form has been added, you can of course edit its HTML code directly in the Source view of the workspace. Apart from that, there are a number of dialogs to change a Form's properties and validation settings.

Changing a Form's properties

1. Select the form (see "Selecting an element" on page 766).
2. On the **Attributes** pane you can change:
    - The **ID** and/or **class**. ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).
    - An **Action**: the URL where the form data should be sent. The URL should be a server-side script that can accept form data (a Workflow process that starts with a Server Input task, for example).
    - A **Method**: this defines whether the form should be sent using the GET or POST method.
    - An Encryption Type (**enctype**):
        - **application/x-www-form-urlencoded**: Default. All characters are encoded before they are sent. Spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values.
        - **multipart/form-data**: No characters are encoded. This value is required when you are using forms that have a file upload control.

- **text/plain**: Spaces are converted to "+" symbols, but no special characters are encoded.

Changing a Form's validation method

In Connect OL Connect, there are two ways in which a Form's input can be validated:

- The **Browser** validation method leaves it up to the browser to validate the user input. When adding fields to the Form (see the next step) you can only make fields required and set the maximum length as an additional requirement for some fields.
- **JQuery Validation** validates input using JQuery scripts. This allows for stricter requirements per field and a different message for each field to display to the user if the input is not valid. This method ensures a more consistent validation, as it is browser independent. The necessary JQuery files will be added to the JavaScript folder on the Resources pane when this option is chosen.

To change a Form's validation method:

1. **Right-click** on the Form element and choose **Validation settings**.
2. Choose a validation type (see above).
3. Double-click each field in the list to edit their validation settings:
    - **Required**: Check if the field is required to submit the form. If a field is required but contains no data, a message will be shown to the user.
    - **Minimum length**: Enter a numerical value for the minimum character length required for this field.
    - **Maximum length**: Enter a numerical value for the minimum character length accepted for this field.
    - **Equal to**: Use the drop-down to select another field that is already added to the same Form. The contents of both fields must match for the data to be validated. This is useful for confirmation fields such as for passwords, email addresses etc.

    Which of these options are available depends on the validation method of the form: with Browser validation you can only make a field required and set a maximum length.

Changing a Form's validation in HTML

In HTML, the validation method is stored in the data-validation-method attribute of the <form> element, with the value "browser" or "jquery-validation".
A custom message to be shown when validation of a particular Form element has failed, can be stored in the data-custom-message attribute of the Form element, for example:

<input id="email1" name="email1" data-custom-message="Enter a valid email address." type="email" required="">

Validation in Connect 1.0.0

In Connect 1.0.0, the validation method of the template was stored using the names "standard" and "custom". Standard has changed to "browser" and custom is now "jquery-validation". When you open a template made with that version of the software, the template will be migrated to use the new attribute values for the data-validation-method attribute of the <form> element. The JavaScript file web-form-validation.js will not be migrated: delete that file and then change the Form's validation method to jQuery Validation, as described above. When you click OK, the new version of the web-form-validation.js file will be added.

Submitting a Form

When a form is submitted, by clicking or touching the Submit button, the **name** and **value** of form elements are sent to the address that is specified in the Form's action (see "Adding a Form" on page 693 or "Changing a Form's properties" on page 696). If the name attribute is omitted, the data of that input field will not be sent at all.

The Form's validation should ensure that the data that the user submits is valid.

Using Form elements

Web Form elements can be used in a Web Form or in a Capture OnTheGo Form (see "Forms" on page 842 and "Capture OnTheGo" on page 707). This topic explains how to add these elements to a Form and how to prepare them so that when the Form is submitted, they provide valid data that can be handled easily.

For a list of Form elements, see "Form Elements" on page 847.
For a list of the extra elements that can be used in a Capture OnTheGo form, see "COTG Elements" on page 835.

Adding elements to a Form

To add an element to a Form or Fieldset, click inside the Form or Fieldset, select **Insert > Form elements**, and choose the respective element on the menu. (When the element isn't available via the menu, see the tip below.) Now you can change the element's settings:

1. Add an **ID** (required) and, optionally, a **class**.

   Note: The ID will be copied to the `name` attribute of the element. The `name` attribute is what identifies the field to the receiving server-side script. To change

the name, select the element after inserting it and type the new name on the **Attributes** pane.

ID's and classes are also useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).

2. Type a label, or choose No label under Style, to omit the label. (For Label elements there are no other options to be set.)
3. If applicable, choose a style for the label (for the label of a Checkbox, for example, you can't set a style).
    - **Wrap input with label** places the input element inside the Label element.
    - **Attach label to input** ties the label to the input element using the `for` attribute of the Label element.
    - **Use label as placeholder** inserts the given label text in the placeholder attribute of the field.
    - **No style** omits the label altogether.

    Note: The first two label styles ensure that when the user clicks the label, the input element gets the focus.
4. The following options are only available for specific elements:
    - For a **Text Area** you can specify a **number of rows**.
    - For a **Radio Button**, the **submit name** indicates to which Radio Button Group the Radio Button belongs.
    - For a **Button**, **Checkbox**, **Hidden Field**, and **Radio Button** you can set the **value**. The value is associated with the input and will be sent on submitting the Form.

      Tip: For other Form elements, you can set the **default value** to be the value of a field in the record set; see "Specifying a default value" on page 701.
    - For a **Checkbox** or **Radio Button** you can check **checked** or **selected** respectively for the element to initially be checked/selected when the web page is shown.
    - For a **Button**, you can set the **button type**:
        - **Submit**: The button will validate the form data and if validation is successful, send the data to the provided URL (the action specified for the

Form; see "Changing a Form's properties" on page 845).

- **Reset**: The button will reset the form to its original configuration, erasing any information entered and options provided. **Note**: This cannot be undone!

5. Depending on the validation method of the form (see "Changing a Form's validation method" on page 846) and the type of element there are a number of options to set under **Validation**. With Browser validation you can only make a field required and set a maximum length.
    - **Required**: Check if the field is required to submit the form. If a field is required but contains no data, a message will be shown to the user.
      Note that some elements can only be made required via the Form's validation settings (see "Changing a Form's validation method" on page 846). It isn't always useful to make a field required; after all, if it has a default value it will never be empty.
    - **Minimum and maximum length**: Enter a numerical value for the minimum and maximum character length required for this field.
    - **Equal to**: Use the drop-down to select another field that is already added to the same Form. The contents of both fields must match for the data to be validated. This is useful for confirmation fields such as for passwords, email addresses etc.
6. Under **Warnings**, type the message that will be displayed to the user if the input is not valid.

The **name** attribute of Form elements is sent to the server (together with the input value) after the form has been submitted. When adding an element to a Form or Fieldset, you cannot specify a `name`; the ID will be copied to the element's `name` attribute. After adding the element to the Form or Fieldset you can change the `name` on the Attributes pane.

Adding new HTML5 elements

HTML5 added several new input element types that can't be found in the Designer menu. To add such an element to a template you can do the following:

1. Add an input element from the menu, for example a Text or Button.
2. Select the element in the template.
3. On the **Attributes** pane, select the desired input type from the **Type** drop-down list.

Changing a form element

Once an element has been added to a Form, it can easily be changed: simply select the element in the template, go to the the **Attributes** pane, and edit the element. An input element can even be changed to another type of input element by selecting the desired input type from the **Type** drop-down list.

Specifying a default value

Attribute a default value to a Text, Textarea and other Form elements by dragging a field from the Data Model pane directly onto the field, once it has been created. This also works when dragging a field from a detail table in a record set into a Form element that is contained within a Dynamic Table.
Note that the default value doesn't disappear when the user clicks the field, as placeholders do. To insert a placeholder in a field, type a label and choose **Use label as placeholder** as its style when adding the element to the form; see "Adding elements to a Form" on page 698.

Making elements required

To change the validation of a COTG or Form element, right-click the element and choose **Validation settings**. Now you can change the Form's validation method and set the requirements per field; see "Changing a Form's validation method" on page 846.

Grouping data using arrays

Grouping data using arrays

In a Connect solution, when a Web Form or COTG Form is submitted, there is a Workflow process that receives the data and creates a **job data file** (which is an XML file). Having arrays in the job data file greatly simplifies creating a data mapping configuration and looping over data in Designer scripts. Here's how to group data in the HTML so that they get submitted as arrays.

> **Note:** To enable submitting arrays, you need to check the **Use PHP arrays** or **Use enhanced PHP arrays** option in the HTTP Server user preferences in Workflow; see Workflow Online Help.

A simple method to create arrays in the job data file is to use **two pairs of square brackets** in the name of the form inputs. Put the name of the array between the first pair of square brackets. Between the second pair of square brackets, define the key to which the value belongs. Consider the following HTML form inputs:

```
<input type="hidden" name="user_account" value="pparker@eu.objectiflune.com">
<input type="text" name="name" value="Peter Parker">
```

```
<input type="text" name="company" value="Objectif Lune">
<input type="text" name="pinElm1[pin_0][left]" value="122">
<input type="text" name="pinElm1[pin_0][top]" value="253">
<input type="text" name="pinElm1[pin_0][type]" value="dent">
<input type="text" name="pinElm1[pin_1][left]" value="361">
<input type="text" name="pinElm1[pin_1][top]" value="341">
<input type="text" name="pinElm1[pin_1][type]" value="dent">
```

With the **Use PHP arrays** option enabled in Workflow, the above HTML results in the following XML:

```
<values count="4">
	<user_account>pparker@eu.objectiflune.com</user_account>
	<name>Peter Parker</name>
	<company>Objectif Lune</company>
	<pinElm1>
		<pin_0>
			<left>122</left>
			<top>253</top>
			<type>dent</type>
		</pin_0>
		<pin_1>
			<left>361</left>
			<top>341</top>
			<type>dent</type>
		</pin_1>
	 </pinElm1>
</values>
```

With the **Use enhanced PHP arrays** option, the XML looks similar, but in this case, the value between the first pair of square brackets is expected to consist of two parts, separated by an underscore (e.g. row_0). The first part becomes the element's name. All content after the first underscore (preferably an integer) is given as an attribute of the element (e.g. <row _ idx=0>). The above HTML results in the following XML:

```
<values count="4">
	<user_account>pparker@eu.objectiflune.com</user_account>
	<name>Peter Parker</name>
	<company>Objectif Lune</company>
	<pinElm1>
		<pin _idx=0>
```

```
                        <left>122</left>
                        <top>253</top>
                        <type>dent</type>
                </pin>
                <pin _idx=1>
                        <left>361</left>
                        <top>341</top>
                        <type>dent</type>
                </pin>
        </pinElm1>
</values>
```

This option makes it easier to select all elements on the same level in a data mapping configuration, and to convert the XML to a JSON object.

You can try out this feature with the COTG Time Sheet template. The COTG Fields Table element (see "Fields Table" on page 839) in that template has an Add button to add rows to a table, and groups data following this approach.

Getting the status of unchecked checkboxes and radio buttons

Unchecked checkboxes and radio buttons are not submitted (as per standard HTML behavior), so how to get the state of those checkboxes and radio buttons? A common approach to get the state of unchecked checkboxes and radio buttons is to add a hidden field to the Form with the same name as the checkbox or radio button, for example:

```
<input type="hidden" name="status_1" value="0" />
<input type="checkbox" id="status_1" name="status_1" value="1" />
```

When multiple fields with the same name are encountered, the previous value is overwritten. This way the values for unchecked checkboxes and radio buttons can be processed easily.

> **Tip:** The Capture OnTheGo (COTG) plugin automatically adds a hidden field for every unchecked checkbox on a Form when the Form is submitted. It does this for every Form; the template doesn't have to be a COTG template. (See: "Using the COTG plugin" on page 739.)

Using JavaScript

JavaScript files, libraries and frameworks can be added to a template, primarily for use in Web pages and Capture OnTheGo Forms.

Which kind of library or framework you'll want to work with depends on the type of features you really desire.

Some JavaScript files are added automatically: When you create a template with a COTG Template Wizard (see "Capture OnTheGo template wizards" on page 717), the Designer automatically adds the **jQuery** library v. 3.5 and the COTG library: **cotg-2.1.0.js**. This also happens when you add a Capture OnTheGo (COTG) element to a template that you didn't start with a COTG template wizard. For more information about this plugin, see "Using the COTG plugin" on page 739.Capture OnTheGo and Jumbotron template wizards automatically add the **Foundation** files v. 5.5.1 to the resources of the template. For more information about the use of Foundation in the Designer, see "Using Foundation" on page 721.

Tip: It is possible to open and edit any JavaScript file in the Designer: select **File > Open**, select **All files (*.*)** as the file type and then select a JavaScript file.

Adding JavaScript files to the Resources

Adding a JavaScript file

To add a JavaScript file to the resources:

1. Add the file:
    - Right-click the **JavaScript** folder on the **Resources** pane, and click **New JavaScript**. Double-click it to open and edit it.
    - Alternatively, **drag and drop** the JavaScript file from the Windows Explorer to the JavaScript folder on the Resources pane.
    - Or **import** one or more JavaScript files from another template; see "Import Resources dialog" on page 1155.
    - Or, right-click a JavaScript file already in the Resources and select **Duplicate**.
2. Make some settings for the file: right-click the file on the **Resources** pane and select **Properties**.
    - **Defer** postpones the execution of the script until the page in which it is included has finished parsing. This attribute is required by APIs like Google Maps.
    - When **async** is checked, the script executes asynchronously with the rest of the page (while the page continues the parsing).

    When neither option is checked, the script is fetched and executed immediately, while the parsing of the page is paused.

Now the JavaScript file is ready to be used in your Web templates; see "Including a JavaScript file in a Web context" on the facing page.

Adding a remote JavaScript file

A Remote JavaScript Resource is a file that is not located within your template but is hosted on an external web server , generally called a **CDN**. Popular hosted frameworks on CDN networks are:

- jQuery on MaxCDN
- Zurb Foundation on CDNJS
- Bootstrap on MaxCDN
- Multiple frameworks on Google Developers

When generating Web output, these files are referenced in the web page's header and are served by the remote server, not by the Connect Server module.

There are a few advantages to using remote resources:

- These resources are not served by your server, saving on space, bandwidth and processing.
- Using a popular CDN takes advantage of caching - a client having visited another website using that same CDN will have the file in cache and not re-download it, making for faster load times for the client.

To add a remote JavaScript:

1. Right-click the **JavaScript** folder on the **Resources** pane, and click **New Remote JavaScript**.
2. Enter a name for the file as it appears in the JavaScript resources. For better management, it's best to use the same file name as the remote resource.
3. Enter the URL for the remote resource. This must be a full URL, including the http:// or https:// prefix, domain name, path and file name.
4. Optionally, check **defer** or **async** to add the async or defer attribute to the <link> element in the <head> of the segment.
   Defer postpones the execution of the script until the page has finished parsing. This attribute is required by APIs like Google Maps.
   When **async** is checked, the script executes asynchronously with the rest of the page (while the page continues the parsing).
   When neither option is checked, the script is fetched and executed immediately, while

the parsing of the page is paused.

5. Optionally, for a Capture OnTheGo Form, you can check **Use cached Capture OnTheGo resource**, to prevent downloading a remote JavaScript file again if it has been downloaded before. The file should be available on a publicly accessible location, for example: a folder location on a corporate website, hosted by a CDN (Content Delivery Network) or shared via a Workflow process.

Note: In Workflow, when using the Create Web Content task, check the **Embed All Resources** option to download and embed all remote resources. (See Workflow Help: Create Web Content.)

Tip: After adding the remote file, you may right-click it and select **Download Resource**. This allows you to maintain a central file, from which you can quickly download a copy to your template without having to copy & paste.

Note that a local copy of a remote resource is a snapshot; it is not automatically kept in sync with its remote content. You can download the remote resource again to overwrite the local copy with updated content. If you don't want a local copy to be overwritten you should rename it before downloading the remote resource again.

Using JavaScript files in a template

Including a JavaScript file in a Web context

To link a JavaScript file to the Web context, or to a certain Web section or COTG template:

1. On the **Resources** pane, expand the **Contexts** folder, and then either right-click the Web **context**, or expand the Web context and right-click a Web **section**.
2. Click **Includes**.
3. From the **File types** dropdown, select **JavaScripts**.
4. The available JavaScript files are listed at the left. Use the arrow buttons or double-click to move the JavaScript files that should be included to the right-hand list. Using the **Up** and **Down** buttons or drag-and-drop you can change the order of the files, too.
5. Click **OK**.

Note: JavaScript files that are linked to (i.e. included in) a section show a chain icon in the Resources pane (see "Resources pane" on page 1250).

Note: Any JavaScript files included in a section run **after** the scripts in the Scripts pane.

Using JavaScript in other contexts

Email clients do not support JavaScript. Therefore, Email contexts cannot include JavaScript resources.

When a JavaScript file is included in a Print section, the Designer itself acts as the browser. When generating Print output, it runs the JavaScript after generating the main page flow contents and the pagination, but before any Master Pages are added. So it is possible to change Print output (except content on Master Pages) by a JavaScript; for example, to add a barcode that includes the page number to each document. A warning is appropriate, however: changing the Print output may change the page flow and doing so at this point may result in bad output and/or serious errors or a crash of the software.

Previewing a template with JavaScript resources

When a section is previewed in the Designer, scripts in the Resources pane are interpreted by an internal browser.

JavaScript resources included in a Web section are only executed on the Live tab, not on the Preview tab.
Remember that the final output of a Web section is eventually processed by a client browser. Since you cannot predict what type of browser will be used, you should make sure your scripts are compatible with a wide range of browser types and versions.

JavaScript resources included in a Print section are executed on the Preview tab. (The editor for a Print section does not have a Live tab.)

Note: The internal browser engine which is used as of version 2018.2, Gecko 38, is compliant with the ECMAScript 5 language specification.
This means that scripts using features described in ECMAScript 2015 (aka ES6) - such as the keyword let - will fail in the Preview and Live tab of the Designer.
Nevertheless, they should work fine when processed by a (modern) client browser.
(See for example the browser compatibility table at the bottom of the Mozilla documentation for "let": Mozilla documentation.)

Capture OnTheGo

With the Designer you can create Capture OnTheGo templates. COTG templates are used to generate forms for the Capture OnTheGo mobile application. For more information about

the application refer to these websites: Capture OnTheGo and Capture OnTheGo Help Center.

COTG Forms

A Capture OnTheGo Form is actually just a Web Form that has a number of characteristic features:

- Its **action** always specifies a Workflow HTTP Server Input task, so that when the Form is submitted, the form data is sent to the Workflow server. (See: "Specifying an action" on the next page.)
- It may contain special **COTG Input elements**, like a Signature, Geolocation, or Camera element. These require the COTG JavaScript library to be added to the template. This happens automatically when the Form is created with a COTG Template Wizard.
- Thanks to the mobile app, it may be used **offline**. The app will submit the Form data when a connection to the internet is available. Just make sure, if the Form uses remotely stored style sheets or JavaScript files, that the option 'Use cached Capture OnTheGo resource' is enabled when adding the resources to the template. This prevents that the app tries to download a file again that has already been downloaded.
- It may be **reusable**. This depends on a setting in the Output to Capture OnTheGo plug-in (found on the Connectors tab) in Workflow (see the Workflow Help: Output to CaptureOnTheGo). A reusable COTG Form is not deleted from the app's form library when it is submitted, so it can be used again.

Creating a COTG Form

A Capture OnTheGo Form is actually just a Web Form, so you could add a Form element to a Web page in the Web context without the use of a Template Wizard. It is strongly recommended however, to start the COTG Template using one of the COTG Template Wizards. They all include the appropriate JavaScript files and style sheets to create user-friendly, responsive forms; see "Capture OnTheGo template wizards" on page 717.

Tip: Have a look at the Sample Project: "Sample Project: COTG Timesheets" on page 272. This wizard creates all parts of a sample COTG project, including the form.

Before starting to create a COTG Form, take some time to structure the design process and to get familiar with the principles of form design, as explained in the topic "Designing a COTG Template" on page 714.

Reusable forms

Capture OnTheGo forms can be single-use or reusable. This doesn't depend on the design (although, of course, this should be reflected in the design). What makes a form reusable is a setting in the Output to Capture OnTheGo plugin in Workflow; see Output to CaptureOnTheGo. In the Capture OnTheGo app a reusable form is called a 'template'.

Forms for offline use

Capture OnTheGo forms can be used offline. This is the case even when the form relies on remotely stored source files like JavaScript files and style sheets, provided that the option **Use cached Capture OnTheGo resource** is checked when adding them to the form.

Specifying an action

The **action** of the Capture OnTheGo Form element should specify a Workflow HTTP Server Input task (see Workflow Help: HTTP Server Input) that receives and handles the submitted data. The action will look similiar to this: **http://192.168.175.1:8080/actionname** (where **actionname** is the HTTP action of the HTTP Server Input task).
For information about specifying an **action** for a Form, see "Adding a Form" on page 843 or "Changing a Form's properties" on page 845.

> **Note:** For testing purposes, it is possible to use another URL for the Form's action or not to specify an action at all; see "Testing a Capture OnTheGo Template" on page 735.

Filling a COTG template

Before inserting elements in a COTG Form, have the design ready; see "Designing a COTG Template" on page 714.

In a Capture OnTheGo form, you can use special Capture OnTheGo Form elements, such as a Signature and a Barcode Scanner element. For a description of all COTG elements, see: "COTG Elements" on page 835. To learn how to use them, see "Using COTG Elements" on page 729.

> **Tip:** If you have started creating your Capture OnTheGo template using a COTG Template Wizard, you can find ready-made elements in the Snippets folder on the Resources pane.

Foundation, the framework added by the COTG template wizards, comes with a series of features that can be very useful in COTG forms; see "Using Foundation" on page 721.

Naturally, Web Form elements can also be used on COTG Forms (see "Forms" on page 842 and "Form Elements" on page 847) as well as text, images and other elements (see "Content elements" on page 761).

Capture OnTheGo templates can be personalized just like any other type of template; see "Personalizing content" on page 923.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

Using JavaScript

COTG plugin

Capture OnTheGo widgets do not function without the COTG plugin: **cotg-2.x.x.js**. This plugin also makes it possible to add COTG Elements dynamically, set defaults for COTG elements, react to events that occur when a user interacts with a COTG element, etc. For more information see: "Using the COTG plugin" on page 739.

Foundation

For COTG templates created with a COTG template wizard, lots of features are already available through the Foundation framework; see "Using Foundation" on page 721.
The Foundation JavaScript files and style sheets are only added automatically when you start creating a COTG template with a template wizard; see "Capture OnTheGo template wizards" on page 717.

Other JavaScript files

You may add other JavaScript files, libraries and frameworks to a template, to enhance your Capture OnTheGo Forms; see "Using JavaScript" on page 703.

Testing the template

A Capture OnTheGo (COTG) template will be used to create a form that can be downloaded, filled out and submitted using the COTG app. Before starting to actually use the template, you will want to make sure that it produces a form that looks good and functions as expected. How to preview the form, how to submit data and how to preview the submitted data is described in another topic: "Testing a Capture OnTheGo Template" on page 735.

Sending the template to the Workflow tool

After testing the template (see "Testing a Capture OnTheGo Template" on page 735) the template must be sent to the Workflow module. Templates sent to the Workflow module can be used in any process within it.

How to send the template and the corresponding data mapping configuration to the Workflow tool is explained in another topic: "Sending files to Workflow" on page 593.

Next, you can start building a Workflow configuration that receives and handles the submitted data. The configuration should start with a HTTP Server Input task (see Workflow Help: HTTP Server Input) of which the HTTP action is the one specified in the COTG Form's action.

Receiving and extracting data from a COTG Form

When a user submits a Capture OnTheGo Form, the submitted data are sent to the address specified in the action of the form (see "Specifying an action" on page 709). They will be received by the process that starts with the matching **Server Input** task (see Workflow Help: HTTP Server Input).
The data is passed on as **XML**.
The next task in the process typically is an **Execute Data Mapping** task. This task can extract the data using a data mapping configuration that must be tailor-made for this XML. In order to create this data mapping configuration you will need an XML file with the exact same structure. The easiest way to get this is to test the template and use the "Get Job Data File on Submit" option; for instructions see "Submitting and previewing data" on page 736.

Tip: Have a look at the Sample Project: "Sample Project: COTG Timesheets" on page 272. This wizard creates all parts of a sample COTG project, including the Workflow and data mapping configuration.

Using COTG data in a template

When a user submits a COTG Form, a Workflow configuration may store the information in a database and/or push it into other Workflow processes, for example to send a letter or an email receipt. To be able to use the submitted data in a template for that letter or email receipt, follow these steps:

1. **Get sample data**
   Before you can create a template that uses COTG data that is submitted from a certain COTG Form, you have to get access to a sample of that data. There are two ways to do this:

   - Using the option **Get Job Data File on Submit** in Connect Designer; see "Testing a Capture OnTheGo Template" on page 735. This way you don't have to create a Workflow configuration first. Once the Job Data File is received by the Connect server, a dialog appears asking where to store it.
   - Using a Workflow configuration. When a user submits a Capture OnTheGo Form, the data are received by a Workflow HTTP Server Input task (see Workflow Help: HTTP Server Input) that receives and handles the submitted data. Even when no other tasks are present in that Workflow configuration, Workflow can output an XML file that contains the submitted data, in a location specified for the Send To Folder plugin in Workflow.

   Note: When a COTG Form is submitted, by clicking or touching the Submit button, the `name` and `value` of form elements are submitted. If a Checkbox or Radio Button is not checked, its name and value are not sent when the form is submitted. Fortunately, there is a workaround for this; see "Using COTG Elements" on page 729.
   The Form's validation should ensure that the data that the user submits is valid (see "Changing a Form's validation method" on page 846 and "How to make COTG elements required" on page 732).

2. **Create a data mapping configuration**
   Use the resulting XML file to create a data mapping configuration (see "Data mapping configurations" on page 337).

a. Choose **File > New > Data mapping Wizards > From XML file**.

b. Select the XML data file as its source and click **Next**.

c. Set the XML Elements option to **/request/values**. This will automatically add an extraction step for the submitted form fields.

d. Click **Finish**. The file is opened in the DataMapper and the form fields are automatically extracted including the data for the signature and camera object.

e. Save the data mapping configuration.

3. **Create a template**
   Create a Designer template and personalize it using the data mapping configuration (see "Personalizing content" on page 923). Strings, base64-encoded strings and SVG data, stored in data fields using the DataMapper can be added to the template just like any other variable data; see "Variable data in the text" on page 924. They will show up in the template **as they are**.
   Strings and base64-encoded strings show up as strings.
   SVG data will be interpreted and displayed as an image.

   Note: The Signature data returned by the COTG app (as of Android 10.6.0, iOS 10.6.0, and Windows 6.0) is formatted so that the signature will automatically be scaled to fit in the containing box in a template. With previous versions of the app the format of returned signatures could vary.

Adding Camera data to the template

The Camera widget submits a base64-encoded string, which can be put in a data field using the DataMapper. When this data field is dragged into a template, the string will show up in the content, instead of the image.
To make the image appear in a template, the data has to be used as the URL of an image. The below procedure describes how to use Camera data as an image inside a container. The benefit of this approach is that the image automatically scales to the size of the container.

1. Click the **Insert Inline Box** icon on the toolbar. The **Insert Inline Box** dialog appears.
2. Enter an ID for the box (anything will do, as long as it helps you identify the box) and click **OK**. The box is added to the text flow and can be resized if needed.

   ```
   <p>
           Div content goes here
   </p>
   ```

3. Switch to the **Source** tab and replace the content of the box: by this text:

   ```
   <img id="camera" src="" width="100%">
   ```

4. Switch back to the **Design** tab. You will see a small, empty rectangle inside at the top of the inline box.
5. Right-click the empty rectangle and choose **New Script**... in the contextual menu. The **Edit Script** dialog appears. The selector of the script is automatically set to the ID of the selected element (#camera).
   Alternatively, you could add a new script on the Scripts pane and make sure that the Selector field is set to #camera.

   ```
   results.attr("src", record.fields.photo);
   ```

6. Enter the following script code: The name of the data field (in this case: photo) must be that of the Camera data in your data model.
   This script updates the attribute “src” with the field containing the base64 image.
7. Click **OK** to save the script and toggle to the **Preview** mode to see the result. You should see your image. When you resize the inline box that surrounds the image, the image should be resized as well.

   If the inline box isn't visible, click the Show Edges button in the toolbar.

Designing a COTG Template

Designing a Capture OnTheGo template is more than adding elements to a Web form. This topic shares some insights regarding the design process and principles.

Design process

Ideally, the design process consists of the following steps.

1. **Gathering information**. It is often tempting to skip this step, especially when a Capture OnTheGo form replaces a paper form, but the research that you do to find out what the company actually needs will prove to be well worth your time. Creating specifications up front prevents discussions, reduces rework and therefore saves time.
2. **Listing the input fields** that are needed, their type, and possible input constraints. Think of how the information should be visually grouped. To get an overview of all the elements and features that can be used in a Capture OnTheGo form, check out the following pages:
   - "COTG Elements" on page 835, about elements that were specially designed for COTG.

- "Form Elements" on page 847, about elements that can be used on COTG forms and on any other Web form.
- "Using Foundation" on page 721, about elements and features that come with the Foundation framework that is added automatically by COTG template wizards.
- After creating a Capture OnTheGo template using a wizard, you can find more ready-made elements in the Snippets folder on the Resources pane.

3. **Creating mockups**. A mockup or wire frame will help you to layout the form and allows your customer to provide feedback early in the project. This will save you a lot of time: typically it is easier to change the sketch than to rework the code. In addition, mockups provide a way to do usability testing before actually creating the form. Note that mobile devices come in various sizes. It is important to adapt the form design to these screen sizes. There are various free and commercial mockup applications (both online and offline), but a sketch on paper will do too. Check out the free mockup templates from www.interfacesketch.com. Their templates are designed to help you sketch your designs for different devices on paper. Sketching tools and related techniques can be found on Zurb's website: Sharpies, Shaders and Highlighters.
4. **Creating the form**. Create the form in accordance with web design principles; see "Form design" below.
5. **Testing the form**. Even if you did proper research and showed a mockup, customers or users will likely come up with new requirements once they've seen the initial live version. Be prepared and plan for this, too.

Form design

Paper forms and web forms are very different in nature. For example, paper forms have a fixed size: the size of the paper they are printed on. Web forms can be viewed on screens with different sizes, in portrait or landscape format. Paper forms are filled out with a pen, while web forms are filled out using one's fingers or a stylus. Good form design requires an understanding on how users enter information on a mobile device and how they expect the form to look and behave.

> **Tip:** If the COTG Form replaces a paper form, it can be tempting to stick to the original layout for the sake of recognizability. Don't fall into that trap. In the end, the users - customers and employees - will be happier with a user-friendly form that adapts to different screen sizes and looks like it was designed for the web.

Most design guidelines for web forms apply to COTG forms as well. Two key concepts are responsive design and usability.

Responsive design

Responsive Design is "an approach to web design aimed at crafting sites to provide an optimal viewing and interaction experience - easy reading and navigation with a minimum of resizing, panning, and scrolling — across a wide range of devices". (Source: Wikipedia.).

With the COTG app for Android or iOS, COTG forms can be viewed on a wide variety of mobile devices, with different screen sizes. A responsive design will adapt to the size and orientation of the screen, to avoid navigation tasks like zooming in or out and scrolling horizontally. The layout may change to optimize the user experience on that device: information that is shown side by side on a larger tablet may be stacked when viewed on a smaller device.

It is complicated and time consuming to create a responsive design all by yourself. Therefore it is advisable to start creating a COTG form with a COTG Template Wizard (see "Capture OnTheGo template wizards" on the next page). All Web and COTG Template Wizards in Connect Designer make use of the Zurb Foundation front-end framework to make the templates responsive (see "Using Foundation" on page 721 and http://foundation.zurb.com/learn/about.html).

> **Tip:** In the Designer, you can test the responsiveness of a form using the Responsive Design button at the top right of the workspace.
> Some browsers also let you test the responsiveness of a form. In Firefox, for example, select Developer > Responsive Design to view a form in different sizes.

Usability

Usability defines the ease of use of a form. Is the layout intuitive? Are the form elements easy to tap on a mobile device? A visually consistent design allows the user to follow the flow while filling out the form. Below are some key usability aspects to keep in mind when designing forms.

Provide clear labels. Many modern web sites show labels inside the actual form inputs while they are empty. This saves space on the form, but once the user has entered data the label is no longer visible. Show a label at all times to help the user review his input.

Use font sizes that are big enough. On paper, smaller fonts are easier to read than on a web form. Of course, on a touch screen you can zoom in and out, but a user-friendly form doesn't force the user to do that.

Provide touch areas that are large enough. COTG forms are used on a mobile device (in the COTG app). Make sure that the user can easily tap the form elements, hyperlinks and buttons. The index finger of most adults covers an area that is between 45 and 55 pixels

wide. There should be enough white space between the form inputs so the user won't accidentally put focus on the wrong element.

Visually group related information. Use headers to mark a section. This makes it easier to navigate the form. Applying a large font size and background color will make them standout. You can use Foundation's off-canvas menu and accordeon (collapse) functionality to make it easier to navigate groups of input fields.

Provide feedback. Show what input data is expected, clearly identify which fields are required and show errors when the entered data doesn't meet the required format.

Capture OnTheGo form characteristics

Reusable forms

Capture OnTheGo forms can be single-use or reusable. This doesn't depend on the design (although, of course, this should be reflected in the design). What makes a form reusable is a setting in the Output to Capture OnTheGo plugin in Workflow; see Output to CaptureOnTheGo. In the Capture OnTheGo app a reusable form is called a 'template'.

Forms for offline use

Capture OnTheGo forms can be used offline. This is the case even when the form relies on remotely stored source files like JavaScript files and style sheets, provided that the option **Use cached Capture OnTheGo resource** is checked when adding them to the form.

Capture OnTheGo template wizards

With the Designer you can create Capture OnTheGo (COTG) templates. COTG templates are used to generate forms for the Capture OnTheGo mobile application. For more information about this application, see the website: Capture OnTheGo.

A Capture OnTheGo Form is actually just a Web Form, that you could add without a wizard, but the COTG Template Wizards include the appropriate JavaScript files for the Capture OnTheGo app, and styles to create user-friendly, responsive forms. They are built upon the Foundation framework.

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop

computers, tablets, and mobile phones. Foundation is tested across many browsers and devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

For more information about the use of Foundation in the Designer, see "Using Foundation" on page 721.

After creating a COTG template, the other contexts can be added, as well as other sections (see "Adding a context" on page 611 and "Adding a Web page" on page 689).

Tip: If the COTG Form replaces a paper form, it can be tempting to stick to the original layout. Although that may increase the recognizability, it is better to give priority to the user-friendliness of the form. Keep in mind that the COTG form will be used on a device and don't miss the chance to make it as user-friendly as possible. See "Designing a COTG Template" on page 714.

Creating a COTG template using a wizard

To create a COTG template with a template wizard:

1.
   - In the **Welcome** screen that appears after startup and when you click the Home icon at the top right, choose **Browse Template Wizards**. Scroll down until you see the **Capture OnTheGo Starter** Template Wizards.
   - Alternatively, on the **File** menu, click **New**, expand the **Template** folder, and then expand the **Capture OnTheGo Starter** folder.
2. Select a template. There are 8 types of Web Template Wizards:
   - **Blank**. The Blank COTG Template has some basic design and the appropriate form, but no actual form or COTG elements.
   - **Bill of Lading**. The Bill of Lading Template is a transactional template that includes a Dynamic Table with a checkmark on each line, along with Signature and Date COTG elements. Use this wizard as a way to quickly start any new Zurb Foundation based form for Capture OnTheGo.
   - **Event Registration**. The Event Registration Template is a generic registration form asking for name, phone, email, etc.
   - **Event Feedback**. The Event Feedback Template is a questionnaire containing different questions used to rate an experience.
   - **Membership Application**. The Membership Application Template is a signed generic request form that can be used for memberships such as gyms, clubs, etc.

- **Patient Intake**. The Patient Intake Template is a generic medical questionnaire that could potentially be used as a base for insurance or clinic form.
- **Kitchen Sink**. The Kitchen Sink Template includes a wide range of basic form and COTG form elements demonstrating various possibilities of the software.
- **Time Sheet**. The Time Sheet Template is a single page application used to add time entries to a list. This template demonstrates the dynamic addition of lines within a COTG template, as the Add button creates a new time entry. There is no limit to the number of entries in a single page. Submitted data are grouped using arrays (see "Grouping data using arrays" on page 701).

3. Click **Next** and make adjustments to the settings. The wizard remembers the settings that were last used for a COTG template.
   - **Create Off-Canvas navigation menu**: an Off-Canvas menu is a Foundation component that lets you navigate between level 4 headings (<h4>) in the form. Check this option to add the menu automatically.
   - **Submit URL**: enter the URL where the form data should be sent. The URL should be a server-side script that can accept COTG Form data.
   - The **Title** and the **Logo** that you choose will be displayed at the top of the Form.
   - **Colors**: Click the colored square to open the Color Picker dialog (see "Color Picker" on page 1143) and pick a color, or enter a hexadecimal color code (see w3school's color picker) for the page background color.
     Do the same for the background color of the navigation bar at the top and for the buttons on the Form.
4. Click **Next** to go to the next settings page if there is one.
5. Click **Finish** to create the template.

The Wizard creates:

- A **Web context** with one web page template (also called a section) in it. The web page contains an 'off-canvas' Div element, Header, a Section and a Footer element with dummy text, and depending on the type of web page, a navigation bar, button and/or Form elements.
- **Style sheets** and **JavaScript files** related to the COTG form itself and others related to the Foundation framework (see above). The style sheets can be found in the Stylesheets folder on the Resources pane. The JavaScript files are located in the JavaScript folder on the Resources pane.

- A collection of **snippets** in the Snippets folder on the Resources pane. The snippets contain ready-to-use parts to build the web form. Double-click to open them. See "Snippets" on page 867 and "Loading a snippet via a script" on page 1091 for information about using Snippets.

The Wizard opens the Web section, so that you can fill the Capture OnTheGo form.

6. Make sure to set the action and method of the form: select the form and then enter the action and method on the Attributes pane.

The **action** of a Capture OnTheGo form should specify the Workflow HTTP Server Input task that receives and handles the submitted data. The action will look like this: **http://127.0.0.1:8080/action** (8080 is Workflow's default port number; 'action' should be replaced by the HTTP action of that particular HTTP Server Input task).

The **method** of a Capture OnTheGo form should be **POST** to ensure that it doesn't hit a data limit when submitting the form. The GET method adds the data to the URL, and the length of a URL is limited to 2048 characters. Especially forms containing one or more Camera inputs may produce a voluminous data stream that doesn't fit in the URL. GET also leaves data trails in log files, which raises privacy concerns. Therefore POST is the preferred method to use.

Filling a COTG template

Before inserting elements in a COTG Form, have the design ready; see "Designing a COTG Template" on page 714.

In a Capture OnTheGo form, you can use special Capture OnTheGo Form elements, such as a Signature and a Barcode Scanner element. For a description of all COTG elements, see: "COTG Elements" on page 835. To learn how to use them, see "Using COTG Elements" on page 729.

> **Tip:** If you have started creating your Capture OnTheGo template using a COTG Template Wizard, you can find ready-made elements in the Snippets folder on the Resources pane.

Foundation, the framework added by the COTG template wizards, comes with a series of features that can be very useful in COTG forms; see "Using Foundation" on the next page.

Naturally, Web Form elements can also be used on COTG Forms (see "Forms" on page 842 and "Form Elements" on page 847) as well as text, images and other elements (see "Content elements" on page 761).

Capture OnTheGo templates can be personalized just like any other type of template; see "Personalizing content" on page 923.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

Tip: If you have started creating your Capture OnTheGo template using a COTG Template Wizard, you can find ready-made elements in the Snippets folder on the Resources pane.

Using Foundation

This topic explains how to use the Foundation Grid and other Foundation components in a Web Form or COTG Form.

Foundation

With the exception of the most basic one, all Web Template Wizards in the Designer make use of the Zurb **Foundation** front-end framework. A front-end framework is a collection of HTML, CSS, and JavaScript files to build upon. Foundation is a **responsive** framework: it uses CSS media queries and a mobile-first approach, so that websites built upon Foundation look good and function well on multiple devices including desktop and laptop computers, tablets, and mobile phones. Foundation is tested across many browsers and devices, and works back as far as IE9 and Android 2. See http://foundation.zurb.com/learn/about.html.

Capture OnTheGo and Jumbotron template wizards automatically add the Foundation files v. 5.5.1 to the resources of the template. In a future version of OL Connect, Foundation 6 will be included. If you'd rather start using the newest Foundation files right away, you have two options:

- Download the Foundation files (from http://foundation.zurb.com/sites/download.html/) and add them to the template manually.
- Use remote Foundation files from a CDN, such as https://cdnjs.com/ (search for Foundation).

See "Using JavaScript" on page 703 and "Adding CSS files" on page 883 for further instructions.

Once the Foundation files have been added to a template, you can use the Grid, as well as many other Foundation components, in your template.

Tip: Take a look in the Snippets folder on the Resources pane. After creating a template with a Capture OnTheGo or Jumbotron template wizard, this folder contains a number of ready-made elements that make use of Foundation.

The Grid

Use the **Grid** to ensure the responsiveness of a form. Using the Grid essentially means building a form or web page with Div elements (a Div is a container element, see "Div" on page 826) that have the following `classes`:

- **row**: This class identifies a Div as a horizontal block (a row) that can contain up to 12 columns.
- **columns**: This class should be used for a Div inside a Div with the class `row`. It identifies a Div as part of a row Div.
- **small-n**, **medium-n**, **large-n**: These classes indicate the number of columns that this Div occupies within in the row, on a small, medium or large screen, respectively. Replace `n` with a number, for example: small-2, large-4. If the numbers declared in one 'row' for one screen size, added together, exceed the maximum of 12, they don't fit in one row on that screen size. In that case the Div elements will appear below each other instead of next to each other.
  These classes can be combined, so that depending on the screen size, a Div can take more or less space in a row. Separate the class names with a space.

Tip: Start with the class for small screens. For example: <div class="small-3 large-6" columns>. Larger devices will inherit those styles (thanks to the mobile-first approach of Foundation's style sheet). Customize for larger screens as necessary.

This very simple layout has only one row:

```
<div class="row">
        <div class="small-2 large-4 columns">Content goes here</div>
        <div class="small-4 large-4 columns">Content goes here</div>
        <div class="small-6 large-4 columns">Content goes here</div>
</div>
```

The Div elements inside the row take up 2, 4 and 6 parts of the total amount of screen size (divided in 12 parts) on a small screen. On a large screen they each take one third of the

available space. If the class `large-4` would have been left out, the Divs would take up 2, 4 and 6 parts of the available space, regardless of the screen size.

There's more that you can do with the Grid, for example, you could center columns, or switch columns depending on the screen size they are viewed on. For information about all these possibilities, see this website: http://foundation.zurb.com/sites/docs/v/5.5.3/components/grid.html.

Adding Divs and classes to a Connect Form template

To insert a Div, select **Insert > Structural Elements > Div** on the menu. To add a class to the Div, select the Div (see "Selecting an element" on page 766) and type the class in the Class field on the Attributes pane.

To add Grid rows and columns quickly, you could also use the **Grid** snippets or **Row** snippets, found in the **Snippets** folder on the **Resources** pane after using a wizard to create a Foundation web page or a Capture OnTheGo template. For more information about Snippets, see "Snippets" on page 867. For more information about template wizards, see "Creating a Web template with a Wizard" on page 683 and "Capture OnTheGo template wizards" on page 717.

Alternatively, If you are familiar with HTML, you can open the Source tab of the Workspace and simply type the HTML to add the Div elements and classes.

> **Tip:** Use Emmet to create a Grid layout on the source tab really fast. See "Emmet" on page 662.

Other Foundation components

Foundation comes with many other components to improve and embellish Web forms and pages . A few examples:

- An **Accordion** can be used to expand and collapse content that is broken into logical sections, much like tabs. It can be very useful on long forms.
- An **Off-Canvas menu** lets the user navigate between level 4 headings (<h4>) in a Web page or form. Capture OnTheGo Template wizards offer the option to add this menu automatically.
- **Switches** are toggle elements that switch between an Off and On state on tap or click. They make use of checkbox inputs (or radio buttons) and require no javascript. Their size can be adapted, to make them easy to use on a touch screen.

For a full overview and explanation of all Foundation components (v. 5), see this web page: http://foundation.zurb.com/sites/docs/v/5.5.3/.

COTG Elements

With the Designer you can create Capture OnTheGo templates. COTG templates are used to generate forms for the Capture OnTheGo mobile application. This topic is about Capture OnTheGo form elements. For more information about the application refer to these websites: Capture OnTheGo and Capture OnTheGo Help Center.

Capture OnTheGo (COTG) elements can only be added within a Form element in a Web context; see "COTG Forms" on page 708. For information about how to add and use COTG Elements, see "Using COTG Elements" on page 729.

It is also possible to add COTG Elements dynamically, to set defaults for COTG elements and to react to custom events that occur when a user interacts with a COTG element. For more information see: "Using the COTG plugin" on page 739 and "Dynamically adding COTG widgets" on page 742.

Barcode Scanner

The Barcode Scanner element adds a button to trigger the device to scan a barcode. A very large variety of barcode types are supported. Once the barcode has been scanned, the data from the barcode will be added to the COTG Form and submitted along with it.

> **Note:** Using the Barcode Scanner element requires the installation of the ZXing Barcode Scanner application on Android devices. The application returns the barcode data after scanning.

Camera

The Camera element adds a group of buttons to capture or select an image. Once the image is selected via the camera or the device's library (aka "gallery"), it is saved within the Form data.

When the form is submitted, the image is sent in a base64-encoded string format. To learn how to add Camera data to a template, see "Adding Camera data to the template" on page 713.

The Camera element has a number of options, of which most can be set in the Design view. These options are described below.
All options, including options that cannot be set via the Design view, can be set via the Source view or in code; see "Changing default settings for widgets" on page 741.

Buttons

By default, the Camera element adds three buttons to the form:

- **Take now**: Opens the device's default Camera application to take a picture using the device's camera. Capture OnTheGo has no impact on the device's applications, so the features available (quality, orientation, filters) are dependent on the device itself. You can, however, set the format, quality and scaling for images that are submitted by the Camera element, as explained below.
- **Library**: Opens the device's default library or gallery application to select a single image that is then saved in the form data. The accessible images and navigation depend on the device itself.
- **Clear**: Removes any existing image data from the Camera element.

To omit the Take now or Library button, edit the Camera element's properties: right-click the Camera element after adding it to the form, select **Camera properties** and then use the **Source** drop-down to select which buttons will be available: Take, Pick from library, or both.

Annotations

Annotations can make a picture much more informative: an arrow, showing in which direction a car was driving; a circle, where the car has been damaged... To allow the user to make annotations, right-click the Camera element after adding it to the form, select **Camera properties**, and then check **Allow annotations**.
Clicking (or rather, touching) the image will bring up the annotation dialog. Annotations can be made in a Marker (semi-transparent) or Pencil (solid) style, in different colors and with different widths.
Annotations are submitted in SVG format by a hidden input added to the Camera element. The name of that input is the ID of the Camera element, followed by "-note-data", for example **camera1-note-data**.

Cropping/editing/deskewing

To allow the user to crop, edit and deskew the image after taking or selecting it, select **Camera properties**, and then check **Edit Image** and/or **Allow Deskew**. Which editing options the user actually gets and how they are presented to the user depends on the operating system of the device. On an Android device for example, the user may be able only to crop the image, while the user of an iOS device may select part of the image and rotate that selection.

Image format

You can set the format, quality and scaling for images that are submitted by the Camera element. Right-click the Camera element after adding it to the form, select **Camera properties** and edit the Image properties:

- **Format**: The image format can be PNG or JPG.
- **Quality**: Set the compression in a percentage.
- **Scale Image**: Check this option to enable image scaling. Then set the maximum width and height of images before they are sent to the server. Note that only the smallest of these is applied and the size ratio is always maintained.

Time stamp

A time stamp can be added to each picture taken. Right-click the Camera element after adding it to the form, select **Camera properties**, and then check **Add Time Stamp**.
The time stamp will be added to the bottom left of the picture, with medium font size, and long date format (for example: 6/15/2009 1:45 PM). These settings can only be changed via the Source tab or in code; see "Changing default settings for widgets" on page 741 and the Capture OnTheGo API: "Camera" on page 752.

> **Note:** The Time stamp feature doesn't work in versions of the app prior to 10.6.

How to use the captured or selected image in a template

After a user has submitted the form and the data has been extracted, you may want to display the captured or selected image in a Designer template, for example in a letter or on a web page. To do this:

1. Load the data mapping configuration (or at least the data model).
2. Insert a dummy image in the template.
3. Right-click the dummy image and select **Dynamic Image**. The Text Script Wizard appears.
4. Under **Field** select the field that contains the base64-encoded string. The script puts the given string in the source (**src**) attribute of the image (**<img>**).

Instead of using the Text Script Wizard, you could also write a script yourself; see "Writing your own scripts" on page 1064.

Date and Formatted Date

The Date element and the Formatted Date element display the current date on the device when the form is first opened. When the element is touched, a date selector appears so the user can modify this date. The Formatted Date element displays dates in a format that depends on the locale of the device on which the user is viewing the form. A Date Element displays dates in the ISO 8601 format: YYYY-MM-DD.

When the form is submitted, the date data is sent as plain text. A Formatted Date element submits the date in two formats: in the format that depends on the device's regional and language settings and in the ISO format mentioned above (using a hidden field). A Date element sends the date in the ISO format only.

Device Info

The Device Info Element adds a field that contains some information about the device (phone or tablet) that is submitting the COTG Form. This includes the device's type (Android or iOS), operating system version, device model and its UUID (Universally Unique Identifier). This information can be useful for both troubleshooting, if errors occur on specific device types for example, as well as for security validation: it is possible to maintain a list of device UUIDs that are allowed access, to prevent unauthorized use even if someone has a user name and password to a repository.

Document ID

The Document ID element retrieves the Document ID of the form currently viewed by the app. You could put the Document ID in a hidden input, so that when the form is submitted, the Document ID is submitted as well. A Document ID can be used on the server side to check (in the Connect database) if the data has already been submitted.

Fields Table

The Fields Table element adds a table with two rows, a Delete button at the end of the first row and an Add button at the end of the second row. Inside the rows you can put whatever elements you need. The user can click (or rather, touch) the Add button to add a row to the table. The new row will contain the same elements as the first row.
The names of all elements in the first row will be extended with __0, while the names of the elements in the second row will be extended with __1, etc. (This means that the submitted data can be grouped; see "Grouping data using arrays" on page 701.)
Nested tables - i.e. Fields Tables used within a Fields Table - are supported as of version 2021.1. They follow the same naming pattern, for example: `level1[row_0][level2][row_1][name]`.

Geolocation

The Geolocation Element adds a button to read the device's current GPS coordinates and save them in a form field. When the button is pressed, the GPS coordinates are requested and saved. When the form is submitted, the Geolocation data is sent in plain text.

High accuracy

By default, devices attempt to retrieve a position using network-based methods. To tell the framework to use more accurate methods, such as satellite positioning, the High Accuracy setting has to be enabled on the Geolocation element.
To make this setting, right-click the Geolocation element (or select it on the Outline pane) after adding it to the form, select **Geolocation properties** and check the **High Accuracy** option.

> **Note:** The Geolocation element has several options, of which only one can be set via the user interface. All options, including those that cannot be set in Design view, can be set via the `data-params` attribute in the HTML, or in code; see "Changing default settings for widgets" on page 741.

Image & Annotation

The Image & Annotation element is meant to be used with an image that needs input from the user. When inserting an Image & Annotation element you have to select the image. The user can simply click (or rather, touch) the image to bring up the annotation dialog. Annotations can be made in a Marker (semi-transparent) or Pencil (solid) style, in different colors and with different widths.
Annotations are submitted in SVG format by a hidden input. The name of that input is the ID of the Image & Annotation element, followed by "-note-data", for example **image_annotation1-note-data**.

Locale

The Locale Element does not have a UI element in the form. Inserting it adds a hidden input field that will contain the device's set locale when the form is submitted. This data is sent in plain text format and is available when processing the form data. The format is defined by the device.

Repository ID

The Repository ID element retrieves the repository ID (read only) from the app based on the currently logged on COTG user. You could put the Repository ID in a hidden input, so that when the form is submitted, the Repository ID is submitted as well. This information can be used on the server side to take specific actions, such as sending properly branded emails.

Signature

The Signature Element adds a signature box to a COTG form. These signatures are filled in via touch input, either with a finger or capacitive pen. Touching the signature box opens up a fullscreen box used to sign (generally more useful in Landscape mode depending on the device); after confirming, the dialog saves the data into the Form.
Signature data is transmitted in SVG plain text format. This type of data can be stored in a data field in a Data Mapping and dragged from the Data Model into a template as is. In Preview mode it will be displayed as an image because the Designer, just like web browsers, knows how to display this kind of data.

> **Note:** The Signature data returned by the COTG app (as of Android 10.6.0, iOS 10.6.0, and Windows 6.0) is formatted so that the signature will automatically be scaled to fit in the containing box in a template. With previous versions of the app, the format of returned signatures could vary.

Time and Formatted Time

The Time element and the Formatted Time element display the current time on the device when the form is first opened. When the element is touched, a time selector appears so the user can modify this time. The Formatted Time element displays times in a format that depends on the locale of the device on which the user is viewing the form. A Time Element displays dates in the ISO 8601 format: HH:MM.

When the form is submitted, the time data is sent as plain text. A Formatted Time element submits the time in both the ISO format mentioned above and in the format that depends on the device's regional and language settings. A Time element sends the time in the ISO format only.

User Account

The User Account Element adds a read only field that contains the Capture OnTheGo user account (an email address) that was used to submit the form to the server. This is useful if a form is available to many different users, to detect who submitted it.
If desired the field can be hidden; it will still be submitted.

Using COTG Elements

Capture OnTheGo (COTG) elements are Web Form elements that are specially designed to be used in a Capture OnTheGo Form (see "Capture OnTheGo" on page 707). This topic explains how to add these elements to a Capture OnTheGo Form or and how to prepare

them so that when the Form is submitted, they provide valid data that can be handled easily.

For a description of all COTG elements, see "COTG Elements" on page 835.

Adding COTG elements to a Form

To add a COTG element to a Form or Fieldset, click inside the Form or Fieldset, select **Insert > COTG elements**, and choose the respective element on the menu. Now you can change the element's settings:

1. Add an **ID** (required) and, optionally, a **class**.

   > **Note:** The ID will be copied to the `name` attribute of the element. The `name` attribute is what identifies the field to the receiving server-side script. To change the name, select the element after inserting it and type the new name on the **Attributes** pane.

   ID's and classes are also useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).

2. Type a label, or choose No label under Style, to omit the label. (For Label elements there are no other options to be set.)
3. If applicable, choose a style for the label (for the label of a Checkbox, for example, you can't set a style).
   - **Wrap input with label** places the input element inside the Label element.
   - **Attach label to input** ties the label to the input element using the `for` attribute of the Label element.
   - **Use label as placeholder** inserts the given label text in the placeholder attribute of the field.
   - **No style** omits the label altogether.

     > **Note:** The first two label styles ensure that when the user clicks the label, the input element gets the focus.

When you add a Capture OnTheGo (COTG) element to a template that you didn't start with a COTG template wizard, the Designer will automatically add the necessary JavaScript files: the jQuery library and the COTG library: cotg-2.1.0.js. (See: "Using the COTG plugin" on page 739.)

If a template contains an earlier version of the COTG library, the newest version will be added to the resources, but you will be asked which version of the library you prefer to use. Your preferred library will be included in the section (see: "Includes dialog" on page 1157).

The Foundation JavaScript files and style sheets will not be added. You only get those automatically when you start creating a COTG template with a template wizard. (See: "Using Foundation" on page 721.)

Element-specific settings

After inserting certain COTG elements, such as the Camera element, some important settings have to be made. These will appear when you **right-click** the element and select it from the short-cut menu.

Alternatively, you can make settings on the Source tab (see "Changing widget settings in HTML" on the facing page).
The default settings can be changed in JavaScript; see "Changing default settings for widgets" on page 741.

Attributes

Some attributes (which aren't the same as the settings mentioned above) of a COTG element can be seen on the **Attributes** pane, after selecting the element (see "Selecting an element" on page 766).

All COTG elements have a `role` attribute. This attribute is not supposed to be edited: without the correct role attribute, the element won't function.

As noted, the `name` attribute is what identifies the element after submitting the form.

Tip: Use the **Outline** pane at the left to see which elements are present in the template and to select an element.

Use the **Attributes** pane at the right to see the current element's ID, class and some other properties.

Use the **Styles** pane next to the Attributes pane to see which styles are applied to the currently selected element.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

How to make COTG elements required

To make a COTG element required, or to change the validation of a COTG Form, right-click the element and choose **Validation settings**. Set the Form's validation method to jQuery and set the requirements and a message per field. For an explanation see "Changing a Form's validation method" on page 846.

Changing widget settings in HTML

The Camera and Geolocation widgets have configurable options; you can decide which buttons will be visible in the Camera element, for example. A number of these options can be set via the user interface (see "Element-specific settings" on the previous page).
The settings are stored in JSON format in the **data-params** attribute of the element in the HTML, as you can see on the Source tab in the Designer after making a setting.
All options, including options that cannot be set via the user interface, can be set via this `data-params` attribute. To define a timeout of 6 seconds for a Geolocation element, for example, you could switch to the Source tab and change its HTML to:
`<div id="geolocation1" role="cotg.Geolocation" data-params="{"timeout":6000}">`.
Settings in the HTML override the default settings for that element. They are applied to the widget when the Form is created and cannot be changed afterwards.

For a complete list of options see the Capture OnTheGo API: "Capture OnTheGo API" on page 751.

Settings for a **dynamically added** element can be made in code; see "Dynamically adding COTG widgets" on page 742.

The default settings that are specified in the COTG plugin can also be changed in code; see "Changing default settings for widgets" on page 741.

Grouping data using arrays

In a Connect solution, when a Web Form or COTG Form is submitted, there is a Workflow process that receives the data and creates a **job data file** (which is an XML file). Having arrays in the job data file greatly simplifies creating a data mapping configuration and looping over data in Designer scripts. Here's how to group data in the HTML so that they get submitted as arrays.

Note: To enable submitting arrays, you need to check the **Use PHP arrays** or **Use enhanced PHP arrays** option in the HTTP Server user preferences in Workflow; see Workflow Online Help.

A simple method to create arrays in the job data file is to use **two pairs of square brackets** in the name of the form inputs. Put the name of the array between the first pair of square brackets. Between the second pair of square brackets, define the key to which the value belongs. Consider the following HTML form inputs:

```
<input type="hidden" name="user_account" value="pparker@eu.objectiflune.com">
<input type="text" name="name" value="Peter Parker">
<input type="text" name="company" value="Objectif Lune">
<input type="text" name="pinElm1[pin_0][left]" value="122">
<input type="text" name="pinElm1[pin_0][top]" value="253">
<input type="text" name="pinElm1[pin_0][type]" value="dent">
<input type="text" name="pinElm1[pin_1][left]" value="361">
<input type="text" name="pinElm1[pin_1][top]" value="341">
<input type="text" name="pinElm1[pin_1][type]" value="dent">
```

With the **Use PHP arrays** option enabled in Workflow, the above HTML results in the following XML:

```
<values count="4">
	<user_account>pparker@eu.objectiflune.com</user_account>
	<name>Peter Parker</name>
	<company>Objectif Lune</company>
	<pinElm1>
		<pin_0>
			<left>122</left>
			<top>253</top>
			<type>dent</type>
		</pin_0>
		<pin_1>
			<left>361</left>
			<top>341</top>
			<type>dent</type>
		</pin_1>
	</pinElm1>
</values>
```

With the **Use enhanced PHP arrays** option, the XML looks similar, but in this case, the value between the first pair of square brackets is expected to consist of two parts, separated by an underscore (e.g. row_0). The first part becomes the element's name. All content after the first underscore (preferably an integer) is given as an attribute of the element (e.g. <row _ idx=0>). The above HTML results in the following XML:

```
<values count="4">
	<user_account>pparker@eu.objectiflune.com</user_account>
	<name>Peter Parker</name>
	<company>Objectif Lune</company>
	<pinElm1>
		<pin _idx=0>
			<left>122</left>
			<top>253</top>
			<type>dent</type>
		</pin>
		<pin _idx=1>
			<left>361</left>
			<top>341</top>
			<type>dent</type>
		</pin>
	 </pinElm1>
</values>
```

This option makes it easier to select all elements on the same level in a data mapping configuration, and to convert the XML to a JSON object.

You can try out this feature with the COTG Time Sheet template. The COTG Fields Table element (see "Fields Table" on page 839) in that template has an Add button to add rows to a table, and groups data following this approach.

Getting the status of unchecked checkboxes and radio buttons

Unchecked checkboxes and radio buttons are not submitted (as per standard HTML behavior), so how to get the state of those checkboxes and radio buttons? A common approach to get the state of unchecked checkboxes and radio buttons is to add a hidden field to the Form with the same name as the checkbox or radio button, for example:

```
<input type="hidden" name="status_1" value="0" />
<input type="checkbox" id="status_1" name="status_1" value="1" />
```

When multiple fields with the same name are encountered, the previous value is overwritten. This way the values for unchecked checkboxes and radio buttons can be processed easily.

Tip: The Capture OnTheGo (COTG) plugin automatically adds a hidden field for every unchecked checkbox on a Form when the Form is submitted. It does this for every

Form; the template doesn't have to be a COTG template. (See: "Using the COTG plugin" on page 739.)

Testing a Capture OnTheGo Template

A Capture OnTheGo (COTG) template will be used to create a form, that can be downloaded, filled out and submitted using the COTG app. Before starting to actually use the template, you will want to make sure that it produces a form that looks good and functions as expected. This topic explains how to preview the form, and how to submit data and preview the submitted data.

Previewing the form

On a PC

A Capture OnTheGo template can be previewed on a PC in two different ways. Note that Capture OnTheGo form elements will not be functional unless they are sent to a device.

- Within OL Connect Designer. You can open the **Preview** tab or the **Live** tab in the Workspace. This displays the output HTML along with any variable data being added. On the **Live** tab you can even fill out the form and submit it, and if the "Get Job Data File on Submit" option is enabled (by clicking the toolbar button of the same name), the Designer will receive an XML with the submitted data (see "Get Job Data File on Submit " on page 737). However, remember that COTG Form elements are only functional in the COTG app, so they won't submit any data.
- Within the default browser on your computer. Click the **Preview HTML** button in the toolbar. This opens your operating system's default browser and displays the form in that context.

  Tip: In the Designer, you can test the responsiveness of a form using the Responsive Design button at the top right of the workspace.
  Some browsers also let you test the responsiveness of a form. In Firefox, for example, select Developer > Responsive Design to view a form in different sizes.

Previewing a COTG Template in the app

A COTG Template cannot only be previewed on a PC; it can also be previewed on a mobile device. This will show the template within the Capture OnTheGo mobile application, and all widgets will be functional.

In order to test or use any Capture OnTheGo features you need to have a **Repository** account (also called a COTG Server account or the Repository ID). You can get a trial account for this purpose; please see this page for more details: http://www.captureonthego.com/en/promotion/.

Once you have your Repository ID and Password, you also need to create a **user** account:

1. Go to the Capture OnTheGo Repository Login: https://config-us.captureonthego.com/.
2. Login with your Repository ID and Password.
3. Go to the Users page.
4. Add a new user. The user name should be in the form of an email address.

Next, make sure that the Capture OnTheGo mobile application is installed and that it is logged on as a known user of the Capture OnTheGo Repository.

Now, with your Capture OnTheGo template open in the Connect Designer module, click on the **Send COTG Test...** button in the toolbar.

Enter the appropriate information in the Send Test dialog (see "Send COTG Test" on page 1203).

Click **Finish** to send the document. It should automatically appear in the app's Repository for 4 days from the moment it is sent. Once downloaded it remains accessible in the app's Library for 2 days.

Tip: To manually delete a test template from the app's Library, swipe it to the left.

Submitting and previewing data

When you hit the Submit button in a template in the **Designer** (on the Live tab), the submitted data can be sent back to the Designer in the form of an XML file (see below). The advantage of this is that you can immediately start creating a data mapping configuration and use the data in a template.

Data submitted from the Capture OnTheGo **app** can be saved via a Workflow configuration, as explained below.

Note: The Form's validation should ensure that the submitted data is valid. Set the Form's validation method to jQuery and set the requirements and a message per field (see "Changing a Form's validation method" on page 846 and "How to make COTG elements required" on page 732).

Get Job Data File on Submit

It is possible to test a COTG Form **in the Designer** and get access to an **XML file** that contains the submitted data, without having a Workflow configuration to handle the data.

This option requires that:

- Workflow has been installed on the local machine, and the Workflow HTTP/Soap Service has been started. To do this, in the Workflow menu, click **Tools > Service Console**, then right-click **HTTP/Soap Server** and start it.
- In the Designer menu **Window > Preferences > Web**, the **Workflow URL** has been set to the correct host. The default is **http://127.0.0.1:8080/_getSampleFormData_**. This points to an internal process of the Workflow component running at that host.

If these conditions are met, you can get the XML file as follows:

1. Open the Form in the Designer, toggle to Live mode and fill out the form. Click the **Add Dummy Data** button (found on the toolbar and only available in the Live view) to populate empty form fields with dummy data. This adds dummy data to standard HTML form inputs as well as COTG inputs like the camera and signature widgets. Inputs that already contain data are left untouched. For a list of dummy data values, see the table below.
2. Click the **Get Job Data File on submit** toolbar button. This replaces the default form submit action and will send the form data to the Workflow's HTTP Service (which needs to be running in the background).

   Note: Workflow's HTTP Service must be running, but not necessarily the Workflow Service itself.
3. Hit the **Submit** button. Now the data file will be sent directly to the Designer. Once the Job Data File is received by the Connect server, a dialog appears asking where to store it.
4. Save the XML file to disk. You can view it, create or update a data mapping configuration for it (see "Data mapping configurations" on page 337), and insert the data in a template, using the data mapping configuration (see "Personalizing content" on page 923).

Note: Checkboxes and Radio buttons that are unchecked will not be submitted to the job data. This is standard behavior in HTML. One can work around that by adding a hidden field before the respective checkbox with the same name and for example value 0 (see "Using COTG Elements" on page 729).

Standard Form input dummy data values

Input	Dummy Value
Text	"Lorem ipsum dolor sit amet"
Textarea (multi-line text field)	"Lorem ipsum dolor sit amet"
Email	"pparker@localhost.com"
Number	random integer
Password	1234567890
URL	"http://www.localhost.com"
Checkbox	Checkboxes in Dynamic Tables and in the Fields Table control (time sheet) are checked.
Radio button	Selects the first radio button that is not disabled in each radio group. The radio group will be left untouched when there is a selected radio button.

Capture On The Go input dummy data values

Input	Dummy Value
Signature	Receives SVG signature data and the onscreen presentation of that data.
Camera	A dummy foto is added, and a (SVG) annotation if that option is set for the widget. Note that the script doesn't look at the PNG/JPG or resolution options, the only option it considers is the annotation option.
Geolocation widget	A geo location
Date Picker (formatted and standard)	Today's date*
TimePicker (formatted and standard)	Now*
Device Info widget	"{"available":true,"platform":"Android","version":"9.9.9","uuid":"17206724b8077491","cordova":"3.6.4","model":"Connect Designer"}"

Input	Dummy Value
User Account widget	"user@localhost.com"
Locale widget	en-US

* Note that the **formatted** date and time can be different from the values that the COTG app provides. In the COTG app the formatted date comes from the COTG API, and the formatted date and time normally depend on the locale/region settings on the mobile device. The **ISO** date and time should be the same as when using the COTG app.

Get submitted data via Workflow

Eventually, when a user submits a Capture OnTheGo Form, the data are received by the Workflow HTTP Server Input task (see Workflow Help: HTTP Server Input) that has the same HTTP action as the one specified in the Form's action (see "COTG Forms" on page 708). The Workflow configuration should then handle the submitted data. But even if it doesn't, when no other tasks are present in that Workflow configuration, Workflow can output an XML file that contains the submitted data and save it in a location specified for the **Send To Folder** plugin in Workflow.
This XML file contains the actual data submitted by all Form elements, including COTG elements, such as a signature or barcode.

Using the COTG plugin

A Capture OnTheGo (COTG) Form may contain special COTG input elements, like a Signature, Geolocation, or Camera element. These elements do not function without the **COTG JavaScript library**. It is this library that links the controls with hardware features on the mobile device.

The COTG JavaScript library offers **options** and **custom events** for COTG widgets to support event-based programming in Capture OnTheGo Forms. For example: your code can automatically set a date for a Date field and retrieve the Geolocation as soon as a form has been signed.
All available options and events are listed in the Capture OnTheGo API: "Capture OnTheGo API" on page 751.

> **Note:** Options and custom events were not available in the cotg-1.x.js versions of the library. They have been introduction in version **2.0.0** with Connect 1.8.

With OL Connect 2021.1 the COTG library has been updated to version **2.1.0** to support nested fields tables in COTG Fields Table elements. This also fixed an issue with the COTG Camera Widget that occurred when it was used in a COTG Fields Table.

This topic explains in detail how to add the COTG plugin, how to change the defaults for COTG widgets and how to use events.
How to add COTG elements to a Form dynamically is explained in another topic: "Dynamically adding COTG widgets" on page 742.

It is assumed that you have a basic understanding of HTML forms, CSS, JavaScript, and jQuery. Examples on this page use jQuery.

About jQuery

This version of the COTG library is entirely based on jQuery. jQuery is a JavaScript library that makes it very easy to select elements in a web page using HTML and CSS selectors, and to manipulate those elements. You will need to use jQuery to dynamically add widgets to a COTG Form. If you are new to it, spend a few minutes on learning it - it's that easy. For more information, see: https://jquery.com/. and http://learn.jquery.com/.

Adding the plugin

When you create a template with a COTG Template Wizard (see "Capture OnTheGo template wizards" on page 717), the Designer automatically adds the jQuery library v. 3.5 and the COTG library: cotg-2.1.0.js.
This also happens when you add a Capture OnTheGo (COTG) element to a template that you didn't start with a COTG template wizard.

If a template contains an earlier version of the COTG library, the newest version will be added to the resources, but you will be asked which version of the library you prefer to use. Your preferred library will be included in the section (see: "Includes dialog" on page 1157).

When this library is included in a Web template instead of a COTG template, it won't affect the template, except when the user submits a Form. At that moment the plugin will automatically add a hidden field for every unchecked checkbox on the Form.

Tip: If you want to take a look at the contents, you can open the plugin within the Designer: double-click **cotg-2.1.0.js** on the Resources pane.

Changing default settings for widgets

The Camera and Geolocation widgets have options that you can configure per element. You can decide, for example, which buttons will be visible in a specific Camera element (see "Element-specific settings" on page 731).
The **default** settings for these options are specified in the COTG plugin. It is possible to change these defaults without modifying the plugin itself.
To do that, create a JavaScript file (see "Adding JavaScript files to the Resources" on page 704) and specify the desired default settings in that file like this:

`$.fn.widget.defaults.setting = value;`

Make sure to include your JavaScript file in the Web context or in the Web section that contains the COTG Form (see "Including a JavaScript file in a Web context" on page 706).
All available settings are listed in the Capture OnTheGo API: "Capture OnTheGo API" on page 751.

The following code sets the default timeout and accuracy for Geolocation objects. and the default maximum height and width for Camera widgets.

```
$.fn.cotgGeolocation.defaults.timeout = 6000; // 6 secs
$.fn.cotgGeolocation.defaults.enableHighAccuracy = true;
$.fn.cotgPhotoWidget.defaults.width = 1024;
$.fn.cotgPhotoWidget.defaults.height = 768;
$.fn.cotgPhotoWidget.defaults.quality = 60;
```

Reacting to, or triggering, widget events

The new COTG plugin introduces custom events for COTG controls. You can trigger and/or react to them as the user interacts with the Form.

- Use jQuery's **.on()** method to attach an event handler to an element (or set of elements). Call this function on the `$(document).ready` event, which is triggered when the Form is loaded in the app.
- Use the **.trigger()** method to trigger an element's event.

The events of all COTG widgets are listed in the Capture OnTheGo API: "Capture OnTheGo API" on page 751.

Examples

The sample below attaches an event handler to the "set" event of a Signature element. Once the signature is set (that is, after the user has clicked the Done button), the event handler triggers events of the Date and Geolocation elements. The Date element is set with today's date and the Geolocation element is updated with the current location.

```
$(document).ready(function() {
    $('#signature').on("set.cotg", function() {
        $("#date").trigger("set.cotg", new Date()); // set current date
        $("#geolocation").trigger("update.cotg"); // get current geolocation
    });
});
```

The following example invokes the Date dialog when the user clicks a button.

```
$(document).ready(function() {
    $('#my-datepicker-button').on('click', function() {
        $('#date1').trigger("show-date-picker.cotg", new Date("2018-01-01"));
    });
});
```

Dynamically adding COTG widgets

Capture OnTheGo (COTG) widgets can be added to a Form dynamically, via jQuery. For example: a new Camera element could be added when the user clicks an Add button. This topic explains how to implement this. It is assumed that you have a basic understanding of HTML forms, CSS, JavaScript, and jQuery.

Prerequisites

Before you can start writing code that adds a widget in response to an action of the user, you need the following:

- Some element on the Form to trigger the creation of the widget. This could be anything that responds to an action of the user; a button or link, for example. Make sure that this element has an ID.
- A Form element to which the new widget can be added; a for example. Again, make sure to give this element an ID.
- The HTML structure and attributes of the widget, so that you can recreate it in code. The HTML structure of a widget can be seen on the Source tab after inserting the same kind of widget into a Form in the Designer.

Also, if you don't have a JavaScript file for your code yet, add one to the resources of your template (see "Adding JavaScript files to the Resources" on page 704) and make sure to include that file in the Web context or in the Web section that contains the COTG Form (see "Including a JavaScript file in a Web context" on page 706).

Adding an event handler

First of all you need to write an event handler that responds to the event that is meant to trigger the creation of the widget (e.g. the onClick event of a button or link), by calling the

function that creates the widget. The event handler has to be added on the $(document).ready() function, which is fired when the Form is loaded in the browser/app.

```
$(document).ready(function() {
    $('#add-element').on('click', function() {
        //call the function that creates the widget
    });
});
```

Creating the widget

Now you can start writing the code that constructs, adds and initializes the widget. This code has to be based on jQuery.

Constructing the HTML

A widget basically is an HTML element with certain attributes and contents. The HTML structure of a widget can be seen on the Source tab after adding the widget to a Form in the Designer. In code, reconstruct the HTML. Make sure to give the new element an ID.

This code constructs the HTML of a Date element:

```
<label>date1
    <input id="date1" role="cotg.DatePicker" readonly="" name="date1" type="text">
</label>
```

Adding it to the Form

Add the HTML to an element on the Form using the append() function.

The following code appends the contents of the variable html to an element on the Form that has the ID cameras:
$('#cameras').append(html);.

Initializing the widget

A widget has to be initialized to allow it to be actually used. Widgets that are already present in a Form at startup are initialized as soon as the Form is loaded in the app. A widget that has been added to a Form dynamically has to be initialized directly after adding it to the Form; otherwise the new widget won't respond to actions of the user, even though it is visible in the app.

Initializing a widget takes just one line of code, in which you select the new widget by its ID and call the initialization function on it. This code, for example, initializes a new Camera element that has the ID myCamera:

```
$(#myCamera).cotgPhotoWidget();
```

Optionally, while initializing an element, you can make settings for this specific element. These settings get prevalence over the options already specified in the HTML and over the default settings specified in the COTG plugin.

The code snippet below initializes a new Camera element (with the ID myCamera) with a number of settings:

```
$('#myCamera').cotgPhotoWidget({
    quality: 50,
    height: 1024,
    width: 1024
});
```

The initialization functions and options of all widgets are listed in the Capture OnTheGo API: "Capture OnTheGo API" on page 751.

To learn how to set the **defaults** for one kind of elements, see "Changing default settings for widgets" on page 741.

Restoring a widget

When a Form is closed, the app stores the values of input fields to the local repository of the app, but the values of input fields of dynamically added widgets are not stored.
When you reopen the Form the original input fields and their values are restored. However, dynamically added widgets are not restored; this needs to be handled in code. How to do this is explained in another topic: "Saving and restoring custom data and widgets" on the next page.

Example: adding Camera widgets dynamically

The following code inserts a Camera widget when the user clicks on a link or button with the ID add-camera. The addCameraWidget() function creates and adds the widget. Each new widget gets the class camera-dyn. The number of input elements that have this class is used to construct a unique ID for each new Camera widget.
The HTML structure of the widget was copied from the Source tab after inserting a Camera widget to a Form in the user interface of the Designer. The addCameraWidget() function appends this HTML to a with the ID cameras, which was already present in the form. Subsequently the widget is initialized so that it is linked to the COTG app and the hardware features of the device.

```
$(document).ready(function() {
    $('#add-camera').on('click', function() {
        var cameraID = "camera_" + getCameraIndex();
        addCameraWidget(cameraID);
    });
});
```

```
function getCameraIndex(){
    return $("input.camera-dyn").length;
}

function addCameraWidget(cameraID, value) {
    if(typeof value == 'undefined') {
        value = '';
    }
    var html = '<label>Camera' +
    var html = '<label>Camera' +
    '<div id="' + cameraID + '" role="cotg.PhotoWidget">' +
    '<div class="panel" role="control-wrapper" style="position:relative;">' +
    '<img role="photo" src="">' +
    '<input role="photo-data" class="camera-dyn" name="' + cameraID + '" type="hidden" value="' + value + '">' +
    '</div>' +
    '<button class="small" role="take-button" type="button">Take now</button>' +
    '<button class="small" role="pick-button" type="button">Library</button>' +
    '<button class="small" role="clear-button" type="button">Clear</button>' +
    '</div></label>';
    $('#cameras').append(html); // add the camera object to the DOM
    $('#' + cameraID).cotgPhotoWidget(); // init COTG widget
}
```

Saving and restoring custom data and widgets

The Capture OnTheGo (COTG) app stores the values of the input fields to the local repository of the app upon closing the Form. On reopening the Form the original input fields and their values are restored. However, dynamically added widgets (see "Dynamically adding COTG widgets" on page 742) don't get restored. This needs to be handled in code. This topic explains how to do that.

Adding event listeners

In the process of closing and opening a Form two important events are triggered by the COTG library:

- **olcotgsavestate**. This event is meant for custom scripts to save information when the user closes or submits the Form. It occurs after the state of all static input fields has been saved.
- **olcotgrestorestate**. This event allows custom scripts to do something when the Form is reopened. It occurs after all static inputs have been restored.

To latch on to these events, you have to register a `olcotgsavestate` listener and a `olcotgrestorestate` listener, respectively, on the `window` object. The `window` object represents a window containing a DOM (document object model), in this case the COTG Form.
The function callback on its `addEventListener` method allows you to implement custom code as part of the save and restore processes.
(For more information about the `addEventListener` method see MDN Web Docs.)

The following code registers the `olcotgsavestate` listener on the `window` object.

```
window.addEventListener('olcotgsavestate', function(event) {
/* code to save custom data */
});
```

The following snippet does the same for the restore process.

```
window.addEventListener('olcotgrestorestate', function(event) {
/* code to retrieve custom data and restore custom widgets */
});
```

The following code combines the previously saved string with the restored URL of a (static) COTG Camera element:

```
window.addEventListener("olcotgrestorestate", function(event) {
       var value = event.detail.state["myString"];
       value = value + $("#camera1 img").attr("src");
       $("form p").html(value);
};
```

Note: You should not register for these events after the document has loaded (e.g. on the `$(document).ready()` event), because these events might get triggered before the document is ready.

Place your code in a separate JavaScript file and make sure to include that file in the Web context or in the Web section that contains the COTG Form (see "Including a JavaScript file in a Web context" on page 706). For this file, the order in which JavaScript files are included in the template doesn't matter.

Handling the save and restore events

The code inside the function callback of the added event listeners should respond to the event, by saving or retrieving custom information, respectively.
When the COTG app saves the Form, you can store extra information in the `event.detail.state` object as follows:
`event.detail.state["key"] = value;`
The **key** can be any string, as long as that string isn't the same as the ID of one of the widgets on the Form.
In the code that handles the `olcotgrestorestate` event, use the same key to retrieve the stored information.

The following sample saves the value "test" using "myString" as the key, when the Form is saved:

```
window.addEventListener("olcotgsavestate", function(event) {
        event.detail.state["myString"] = "test";
});
```

The following code retrieves the value that was stored with the `myString` key and displays it in a paragraph (a <p> element) at the top of the form.

```
window.addEventListener("olcotgrestorestate", function(event) {
        var value = event.detail.state["myString"];
        $("form p").html(value);
};
```

Note: When you've used jQuery to register for the events - `$(window).on()` - you must use `event.originalEvent` in the event handler functions, for example:

```
$(window).on("olcotgsavestate", function(event) {
        event.originalEvent.detail.state["mywidget"] = "test";
});
```

Restoring widgets

When a COTG Form is reopened, the app restores all input fields and widgets that were already present in the original Form (i.e. the Form deployed via Workflow or sent as test using the Designer). Dynamically added widgets don't get restored. To restore dynamically added widgets, you have to:

- **Save** information that reveals which widgets were added dynamically, and save the values of their input fields.
  This code should go in the event handler for the `olcotgsavestate` event.
- **Add and initialize** the widgets again after the Form is reopened. Make sure to put any saved values back in the HTML.
  This code should go in the event handler for the `olcotgrestorestate` event.
- **Trigger** the `restore-state.cotg` event on the newly added widget, to make sure that it is displayed correctly. For example:
  `$('#myCamera").trigger('restore-state.cotg', event.detail.state);`
  Put this code in the event handler for the `olcotgrestorestate`. event.
  Note that the widget must have the same ID as before in order to be able to retrieve its state.

For a detailed example, see: "Saving and restoring Camera widgets" on the facing page.

Example

Saving and restoring Camera widgets

This example demonstrates a way to save and restore dynamically added Camera widgets. How to add Camera widgets is explained in another topic: "Dynamically adding COTG widgets" on page 742.

When a user takes or selects a picture with a Camera widget, the COTG app stores the path to the image that was taken or selected in a hidden input. This is the path to the image on the device that runs the COTG app. On submitting the Form the COTG app replaces this value - the path - with the actual image data.
In this example the hidden fields of dynamically added Camera elements have got the `.camera-dyn` class. In the event handler for the `olcotgsavestate` event, the jQuery `each()` method is used to iterate over all inputs that have this class, storing their name and value in an object.
Next, this object is stored - in JSON format - in the `event.detail.state` data with the key `my-camera-data`.

```
window.addEventListener('olcotgsavestate', function(event) {
    var camObj = {};
    $('input.camera-dyn').each(function(){
        var camera = $(this).attr('name');
        var val = $(this).val();
        camObj[camera] = val;
    });
    event.detail.state['my-camera-data'] = JSON.stringify(camObj);
});
```

In the `olcotgrestorestate` event handler the previously stored JSON is read from the `event.detail.state` and parsed into a JavaScript object. A `for ... in` loop is then used to iterate over the keys (the camera names) in that object. Inside this loop the cameras are added by calling the `addCamera()` function with the ID and value (the path to the picture) of each Camera element. Subsequently the `restore-state.cotg` event of the new widget is called to make sure that the thumbnail of the picture is shown and the Clear button becomes visible.

```
window.addEventListener('olcotgrestorestate', function(event) {
    var json = JSON.parse(event.detail.state['my-camera-data']);
    for (var cameraID in json) {
        var value = json[cameraID];
        addCameraWidget(cameraID,value);
        $('#' + cameraID).trigger('restore-state.cotg', event.detail.state);
    }
});

function addCameraWidget(cameraID, value) {
    if(typeof value == 'undefined') {
        value = '';
    }
    var html = '<label>Camera' +
    '<div id="' + cameraID + '" role="cotg.PhotoWidget">' +
    '<div class="panel" role="control-wrapper" style="position:relative;">' +
```

```
'<img role="photo" src="">' +
'<input role="photo-data" class="camera-dyn" name="' + cameraID + '" type="hidden" value="' + value + '">' +
'</div>' +
'<button class="small" role="take-button" type="button">Take now</button>' +
'<button class="small" role="pick-button" type="button">Library</button>' +
'<button class="small" role="clear-button" type="button">Clear</button>' +
'</div></label>';
$('#cameras').append(html); // add the camera object to the DOM
$('#' + cameraID).cotgPhotoWidget(); // init COTG widget
```

Using submitted COTG data in a template

When a user submits a Capture OnTheGo Form, the Workflow process that receives the data may store the information in a database and/or push it into other Workflow processes. It could, for example, send a letter or an email receipt that is personalized with the submitted data. That letter or email would need to be produced using a template that incorporates the submitted data.

Follow these steps to develop a template that uses submitted COTG data.

1. **Get sample data**
   First, you need to get a sample of the submitted data. There are two ways to do this:
   - Using the option **Get Job Data File on Submit** in Connect Designer; see "Testing a Capture OnTheGo Template" on page 735. This way you don't have to create a Workflow configuration first. Once the Job Data File is received by the Connect server, a dialog appears asking where to store it.
   - Using a **Workflow configuration**. When a user submits a Capture OnTheGo Form, the data are received by a Server Input task (see Workflow Help: HTTP Server Input or NodeJS Server Input) that receives and handles the submitted data. Add a Send To Folder plugin to the same process. Workflow will then output the XML file that contains the submitted data in the location specified for the Send To Folder plugin. No other tasks are needed in the process, or in the Workflow configuration.

   Note: When user of the COTG app clicks or touches the Submit button in a Form, the `name` and `value` of form elements are submitted. Normally, the name and value of an unchecked Checkbox or Radio Button would not be submitted, but in Connect, they are. Connect uses the workaround described in the following topic: "Using COTG Elements" on page 729.

   The Form's validation should ensure that the data that the user submits is valid

(see "Changing a Form's validation method" on page 846 and "How to make COTG elements required" on page 732).

2. **Create a data mapping configuration**
   The next step is to use the sample data - an XML file - to create a data mapping configuration (see "Data mapping configurations" on page 337).

a. Choose **File > New > Data mapping Wizards > From XML file**.

b. Select the XML data file as its source and click **Next**.

c. Set the XML Elements option to **/request/values**. This will automatically add an extraction step for the submitted form fields.

d. Click **Finish**. The file is opened in the DataMapper and the form fields are automatically extracted, including the data for the Signature and Camera object.

e. Save the data mapping configuration.

3. **Create a template**
   Finally, create a Designer template and personalize it using the data mapping configuration (see "Personalizing content" on page 923). Strings, base64-encoded strings and SVG data stored in data fields using the DataMapper can be added to the template just like any other variable data (see "Variable data in the text" on page 924). They will show up in the template **as they are**. SVG data will be interpreted and displayed as an image. Strings and base64-encoded strings show up as strings. In order for a base64-encoded string to be interpreted as an image, it must be added to the URL of an image.

Adding Camera data to the template

The Camera widget submits a base64-encoded string, which can be put in a data field using the DataMapper. When this data field is dragged into a template, the string will show up in the content, instead of the image.
To make the image appear in a template, the data has to be used as the URL of an image.

The below procedure describes how to use Camera data as the source of an image inside a container. The benefit of this approach is that the image automatically scales to the size of the container.

1. Click the **Insert Inline Box** icon on the toolbar. The **Insert Inline Box** dialog appears.
2. Enter an ID for the box (anything will do, as long as it helps you identify the box) and click **OK**. The box is added to the text flow and can be resized if needed.

```
<p>
        Div content goes here
</p>
```

3. Switch to the **Source** tab and replace the content of the box: by this text:

   ```
   <img id="camera" src="" width="100%">
   ```

4. Switch back to the **Design** tab. You will see a small, empty rectangle inside at the top of the inline box.

5. Right-click the empty rectangle and choose **New Script**... in the contextual menu. The **Edit Script** dialog appears. The selector of the script is automatically set to the ID of the selected element (#camera).
   Alternatively, you could add a new script on the Scripts pane and make sure that the Selector field is set to #camera.

   ```
   results.attr("src", record.fields.photo);
   ```

6. Enter the following script code: The name of the data field (in this case: photo) must be that of the Camera data in your data model.
   This script updates the attribute "src" with the field containing the base64 image.

7. Click **OK** to save the script and toggle to the **Preview** mode to see the result. You should see your image. When you resize the inline box that surrounds the image, the image should be resized as well.

   If the inline box isn't visible, click the Show Edges button in the toolbar.

Capture OnTheGo API

As of Connect 1.8, **cotg-2.0.0.js** has replaced the cotg-1.x.js versions of the Capture OnTheGo (COTG) plugin, introducing **events** and **options** for COTG widgets. This topic lists all available options and custom events for widgets, as well as their initialization function.

With OL Connect 2021.1 the COTG library has been updated to version **2.1.0** to support nested fields tables in COTG Fields Table elements. This also fixed an issue with the COTG Camera Widget that occurred when it was used in a COTG Fields Table.

How to use the COTG plugin is explained in the following topic: "Using the COTG plugin" on page 739.
To learn how to create widgets in code, see "Dynamically adding COTG widgets" on page 742 and "Saving and restoring custom data and widgets" on page 745.

For a list of all COTG elements and their intended use, see "COTG Elements" on page 835.

Barcode Scanner

cotgBarcode()

Initializing a Barcode Scanner element prepares it for user interaction.

Example: `$('myScanner').cotgBarcode();`

Events

The Barcode Scanner **listens for** the following events.

Event	Description
	Removes the scanned Barcode data.
	Opens the scanner.

The Barcode Scanner **broadcasts** the following events.

Event	Description
	This event is fired after Barcode data has been set to the value of the input.

Camera

cotgPhotoWidget([*options*])

options

Optional. An array containing the desired settings, e.g. {quality: 50, height: 1024, width: 1024}. For any unspecified options the default settings will be used.

Initialize the new Camera element with any settings that you want to be different from the defaults.

Example: `$('#myCamera').cotgPhotoWidget({quality: 50, height: 1024});`

How to change the default settings is explained in another topic: "Changing default settings for widgets" on page 741.

Option	Description	Type	Default
	Allows the user to edit the image after taking or selecting it. Which editing options the user actually gets and how they are presented depends on the operating system of the device.	Boolean	false
	The returned image's file encoding: `jpg` or `png`.	String	'jpg'
	The maximum height in pixels	Number	864
	The maximum width in pixels.	Number	1152
	Which buttons are enabled: Take now (`take`), Library (`pick`), or both (`takeandpick`).	String	'takeandpick'
	Scales the image to fit the maximum width or height. The aspect ration is maintained.	Boolean	true
	Quality of the saved image, expressed as a range of 0-100, where 100 is full resolution with no loss from file compression.	Number	80
	Adds an Image & Annotation widget to edit the picture.	Boolean	false
	Allows the user to perform a perspective cropping after taking or selecting a picture.	Boolean	false
	Adds a time stamp	Boolean	false
	The date format. For all possible formats see Cordova documentation. 'L' stands for localized date and time.	String	'L'
	The position of the stamp, specified by combining horizontal alignment (left, center, right) with vertical alignment (top, middle, bottom) and joining them with a vertical bar ('|').	String	'bottom|left'
	The time stamp's font size: `small`, `medium`, or `large`.	String	'medium'

Events

The Camera **listens for** the following custom events.

Event	Description
	Removes the picture.

Event	Description
	Saves the path of the current picture to the local storage of the COTG app.
	Restores the state of the widget when the form is reopened in the COTG app, after the app has restored previously entered values of static input fields.

The Camera **broadcasts** the following events.

Event	Description
	Fired after the user has clicked the Clear button.
	Fired after the user has taken or selected a picture.

Date and Formatted Date

cotgDatePicker()

Initializing a Date or Formatted Date element prepares it for user interaction.

Example: `$('myDatePicker').cotgDatePicker();`

Note that the difference between a Date and a Formatted Date is laid down in the HTML structure of the element.

Events

The Date and Formatted Date elements **listen** for the following custom events.

Event	Description
	Removes the date.
	Sets the given date. The date should be given as a Date object, for example: `$("#date").trigger("set.cotg", new Date()); // set current date`
	Opens the Date picker. Optionally, you can provide a date (as a Date object) for the Date picker to be opened with, for example: `$('#date1').trigger("show-date-picker.cotg", new Date("2018-01-01"));`

Device Info

cotgDeviceInfo()

Initializing a Device Info element puts information about the device (phone or tablet) that displays the form, in the hidden input of the element.

Example: `$('#myDeviceInfo').cotgDeviceInfo();`

Events

The Device Info element **listens for** the following event.

Event	Description
	Removes the Device Info data.

The Device Info element **broadcasts** the following event.

Event	Description
	This event is fired during initialization of the element, after setting its info to the current device.

Document ID

cotgDocumentId()

Initializing a new Document ID puts the current Document ID in the hidden input of the element.

Example: `$('#myDocID').cotgDocumentId();`

Events

The Document ID element **listens for** the following event.

Event	Description
	Removes the Document ID.

The Document ID element **broadcasts** the following event.

Event	Description
	This event is fired during initialization of the element, after setting its value to the current Document ID.

Fields Table

The Fields Table element itself is just a table, so it doesn't have to be initialized. However, after dynamically adding it to a Form, you still have to add the correct event handlers to the Add and Delete buttons, as follows (replace `myTable` by the ID of your table):

```
$("#myTable [role=cotg-add-row-button]").on("click", function(){
    $(this).closest('table').cotgAddRow(true);
});

$("myTable").on("click", "[role=cotg-delete-row-button]", function(){
    $(this).closest('tr').cotgDeleteRow();
});
```

Note that nested tables are not supported.

Geolocation

cotgGeolocation([*options*])

options

Optional. An array containing the desired settings, e.g. {enableHighAccuracy: true, timeout: 3000}. For any unspecified options the default settings will be used.

Call `cotgGeolocation([options])` on the new Geolocation element with any settings that you want to be different from the defaults.

Example: `$('#myGeolocation').cotgGeolocation({timeout: 6000});`

How to change the default settings is explained in another topic: "Changing default settings for widgets" on page 741.

Option	Description	Type	Default
	By default, the device attempts to retrieve a position using network-based methods. Setting this property to true tells the framework to use more accurate methods, such as satellite positioning.	Boolean	false
	Accept a cached position if it isn't older than the specified time in milliseconds.	Number	3000
	The maximum length of time (milliseconds) that is allowed to pass before the position is retrieved.	Number	2700

Events

The Geolocation element **listens for** the following events.

Event	Description
	Removes the Geolocation data.
	Restores the state of the widget when the form is reopened in the COTG app, after the app has restored previously entered values of static input fields.
	Sets the element to the current geolocation.

Image & Annotation

cotgNoteOnImage()

Initializing a new Image & Annotation element prepares it for user interaction.

Example: `$('#myNoteOnImage').cotgNoteOnImage();`

Note: To use this element with a Camera widget instead of a static image, you only have to initialize the Camera widget with the `allowannotations` option set to `true`. The Camera widget will automatically initialize the Image & Annotation widget.

The Image & Annotation element **listens for** the following events.

Event	Description
	Bind an annotation to the picture.
	Removes any annotations.
	Removes any annotations. When using this element with a Camera element, trigger this event to remove annotations without removing the picture.
	Restores the annotations when the form is reopened in the COTG app, after the app has restored previously entered values of static input fields.
	Saves the annotations.

The Image & Annotation element **broadcasts** the following events.

Event	Description
	Fired after an annotation has been made.

Locale

cotgLocale()

Initializing a new Locale element sets it to the device's locale.

Example: `$('#myLocale').cotgLocale();`

Events

The Locale element **listens** for the following event.

Event	Description
	Removes the Locale data.

The Locale element **broadcasts** the following event.

Event	Description
	This event is fired during initialization of the element, after setting its value to the current locale.

Repository ID

cotgRepositoryId()

Initializing a new Repository ID puts the current Repository ID in the hidden input of this element. The Repository ID is based on the currently logged on COTG user.

Example: `$('#myRepID').cotgRepositoryId();`

Events

The Repository ID element **listens** for the following custom events.

Event	Description
	Removes the Repositiory ID data.

The Repository ID element **broadcasts** the following event.

Event	Description
	This event is fired during initialization of the element, after setting its value to the current Repository ID.

Signature

cotgSignature()

Initializing a new Signature element prepares it for user interaction.

Example: `$('#mySignature').cotgSignature();`

Events

The Signature element **listens** for the following custom events.

Field	Description
	Removes the signature drawing and data.
	Draws the signature on the form (e.g. after a set.cotg or restore-state.cotg event).
	Called when the form is reopened in the COTG app, after the app has restored previously entered values of static input fields.
	Saves the signature data to the local storage of the COTG app.
	Sets the given signature. The signature should be given as an SVG string, for example: `$("#signature").trigger("set.cotg", "<svg>…</svg>");`

The Signature element **broadcasts** the following events.

Field	Description
	This event is fired after a user has entered a signature and clicked the Done button of the Signature window on the device.
	Fired after a user has clicked the Done button of the Signature window without entering a signature.
	Fired each time the signature is drawn on the form (by the app, not by the user; e.g. after a set.cotg or restore-state.cotg event).

Time and Formatted Time

cotgTimePicker()

Initializing a new (Formatted) Time element prepares it for user interaction.

Example: `$('#myTimePicker').cotgTimePicker();`

Note that the difference between a Time and a Formatted Time is laid down in the HTML structure of the element.

Events

The Time and Formatted Time elements **listen for** the following custom events.

Event	Description
	Stores the given time (specified in a Date object).
	Removes the set time.
	Opens the Time Picker. If no time is provided (specified in a Date object), the current time will be shown.

User Account

cotgUserAccount()

Initializing a new User Account element puts the account of the current user in the hidden input of this element.

Example: `$('#myUser').cotgUserAccount();`

Events

The User Account element **listens for** the following custom event.

Event	Description
	Removes the User Account data.

The User Account element **broadcasts** the following event.

Event	Description
	This event is fired during initialization of the element, after setting its value to the current user account.

Content elements

Once you have created a template, it can be filled with all kinds of elements: text, barcodes, tables, graphics... All types of elements are listed on this page.

There are several ways to insert elements, see "Inserting an element" on page 765.

Each element can have an ID and a class, as well as a number of other properties, depending on the element's type. When an element is selected, its properties can be changed; see "Selecting an element" on page 766, "Attributes" on page 764 and "Styling and formatting an element" on page 767.
ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file. It is possible to edit the source of the HTML file directly in the Designer; see "Editing HTML" on the facing page.

Element types

The following types of content can be added to the content of a template:

- "Images" on page 852 and "Dynamic images" on page 972
- "Text and special characters" on page 865
- "Date" on page 841
- "Table" on page 860 and "Dynamic Table" on page 974
- "Boxes" on page 823: Positioned Box, Inline Box, Div and Span.

  Tip: Wrapping elements in a box (see "Boxes" on page 823) or in a semantic HTML element makes it easier to target them in a script or in a style sheet. Place the cursor in the element or select multiple elements. Then, on the menu, click **Insert > Wrap in Box**. You can now use the wrapper element as a script's or style's selector; see "Using the Text Script Wizard" on page 948 and "Styling and formatting" on page 879.

- "Hyperlink and mailto link" on page 850

- "Barcode" on page 768
- Web "Forms" on page 842 and Web "Form Elements" on page 847
- "Whitespace elements: using optional space at the end of the last page" on page 643 (Print context only)
- "Page numbers " on page 644 (Print context only)
- Article, Section, Header, Footer, Nav and Aside are content sectioning elements; see MDN Web Docs - Content sectioning elements.
- Other HTML elements: Heading (H1 - H6), Address and Pre.
  To quickly change a paragraph into a Heading, place the cursor inside of it, or select the paragraph (see: "Selecting an element" on page 766). Then select the appropriate element, either on the **Format** menu, or from the 'Element type' drop-down on the toolbar.
- "Snippets" on page 867: a Snippet is a small, ready-to-use piece of content in a file.
- "Business graphics" on page 827

Most elements are suitable for use in all contexts. There are a few exceptions, however. Forms and Form elements can be used on web pages only, whereas Whitespace elements and Page numbers can only be used in a Print context. Positioned boxes are well suited for Print sections, but are to be avoided in the Email and Web.

Whether it is best to use a Table or Box to position text, images and other elements, depends on the context in which they are used; see "How to position elements" on page 897 for more information.

Editing HTML

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file.

To see this, toggle to the **Design** tab in the workspace. Click anywhere in the content. Take a look at the *breadcrumbs* at the top of the workspace or select the Outline pane. The breadcrumbs show the HTML tag of the clicked element, as well as the HTML tags of other elements to which the clicked element belongs. The clicked element is at the end of the line.

To edit the HTML text directly:

- In the workspace, toggle to the **Source** tab.

On this tab you can view and edit the content of the template in the form of plain text with HTML tags (note the angle brackets: <>). You may add and edit the text and the HTML tags, classes, ID's and other attributes.

To learn more about HTML, see for example:

- MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction
- W3schools: http://www.w3schools.com/html/default.asp

Many video courses and hands-on courses about HTML (and CSS) are offered on the Internet as well, some for free. Go, for example, to www.codeschool.com or www.codeacademy.com and look for HTML (and CSS) courses.

Emmet

Emmet is a plugin that enables the lightning-fast creation of HTML code though the use of a simple and effective shortcut language. The Emmet functionality is available in the HTML and CSS source editors of Connect Designer. Emmet transforms abbreviations for HTML elements and CSS properties to the respective source code. The expansion of abbreviations is invoked with the **Tab** key.

In the Source tab of the Workspace, you could for example type `div.row`. This is the abbreviation for a element with the class row.
On pressing the Tab key, this abbreviation is transformed to:

```
<div class="row"></div>
```

To quickly enter a table with the ID 'green', one row, and two cells in that row, type:

```
table#green>tr>td*2
```

On pressing the Tab key, this is transformed to:

```
<table id="green">
	<tr>
		<td></td>
		<td></td>
	</tr>
</table>
```

> **Tip:** For more examples, see this how-to: Streamline coding with emmet and custom abbreviations.

All standard abbreviations can be found in Emmet's documentation: Abbreviations.

To learn more about Emmet, please see their website: Emmet.io and the Emmet.io documentation: http://docs.emmet.io/.

Preferences

To change the way Emmet works in the Designer, select **Window > Preferences**, and in the Preferences dialog, select **Emmet**; see "Emmet preferences" on page 1049.

Attributes

ID and class

Every element in the content of a template can have an **ID** and a **class**. ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).

You can specify an ID and/or class when you add the element to the content.

To add an ID and/or class to an element that has already been added to a template, select the element (see "Selecting an element" on page 766) and type an ID and/or a class in the respective fields on the **Attributes** pane at the top right.

> **Note:** Each ID should be unique. An ID can be used once in each section.

Other attributes

Apart from the ID and class, elements can have a varying number of properties, or 'attributes' as they're called in HTML (see "Editing HTML" on page 762). Which properties an element has, depends on the element itself. An image, for example, has at least four attributes: `src` (the image's URL), `alt` (alternate text), `width` and `height`. These attributes are visible on the **Attributes** pane when you click an image in the content.

For each type of element, a small selection of attributes is visible on the **Attributes** pane at the top right.

In a multilingual template, the proprietary `data-translate` attribute marks an element for translation. For more information see "Translating templates" on page 1120 and "Tagging elements for translation" on page 1122.

Changing attributes via script

Many attributes can be changed via the user interface. Another way to change attributes is by using a script.

Any of the Script Wizards can produce a script that changes an attribute of an HTML element. Set the **Options** in the Script Wizard to **Attribute**, to output the script's results to the value of a specific attribute. See "Using the Text Script Wizard" on page 948.

In code, you can change an element's attribute using the function `attr()`; see "Writing your own scripts" on page 1064 and "Standard Script API" on page 1481.

Inserting an element

To insert an element in a section:

1. Navigate to where you want to insert the element, using the arrow keys, the mouse, the Breadcrumbs (see "Selecting an element" on the facing page) or the Outline pane.
2. Click the respective toolbar button. Alternatively, click the element on the **Insert** menu.
3. Add an ID and/or a class. ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).

   Note: Do not give an element the ID 'pages' or the class name 'dynamic'. These are reserved words. Using them as an ID or class name leads to undesirable effects.

4. Use the **Location** drop-down (if available) to select where to insert the element.
   - **At cursor position** inserts it where the cursor is located in the template.
   - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the tag.*
   - **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*
   - **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
   - **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph ( ).*
   - **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

   * If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

Note: HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph, the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

For a list of links to the different types of elements, see "Element types" on page 761.

Selecting an element

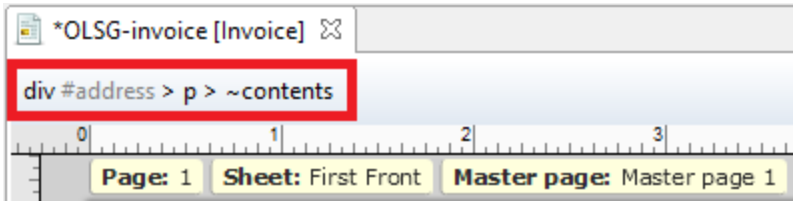
When an element is selected, the **Attributes** pane shows the attributes of that element, and the **Styles** pane, next to the **Attributes** pane, shows which styles are applied to it.

To select an element in the content, you can of course click on it, but this isn't always as easy as it seems, especially when the element has elements inside it.

Tip: Click the **Edges** button on the toolbar temporarily adds a frame to certain elements on the Design tab. These will not Print or output.

There are two more ways to select an element in the content:

- Using the *Breadcrumbs* at the top of the workspace.



  Breadcrumbs show the HTML tag of the clicked element, as well as the HTML tags of 'parent elements': elements inside of which the clicked element is located. The clicked element is at the end of the line.
  Elements with classes or IDs show these details next to them, for instance `div #contents > ol.salesitems > li ~contents`.
  Click any of the **elements** in the Breadcrumbs to select that element. If an element is selected in the Breadcrumbs and the Backspace key is pressed, that element is deleted.
  Click `~contents` to select the **contents** of the element. This way you may, for example, quickly change the text of a hyperlink.

- Using the **Outline** pane. You can find this pane next to the **Resources** pane. It displays a tree view of the elements in the file. Click an element in the tree view to select it.

Deleting an element

To delete an element, select it - as described above - and press the Delete key.
If the deleted element was targeted by a script, you will be asked if you want to delete the script as well.

Scripts are used to personalize templates. To start learning more about scripts, see "Personalizing content" on page 923 and "Writing your own scripts" on page 1064.

Styling and formatting an element

Format elements directly

Images and other graphical elements can be resized by clicking on them and dragging the resize handles. There are toolbar buttons to color, indent or style text. Other toolbar buttons can left-align, right-align, or rotate graphical elements.

The toolbar buttons only represent a selection of the formatting options for each element. There are no toolbar buttons to change an element's margins, or to add a border to it, for example. To access all formatting properties of an element, you have to open the Formatting dialog. There are two ways to do this:

- Right-click the element and select the type of element on the shortcut menu.
- Select the element (see "Selecting an element" on the previous page) and select the type of element on the **Format** menu.

See "Styling and formatting" on page 879 for more information about the formatting options.

Format elements via Cascading Style Sheets (CSS)

It is highly recommended to use style sheets in templates right from the start. Even more so if the communications are going to be output to different output channels, or if they consist of different sections (for example, a covering letter followed by a policy). Using CSS with templates allows a consistent look and feel to be applied. A style sheet can change the look of multiple elements, making it unnecessary to format each and every element in the template, if the company's layout preferences change. See "Styling templates with CSS files" on page 881.

Barcode

In OL Connect Designer, you can add a variety of barcodes to your template. The supported barcode types include 1d barcodes (the striped ones) and 2d barcodes (encoded horizontally and vertically).

Note: When generating Print output, you can add extra barcodes and OMR marks. This is particularly useful when you might need to drive custom processes on production machines using either barcodes or OMR marks. At merge time more information is available about the actual output document. The page count, for example, is not available at design time.
Having a barcode added at the time of printing is also a time saver in an automated process where only a barcode needs to be added to an existing PDF and a template is actually unnecessary. In this case, content creation can be skipped. (See "Print processes with OL Connect tasks" on page 306.)

To add barcodes and OMR marks on the fly when generating Print output from the Designer, select **File > Print** and check the option **Add additional content** (see "Additional Content" on page 1409) in the Print Wizard. To have this done automatically, save this and other output options in an Output Preset (see "Output Presets" on page 1375).

Send the preset to Workflow to use it in Workflow configurations. See "Sending files to Workflow" on page 593.

Adding a barcode

Before adding a barcode, load data or at least a Data Model; see "Loading data" on page 926.

Then, to add a barcode to a section, Master Page or snippet:

1. Select **Insert > Barcode** on the menu or click the **Barcode** toolbar button.

   Tip: A KIX code can also be part of an address block created by the Address Block wizard. See "Address Block" on page 955.

2. Choose the desired barcode **type**.
3. Select the **field** that contains the barcode value. The barcode type dictates the length and exact format of the required value. For a detailed description or for background information on a specific barcode, please refer to the documentation provided by the individual barcode supplier. Note that some barcode readers may require specific

parameters as well.
If it is necessary to concatenate fields to compose the barcode value, edit the expression (or the script) after adding the barcode; see "Adding barcodes to a Dynamic Table" on the facing page.

> **Note:** For barcodes that require a Checksum, the Designer can calculate a Checksum if that isn't provided by your data. In that case the field should contain the required value minus the Checksum. The option to include a calculated Checksum in the barcode value is given on the Properties tab (in step 6), if applicable.

4. Give the barcode an **ID** and/or **class**.
5. Check the option **Absolute** to insert the barcode in an absolute-positioned box inside the <body> of the HTML, but outside other elements. (This option is not available in Email sections and in snippets.) Alternatively, use the **Location** drop-down to select where to insert the barcode.
   - **At cursor position** inserts it where the cursor is located in the template.
   - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the tag.*
   - **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*
   - **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
   - **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph ( ).*
   - **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

   * If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

> **Note:** HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph, the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

6. The settings to make on the **Properties** tab and **Output** tab depend on the selected barcode type. See "Barcode" on page 768.
7. Click **OK** to close the dialog.

In the template the barcode shows up as a gray box. A barcode is always added with the barcode type's default dimensions.

The barcode box has an **expression** containing the name of the selected field. For example: *{{~url~}}*. The value for the barcode is obtained by evaluating the expression (or expressions) in the box.

The expression has a **tilde** character (**~**) inside the double brackets, on both sides. The tildes remove any white space added by the editor when switching between the Design, Source, and Preview modes. They do not remove white space from the value itself.

The barcode properties are written to the `data-params` attribute on the Barcode element in JSON format. (To see this, select the barcode and open the document in the Source view.)

To see the barcode working, toggle to the **Preview** tab in the Workspace.

> **Note:** If expressions are not supported in the current section, a **script** that is associated to the barcode is added instead of an expression.
>
> To enable expressions in a section, right-click the section in the Resources pane, select **Properties** and check the option **Evaluate Handlebars expressions**. In templates made with OL Connect 2022.1 or older this option is unchecked by default.

Adding barcodes to a Dynamic Table

If you want the rows in a Dynamic Table to have a barcode that is based on the value of a detail table field, first create the Dynamic Table. See "Dynamic Table" on page 974.

Then click in the cell that should contain the barcode and follow the same procedure as described in "Adding a barcode" on page 768. The only differences will be:

- In Step 3, when selecting the field that contains the barcode value, you can only choose a field from the detail table the Dynamic Table row is based on.
- In Step 5, the Location option is not available. The barcode can only be inserted **At cursor position**.

Changing a barcode

Barcode value

The value to feed to the barcode generator is either obtained by evaluating the expression or expressions in the barcode box or by running a script that is associated to the barcode.

Expression

When a barcode has just been inserted, it has one expression that contains the name of a data field (e.g. *{{~url~}}*).

To change the expression, click in the box and type the new expression - or multiple expressions, if you want the barcode value to be composed of the values of multiple fields. (See also: "Handlebars expressions" on page 1005.)

In each expression, place a **tilde** character (**~**) inside the double brackets, on both sides. The tildes remove any white space added by the editor when switching between Design, Source, and Preview modes. They do not remove white space from the value itself, which is the result of the expression.

Script

To change the script that is associated with the barcode, double-click it on the **Scripts** pane. When you select more than one field, the script puts the values of the selected fields in one string and passes that to the barcode generator.

> **Tip:** If you don't know which script matches the barcode, click the box that contains the barcode and check the ID and class of that box on the Attributes pane. Then take a look at the Scripts pane: the selector of the associated script is the same as the ID and/or class of the barcode box.
> If a barcode has more than one class name, only the first one is used in the script's selector.

Barcode type and properties

To change the barcode type or the barcode's properties such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the workspace) and select **Barcode** from the shortcut menu.

A barcode's type and size can also be changed on the Attributes pane.

Barcode types

Click the barcode type below for information about its properties.

- Australia Post, see "Australia Post, Japan Post, KIX, USPS IMb" on page 821
- "Aztec Code" on page 774
- "Codabar" on page 776
- Code 11, see "Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5" on page 801
- "Code 39" on page 779
- "Code 39 extended" on page 781
- Code 93, see "Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5" on page 801
- "Code 93 extended" on page 784
- "Code 128" on page 786
- "Data Matrix" on page 788
- EAN-**8**, EAN-**13**, see "UPC-A, UPC-E, EAN-8, EAN-13" on page 819
- "GS1 DataMatrix" on page 791
- "GS1-128" on page 793
- Industrial 2 of 5, see "Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5" on page 801
- Interleaved 2 of 5, see "Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5" on page 801
- Japan Post, see "Australia Post, Japan Post, KIX, USPS IMb" on page 821
- KIX, see "Australia Post, Japan Post, KIX, USPS IMb" on page 821
- Matrix 2 of 5, see "Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5" on page 801
- "MaxiCode" on page 803
- "MSI" on page 804
- "PDF417" on page 805
- "POSTNET" on page 807

- "QR Code" on page 809
- "Royal Mail 2D Mailmark" on page 811
- "Royal Mail 4-State (CBC)" on page 813
- Royal Mail 4-State Mailmark **C**, Royal Mail 4-State Mailmark **L**, see "Royal Mail 4-State Mailmark C, Royal Mail 4-State Mailmark L" on page 816
- "Swiss QR Code" on page 817
- UPC-**A**, UPC-**E**, see "UPC-A, UPC-E, EAN-8, EAN-13" on page 819
- USPS IMb, see "Australia Post, Japan Post, KIX, USPS IMb" on page 821
- "USPS IMpb" on page 822

Australia Post, Japan Post, KIX, USPS IMb

Australia Post, **Japan Post**, **KIX** and **USPS IMb** are some of the types of barcodes that can be added to a template; see "Barcode" on page 768.

In OL Connect versions prior to 2019.2 the USPS IMb barcode was called "OneCode".

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types don't process tilde characters (~) in the data as special characters.

This topic lists the properties and output options of the barcode types Australia Post, Japan Post, KIX and UPS IMb. For the properties of other barcode types, see "Barcode type and properties" on page 771.

> **Tip:** A KIX code can also be part of an address block created by the Address Block wizard. See "Address Block" on page 955.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: The height of the (shorter) bars.
- **Extended bar height**: The total height of the extended bars.

- **Bar width**: The width of the bars.
- **Spacing**: The distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Aztec Code

Aztec Code is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the Aztec Code barcode type. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module size

Enter the size of the square modules in pixels.

Configuration type

Use the drop-down to select the format type used when creating the barcode: only full range format, only compact formats, or any format.

Preferred configuration

Use the drop-down to select the preferred format for the barcode. Note that the barcode generator may choose a different format if the data cannot be represented by the preferred format.

Encoding

Use the drop-down to select the encoding type:

- **Normal** can encode any character but is not very efficient for encoding binary values (above 128).
- **Binary** is to be used only if the data contains many bytes/characters above 128.

Error Correction Level

This option reserves a percentage of the symbol capacity for error correction. The recommended percentage for this type of barcode is 23.

Rune

When set to a value between 0 and 255, an Aztec Rune corresponding to the selected value is created. Set the Rune to -1 to disable this feature.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Codabar

Codabar is one of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that this type of barcode processes tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the Codabar barcode. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Start Char and Stop Char

Use the drop-down to select the start and stop character for the barcode, which defines the encoding mode. Available characters are A, B, C.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, and Matrix 2 of 5 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types process tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the following barcode types :

- Code 11
- Code 93
- Industrial 2 of 5
- Interleaved 2 of 5
- Matrix 2 of 5

For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 39

Code 39 is one of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of this type process tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the barcode types Code 39 and Code 39 extended. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Inter Character Gap

Two adjacent characters are separated by an inter-character gap. A value of 1 means that the separator will have the same length as the width of the narrow bars (in centimeters).

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 39 extended

Code 39 extended is one of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the Code 39 extended barcode type. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Inter Character Gap

Two adjacent characters are separated by an inter-character gap. A value of 1 means that the separator will have the same length as the width of the narrow bars (in centimeters).

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII

character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, and Matrix 2 of 5 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types process tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the following barcode types :

- Code 11
- Code 93
- Industrial 2 of 5
- Interleaved 2 of 5
- Matrix 2 of 5

For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 93 extended

Code 93 extended is one of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the **Code 93 extended** barcode. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 128

Code 128 is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the barcode type Code 128. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Code set

Set of characters to be used:

- **A**: ASCII characters 00 to 95 (0–9, A–Z and control codes), special characters, and FNC 1–4
- **B**: ASCII characters 32 to 127 (0–9, A–Z, a–z), special characters, and FNC 1–4
- **C**: 00–99 (encodes each two digits with one code) and FNC 1

In **Auto** mode, the barcode generator will automatically select the correct encoding mode (set A, B or C) according to the input data.

If the barcode starts as one subset and then switches to another, set **Code set** to whatever the starting set is (not Auto). Enable the **Tilde processing** option (see below). At the position where the subset switches, add the following to the input data:

- ~d199 to switch to subset C
- ~d200 to switch to subset B

- ~d201 to switch to subset A

For example, if a barcode in the format AA123456789 must start as subset B, and then switch to subset C after 1, set the barcode to be: AA1**~d199**23456789.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.

- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Data Matrix

Data Matrix is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Data Matrix barcodes can be read even if they are partially damaged, because of the error correction codes added to them (according to the ECC200 standard).

This topic lists the properties and output options of the Data Matrix barcode. For the properties of other barcode types, see "Barcode type and properties" on page 771.

> **Note: Tilde** characters in the data are processed as special characters (see the Java4less Barcodes Guide to learn what the tilde character can be used for).
> Any tilde that needs to be included in the output must be escaped by adding another tilde (either "~~", or "7E7E" if the Hex input option is enabled). This can be done with a `replace()` call in the Barcode script, after expanding the script.

Barcode properties

Hex Input

For optimized mailings, German Post requires the supplied data for the Data Matrix barcode to be hexadecimal input.
Check this option if your input data is a hexadecimal code. The incoming data will be interpreted as hexadecimal input and decoded to ASCII before passing the string to the Barcode library.

Dots Per Pixels

Type the number of dots per pixel. To optimize barcode quality a Data Matrix symbol should not be printed with dots smaller than 4 pixels.

Encoding

The data represented in the symbol can be compressed using of the following algorithms.

- **ASCII** is used to encode data that mainly contains ascii characters (0-127)
- **C40** is used to encode data that mainly contains numbers and uppercase characters.
- **Text** is used to encode data that mainly contains numbers and lowercase
- **Base256** is used to encode 8 bit values
- **Auto Detect** automatically detects the data content and encodes using the most appropriate method.
- **None** does not use any encoding.

Preferred format

Use the drop-down to select the size of the Data Matrix.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

UPC-A, UPC-E, EAN-8, EAN-13

UPC-A, UPC-E, EAN-8 and EAN-13 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that these barcode types process tilde characters in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the barcode types UPC-A, UPC-E, EAN-8 and EAN-13. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Supplement

UPC-A, UPC-E, EAN-13, and EAN-8 may all include an additional barcode to the right of the main barcode.

- **Type**: The supplement type can be 2-digit (originally used to indicate the edition of a magazine or periodical) or 5-digit (used to indicate the suggested retail price for books). In case this option is set to None, and the data includes digits for the 2 or 5 supplement, the supplement data will be skipped and the additional barcode will not be rendered.

  Note: When the chosen supplement type doesn't match the data, the supplement data will be skipped and the additional barcode will not be rendered.

- **Height Factor**: This is the relative height of the supplement's bars compared to the normal bars.
- **Space Before** : Defines the space between the main symbol and the supplement, in cm.

Human Readable Message

OL Connect shows a human readable text below the barcode, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

GS1 DataMatrix

GS1 DataMatrix is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

About GS1 DataMatrix barcode data

In order to create a GS1 DataMatrix barcode, the barcode data must meet the following requirements.

- The barcode data must start with a leading FNC1 character. Either ~1 or ~d232.
- The GS1 Application Identifiers (AI) must be used for all data.
  The function code ~1 must be used as field separator for variable length AI elements.
- Only ASCII characters should be used.

This topic lists the properties and output options of the barcode type GS1 DataMatrix. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Dots Per Pixels

Type the number of dots per pixel. To optimize barcode quality a Data Matrix symbol should not be printed with dots smaller than 4 pixels.

Preferred format

Use the drop-down to select the size of the Data Matrix.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Note: The following tilde codes are supported in GS1 DataMatrix barcodes:

- ~1 = FNC1 character (for GS1 DataMatrix barcodes)
- ~2 = Structure Append
- ~3 = Reader programming
- ~5 = Macro5
- ~6 = Macro6
- ~7 = ECI expressions

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

GS1-128

GS1-128 is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the barcode type GS1-128. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, and Matrix 2 of 5 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types process tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the following barcode types :

- Code 11
- Code 93
- Industrial 2 of 5
- Interleaved 2 of 5
- Matrix 2 of 5

For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, and Matrix 2 of 5 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types process tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the following barcode types :

- Code 11
- Code 93
- Industrial 2 of 5
- Interleaved 2 of 5
- Matrix 2 of 5

For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Australia Post, Japan Post, KIX, USPS IMb

Australia Post, **Japan Post**, **KIX** and **USPS IMb** are some of the types of barcodes that can be added to a template; see "Barcode" on page 768.

In OL Connect versions prior to 2019.2 the USPS IMb barcode was called "OneCode".

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types don't process tilde characters (~) in the data as special characters.

This topic lists the properties and output options of the barcode types Australia Post, Japan Post, KIX and UPS IMb. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Tip: A KIX code can also be part of an address block created by the Address Block wizard. See "Address Block" on page 955.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: The height of the (shorter) bars.
- **Extended bar height**: The total height of the extended bars.
- **Bar width**: The width of the bars.
- **Spacing**: The distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Australia Post, Japan Post, KIX, USPS IMb

Australia Post, **Japan Post**, **KIX** and **USPS IMb** are some of the types of barcodes that can be added to a template; see "Barcode" on page 768.

In OL Connect versions prior to 2019.2 the USPS IMb barcode was called "OneCode".

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types don't process tilde characters (~) in the data as special characters.

This topic lists the properties and output options of the barcode types Australia Post, Japan Post, KIX and UPS IMb. For the properties of other barcode types, see "Barcode type and properties" on page 771.

> **Tip:** A KIX code can also be part of an address block created by the Address Block wizard. See "Address Block" on page 955.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: The height of the (shorter) bars.
- **Extended bar height**: The total height of the extended bars.
- **Bar width**: The width of the bars.
- **Spacing**: The distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, Matrix 2 of 5

Code 11, Code 93, Industrial 2 of 5, Interleaved 2 of 5, and Matrix 2 of 5 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types process tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the following barcode types :

- Code 11
- Code 93
- Industrial 2 of 5
- Interleaved 2 of 5
- Matrix 2 of 5

For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

MaxiCode

MaxiCode is one of the barcode types that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the MaxiCode barcode. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Resolution

Select the printer output definition for the barcode (200, 300, 400, 500 or 600 dpi).

Mode

OL Connect supports several modes; for an explanation of these modes see the MaxiCode page on Wikipedia.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

MSI

MSI is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that this type of barcode processes tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the barcode type MSI. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Checksum Type

The Checksum type can be MSI10, MSI11, MSI1010 or MSI1110; see https://en.wikipedia.org/wiki/MSI_Barcode.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

PDF417

PDF417 is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the barcode type PDF417. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Mode

Use the drop-down to set the compaction mode:

- **Binary**: allows any byte value to be encoded
- **Text**: allows all printable ASCII characters to be encoded (values from 32 to 126 and some additional control characters)
- **Numeric**: a more efficient mode for encoding numeric data

Error Correction Level

Use the drop-down to select the built-in error correction method based on Reed-Solomon algorithms. The error correction level is adjustable between level 0 (just error detection) and level 8 (maximum error correction). Recommended error correction levels are between level 2 and 5, but the optimal value depends on the amount of data, printing quality of the PDF417 symbol and decoding capabilities of the scanner.

Nr. of Columns

The number of data columns can vary from 3 to 30.

Nr. of Rows

A PDF417 bar code can have anywhere from 3 to 90 rows.

Bar height

Defines the height of the bars for a single row measured in pixels drawn.

Compact

Check this option to use Compact PDF417 instead of the PDF417 barcode. This shortened form of the PDF417 barcode is useful where the space for the symbol is restricted.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.
- **Proportionally**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII

character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

POSTNET

POSTNET is one of the barcode types that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that this type of barcode processes tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the barcode type POSTNET. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Bar height

You can set the height (in cm) of the short bars and the tall bars in the Postnet barcode.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.
- **Proportionally**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.

- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

QR Code

A QR Code is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Instead of using the Script wizard (see "Adding a barcode" on page 768) you could write your own script to get the data for the QR Code.

This topic lists the properties and output options of the QR barcode. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Preferred version

There are 40 sizes of QR codes. Select the preferred version for the QR code.

Auto configure

When this option is checked, the barcode generator overwrites the selected Preferred version (the previous selection) and defines the barcode version based upon on the supplied data.

Module size

Enter the size of the square modules in pixels.

Encoding

This option defines the encoding of the barcode. When **Auto** is selected, the barcode generator determines the encoding based on the supplied string. The other options are:

- **Numeric**: 10 bits per 3 digits, with a maximum of 7089 numerical characters.
- **Alphanumeric**: 11 bits per 2 characters, with a maximum of 4296 alphanumerical characters.

- **Byte**: 8 bits per character, with a maximum of 2953 characters.
- **Kanji**: 13 bits per character, with a maximum of 1817 characters.

Extended Channel Interpretation (ECI)

This setting enables data using character sets other than the default set. Select **Latin-1**, **Latin-2**, **Shift JIS** or **UTF-8**, or select **None** to disable extended channel interpretation.

Correction level

Part of the robustness of QR codes in the physical environment is their ability to sustain 'damage' and continue to function even when a part of the QR code image is obscured, defaced or removed. A higher correction level duplicates data within the QR Code to that effect, making it larger.

FNC

Use the drop-down to either disable FNC or select a FNC option:

- **No**: No FNC option.
- **First**: This mode indicator identifies symbols encoding data formatted according to the UCC/EAN Application Identifiers.
- **Second**: This mode indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. You must then set a value for the Application Indicator property.

Application Indicator

Enter the desired Application Indicator value here; either one character (a-z or A-Z) or a 2-digit number (00 - 99).

This option is only enabled when the FNC property is set to Second.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Barcode Data

QR Codes can have many different types of data, which determines how the code will be generated. On top of just straightforward data, special data structures are used to trigger actions on the device that reads them. This can include contact cards, phone numbers, URLs, emails, etc.

To learn more about the specifications of the different QR code types, see the ZXing Project barcode contents page.

Royal Mail 2D Mailmark

Royal Mail 2D Mailmark is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the barcode type Royal Mail 2D Mailmark. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

The recommendation is to print these barcodes with a module size of 0.5 mm, which equates to 6 dots when printed at 300dpi. The maximum module size for printing is 0.7 mm.

Preferred version

Use the drop-down to select the size of the barcode, in a number of modules. The actual size of the barcode can be 12 mm x 12 mm up to 22.4 mm x 22.4 mm, depending on the preferred version and the module width.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Royal Mail 4-State (CBC)

Royal Mail 4-State (CBC) is one of the types of barcodes that can be added to a template; see "Barcode" on page 768.
In OL Connect versions prior to 2019.2 this barcode was called "Royal Mail 4 State (RM4SCC)".

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that this type of barcode doesn't process tilde characters (~) in the data as special characters.

This topic lists the properties and output options of the barcode type Royal Mail 4-State (CBC). For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: the height of the (shorter) bars.
- **Extended bar height**: the total height of the extended bars.
- **Bar width**: the width of the bars.
- **Spacing**: the distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Add Checksum

When checked, OL Connect will calculate a Checksum character and add that to the result of the Barcode script. If the value to be encoded is longer than 10 digits, a second check character will be calculated.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Royal Mail 4-State Mailmark C, Royal Mail 4-State Mailmark L

Royal Mail 4-State Mailmark C and Royal Mail 4-State Mailmark L are some of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the barcode types Royal Mail 4-State Mailmark C and Royal Mail 4-State Mailmark L.
For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: The height of the (shorter) bars.
- **Extended bar height**: The total height of the extended bars.
- **Bar width**: The width of the bars.
- **Spacing**: The distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Royal Mail 4-State Mailmark C, Royal Mail 4-State Mailmark L

Royal Mail 4-State Mailmark C and Royal Mail 4-State Mailmark L are some of the types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

This topic lists the properties and output options of the barcode types Royal Mail 4-State Mailmark C and Royal Mail 4-State Mailmark L.
For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: The height of the (shorter) bars.
- **Extended bar height**: The total height of the extended bars.
- **Bar width**: The width of the bars.
- **Spacing**: The distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Human Readable Message

When this option is checked, OL Connect shows a human readable text below or above the barcode, as defined using the Text Position, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Swiss QR Code

The Swiss QR Code is a specific sub-set of QR barcode standards and is one of the several types of barcodes that can be added to a template; see "Barcode" on page 768.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Instead of using the Script wizard (see "Adding a barcode" on page 768) you could write your own script to get the data for the QR Code.

This topic lists the properties and output options available for Swiss QR barcodes. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Encoding

This option defines the encoding of the barcode. When **Auto** is selected, the barcode generator determines the encoding based on the supplied string. The other options are:

- **Numeric**: 10 bits per 3 digits, with a maximum of 7089 numerical characters.
- **Alphanumeric**: 11 bits per 2 characters, with a maximum of 4296 alphanumerical characters.
- **Byte**: 8 bits per character, with a maximum of 2953 characters.
- **Kanji**: 13 bits per character, with a maximum of 1817 characters.

FNC

Use the drop-down to either disable FNC or select a FNC option:

- **No**: No FNC option.
- **First**: This mode indicator identifies symbols encoding data formatted according to the UCC/EAN Application Identifiers.
- **Second**: This mode indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. You must then set a value for the Application Indicator property.

Tilde processing

Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.)
The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example

w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Barcode Data

To learn more about the specifications of the Swiss QR code , please see the Swiss Payments Standard page, or download the barcode specification ig-qr-bill-en.pdf.

> **Note:** The Swiss QR code standard dictates that the barcode dimensions must be 46 mm by 46 mm. Please do not change the barcode dimensions in the entry as this will invalidate the barcode.

UPC-A, UPC-E, EAN-8, EAN-13

UPC-A, UPC-E, EAN-8 and EAN-13 are a few of the barcode types that can be added to a template.

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that these barcode types process tilde characters in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the properties and output options of the barcode types UPC-A, UPC-E, EAN-8 and EAN-13. For the properties of other barcode types, see "Barcode type and properties" on page 771.

Barcode properties

Module width

Specifies the width of the narrow bars in centimeters. Changing this value to a higher value will make the barcode bigger when Scale is set to None.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Fit to box**: The barcode is stretched to fit the parent box in both width and height.

Supplement

UPC-A, UPC-E, EAN-13, and EAN-8 may all include an additional barcode to the right of the main barcode.

- **Type**: The supplement type can be 2-digit (originally used to indicate the edition of a magazine or periodical) or 5-digit (used to indicate the suggested retail price for books). In case this option is set to None, and the data includes digits for the 2 or 5 supplement, the supplement data will be skipped and the additional barcode will not be rendered.

  > **Note:** When the chosen supplement type doesn't match the data, the supplement data will be skipped and the additional barcode will not be rendered.

- **Height Factor**: This is the relative height of the supplement's bars compared to the normal bars.
- **Space Before** : Defines the space between the main symbol and the supplement, in cm.

Human Readable Message

OL Connect shows a human readable text below the barcode, using the specified font and font size. The font size is given in points (pt).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example

w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Australia Post, Japan Post, KIX, USPS IMb

Australia Post, **Japan Post**, **KIX** and **USPS IMb** are some of the types of barcodes that can be added to a template; see "Barcode" on page 768.

In OL Connect versions prior to 2019.2 the USPS IMb barcode was called "OneCode".

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that barcodes of these types don't process tilde characters (~) in the data as special characters.

This topic lists the properties and output options of the barcode types Australia Post, Japan Post, KIX and UPS IMb. For the properties of other barcode types, see "Barcode type and properties" on page 771.

> **Tip:** A KIX code can also be part of an address block created by the Address Block wizard. See "Address Block" on page 955.

Barcode properties

Height, width and spacing

The height, width and spacing of the barcode are all measured in pixels (38 dpi).

- **Bar height**: The height of the (shorter) bars.
- **Extended bar height**: The total height of the extended bars.

- **Bar width**: The width of the bars.
- **Spacing**: The distance between the bars.

Scale

Defines if and how the rendered barcode is scaled in relation to the parent element:

- **None**: The barcode is rendered based on the module width.
- **Proportional**: The barcode is stretched up to where it fits either the width or height of the parent box, whichever requires the less stretching.

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

USPS IMpb

US Postal Service Intelligent Mail Package Barcodes (USPS IMpb) are used by the United States postal service for parcel deliveries. They contain parcel destination information along with other information about the delivery. They are one of the barcode types that can be added to a template; see "Barcode" on page 768.
In OL Connect versions prior to 2019.2 this barcode was called "IMPB".

The barcode can be added either using the Barcode toolbar button or through selecting **Insert > Barcode** on the menu; see "Adding a barcode" on page 768.

To change a barcode's properties, such as the scale and color, open the Barcode properties dialog: right-click the barcode (on the **Design** tab in the Workspace) and select the **Barcode...** option in the context menu.

Note that this type of barcode processes tilde characters (~) in the data as special characters. See the Java4less Barcodes Guide to learn what the tilde character can be used for.

This topic lists the output options of the USPS IMpb barcode type. For the properties of other barcode types, see "Barcode type and properties" on page 771.

> **Note:** Application Identifiers in the barcode value need to be surrounded by round brackets, otherwise the barcode will not be generated correctly.
> For example, this value contains both a Postal Code Application Identifier (*420*) and a Channel Application Identifier *93*: *420123459302234567890123456788*. It should be written as follows: ***(420)****12345****(93)****02234567890123456788*.
> More information about Application Identifiers can be found in the barcode specification here: https://postalpro.usps.com/shipping/impb/BarcodePackageIMSpec (8.0 Appendix D).

Barcode output

Color

The **Color** group allows you to choose a different **Barcode** color (instead of black) and **Background** color (instead of white), by typing a hexadecimal color value (see for example w3school's color picker).

When black is a composite of CMYK or RGB values, it may incur a color click charge on some printers. Check the **Output in Grayscale** option to make sure that pure black is used.

Output format

Defines how the barcode is output on the page. There are two possible formats:

- **SVG**: Vector format. This is usually of higher quality, but may take longer to generate. It is not compatible with Email output.
- **PNG**: Binary rasterized format. This has a slightly lower quality than SVG but is usually generated faster. It will display properly in Email output.

Boxes

Boxes are elements that are used to surround other elements, either to style them, to find them, or to place them in specific locations.

Wrapping elements in a box (or in a semantic HTML element) makes it easier to target them in a script or in a style sheet. Place the cursor in the element or select multiple elements.

Then, on the menu, click **Insert > Wrap in Box**. You can now use the wrapper element as a script's or style's `selector`; see "Using the Text Script Wizard" on page 948 and "Styling and formatting" on page 879.

> **Tip:** With the **Copy fit** feature, text can automatically be scaled to the available space in a Box or Div. See "Copy Fit" on page 896.

Positioned Box

A Positioned Box is one that can be freely moved around the page and does not depend on the position of other elements. A positioned box is actually a element that has an **absolute position**; in other words, it has its CSS property `position` set to `absolute` (see also: "Using the CSS position property" on page 900).

Positioned Boxes are suitable for use in Print templates only.

Adding a Positioned Box

To insert a Positioned Box, use the icon on the toolbar.

Or, keep the **Ctrl** key pressed down while dragging an image or a data field into the template. This inserts a Positioned Box with the image or an expression in it. (See: "Adding images" on page 854 and "Variable data in text: expressions" on page 999.)

Moving and resizing a Positioned Box

Positioned Boxes can be moved by **dragging** the borders, and resized using the handles on the sides and the corners. Pressing any arrow key moves the box by 1 pixel in the direction of that key.
Alternatively the size and position can be set on the **Attributes** pane. Note that the size and offset values will be displayed in the default print units as defined in the preferences.

- To move or resize multiple Positioned Boxes at the same time, hold the **Ctrl** key while selecting them. You could either select them in the Design view (the main editor) or in the Outline pane.
- Hold the **Shift** key while resizing a Positioned Box to ensures that the box retains its aspect ratio.
- Hold the **Shift** key while dragging a Positioned Box to move it horizontally or vertically.

Dynamically changing the position

A Positioned Box in a Print section (not in a Master Page) has the following attributes:

- **anchor** defines the page number (starting by 0) the box is placed on
- **offset-x** defines the horizontal position of the box relative to its container
- **offset-y** defines the vertical position of the box relative to its container.

These attributes can be set in a script. The following script dynamically changes the position of a Positioned Box in a Print context by setting the offset-x and offset-y values.

```
results.attr('offset-x','96');
results.attr('offset-y','96');
```

The measurements are in pixels (e.g. 96px = 1in). Note that you do not need to set the units.

> **Note:** Do not set the `top` or `left` property of a Positioned Box in a style sheet. The position of a Positioned Box in a Print context is handled via its attributes to take the page (or Master Page) and page margins into account. Attributes cannot be overwritten from within a style sheet: style sheets specify style properties, not values of attributes.

Styling a positioned box

A Positioned Box can be styled using the **Format > Box** menu item, through the CTRL+M keyboard shortcut, or through CSS; see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881.

For tips on aligning text in a positioned box, see "Aligning text" on page 895.

Inline Box

An Inline Box is one that is placed within the text flow, where other elements (including text) can wrap around it. An inline box is actually a element that is **floating**; in other words, it has its CSS property `float` set to `left`, `right` or `no float`.

Inline Boxes can be used in Print context and in Web pages. It is common to do entire web layouts using the `float` property. In Email templates, it is best to use Tables to position elements.

> **Tip:** In Print sections, an Inline Box can be aligned to the bottom of the last page. See "Align a box to the bottom of the last page" on page 899 for instructions.

Adding an Inline Box

To insert an inline box, use the icon on the toolbar. Inline Boxes can be resized using the handles on the sides and corner. They can be styled using the **Format > Box** menu item,

through the CTRL+M keyboard shortcut or through the CSS files; see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881.

Positioning an Inline Box

Initially an Inline Box will float to the left. Use the Float left, No float, and Float right icons on the toolbar to change the position of an Inline Box within the text.

- The **Float left** button aligns the Inline Box to the left. The text is positioned to the right of it and is wrapped around the box.
- The **Float right** button aligns the Inline Box to the right, with the text wrapped around it to the left.
- The **No float** button positions the Inline Box where it occurs in the text.

It is not possible to move an Inline Box using drag and drop. To move the Inline Box to another position in the text, you have to edit the HTML on the Source tab in the Workspace, moving the element using cut and paste. To open the Source tab, click it (at the bottom of the Workspace) or select **View > Source View**.

Span

The Span element (in HTML code) is used to group inline elements, such as text in a paragraph. A Span doesn't provide any visual change by itself, but it provides a way to target its content in a script or in a style sheet.

To wrap content in a span, select the text and other inline elements and click **Insert > Wrap in Span** on the menu. Give the span an ID if you are going to add a style rule or script for it that is unique to this span; or give the span a class if this span can be targeted by a style or script along with other pieces of content. Now you can use the wrapper's ID or class as a script's or style's **selector**; see "Using the Text Script Wizard" on page 948 and "Styling and formatting" on page 879.

Div

The Div is the element used to create both Positioned Boxes and Inline Boxes. By default, a Div element reacts pretty much like a paragraph (<p>) or an inline box set to 'no float' except that it can be resized directly. Just like Positioned Boxes and Inline Boxes, Div elements can be styled using the **Format > Box** menu item, through the CTRL+M keyboard shortcut or through the CSS files; see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881.

Adding a Div element

To add a Div, select **Insert > Structural Elements > Div** on the menu. For an explanation of the options, see "Inserting an element" on page 765.

HTML tag: div, span

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file. It is possible to edit the source of the HTML file directly in the Designer; see "Editing HTML" on page 762.

In HTML, boxes are elements. Spans are elements. To learn how to change the attributes of elements, see "Attributes" on page 764.

Business graphics

Business graphics are a great way to visualize data in your documents. Connect Designer lets you create three types of grapics: Bar Charts, Line Charts and Pie Charts. The charts are generated by the integrated **amCharts** library (https://www.amcharts.com/javascript-charts/).

This topic explains how to create and edit a business graphic using a **Wizard**, without scripting. The wizard helps you to select data and it shows a preview of both the chart and the supplied data.
Connect supports the majority of features of the amCharts library (version: 3.21.12), whether or not they are directly available in the Chart wizard or via the Chart Properties dialog. As a rule, configuration options that aren't available as an option in a dialog can be set manually on the Source tab of the Chart Properties dialog.

Instead of using a Wizard, you could insert a business graphic **manually** by turning a standard element into a chart object using scripting.

> **Note:** As of OL Connect version 2018.1, the way charts can be compiled and presented has been greatly improved. As a consequence, charts made with a version of Connect prior to 2018.1 may not be converted correctly when opened in a later version. For a list of known backward compatibility issues see: "Business Graphics: Backward Compatibility Issues introduced in 2018.1" on page 136. Also note that charts based on expanded, custom scripts are never converted.

Inserting a business graphic

The Connect Designer has wizards for adding three kinds of graphics: Pie Charts, Bar Charts and Line Charts. These wizards insert the chart and its accompanying script, after

you select the data that should be displayed, their colors and labels, and a few display options.

To insert a business graphic in a template:

1. Place the cursor where the graphic should be added. Also make sure that you have loaded data or a data mapping configuration (see "Loading data" on page 926).
2. Click the toolbar button of the type of chart you want to add, or select **Insert > Business graphic** and choose the chart type.
   The wizard opens with a page on which you can specify the attributes and insertion point of the graph.
3. An **ID** is required. You can change the given ID and, optionally, add a class.
4. - **At cursor position** inserts it where the cursor is located in the template.
   - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the tag.*
   - **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*
   - **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
   - **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph ( ).*
   - **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

   * If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

   Note: HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph,

the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

Check the option **Absolute** if you want the chart to have an absolute position. Leave it unchecked to insert the chart as part of the text, and use the **Location** drop-down to select where in the text the graphic should be inserted:

5. Click **Next** and specify the data for the graph. These settings can also be edited when you open the script, after you've inserted the chart. For an explanation of the options in the wizard, see "Selecting data for a Business Graphic" on the facing page.
6.
   - **Stack Series**: Check to stack the values in one bar, instead of having one bar per value. This is only useful with a chart that is based on a data table.
   - **Rotate**: Check to flip the x and y axes, so that the bars are horizontal starting from the left.
   - **Show Legend**: Check to show the labels of the selected data in a legend. This is only useful with a chart that is based on a data table. In a chart based on data fields, the labels of the fields appear under the x-axis.
   - **Titles**: Type the title that should appear near the horizontal (x) axis and/or the vertical (y) axis. (Note that the axes with their titles are exchanged when you rotate the chart.)

   For a Bar Chart or Line Chart, you can click **Next** and set some basic parameters of the chart: These are just of few of the numerous display options that can be edited via the Chart Properties dialog; see "Enhancing a charts' design" on page 832.
7. Click **Finish** to insert the chart into the template.

The wizard adds the following to the template:

- A Div element. It has a `data-amchart` attribute, as you can see when you select the chart and open the Source view in the Workspace. The `data-amchart` attribute contains settings for how the data is displayed. These settings are made via the chart's properties (see "Enhancing a charts' design" on page 832).
- A script. The script determines which data are displayed in the chart, with which colors and labels. The script can be edited any time; see "Selecting data for a Business Graphic" on the facing page.
  When a preview or output must be generated, the script adds the data for the chart to the `data-amchart` attribute (in a `dataProvider` property). Connect then passes the value of that attribute to the integrated amCharts library.

Rasterizing a business graphic

Business graphics are output as SVG images, but not all email clients may support that format. Before generating Email output, you may want to 'rasterize' it. This converts the business graph into a JPG or PNG image.
To rasterize a business graphic, right-click it and select **Rasterize Options**.
For a JPG image you can set the quality of the resulting image in a percentage.

> **Note:** A business graphic in an Email section is rasterized by default and output as PNG image, because email clients usually don't support SVG images.
> SVG images in an Email section give an error in the Preflight window (see "Doing a Preflight" on page 1075).

Selecting data for a Business Graphic

A Business Graphic script determines which data, colors and labels are used in the chart. Double-click the script on the **Scripts** pane to open it.

> **Tip:** To find the script that provides data for a chart, hover over the names of the script in the Scripts pane. That particular script will highlight the chart in the template and its selector is the same as the ID of the business graphic (preceded by #).

1. Use the **Input Type** drop-down to select the source of the data for the Chart:
   - **Data Fields**: The data originate from data fields in the main record. The chart will have the same number of items for each record.
   - **Data Table**, the data are taken from a detail table that you select using the **Detail Table** drop-down.

     > **Note:** In Pie Charts, only data from the *first* record in the detail table are used.
2. For a Bar or Line Chart based on a detail table, you also have to select a **Category**: one data field (in the detail table) of which the values will appear under the bars or the line; in other words, on the **x** axis.
3. Next to **Values**, select **data fields** with a numerical value.
   The **Toggle non-numeric fields** button filters non-numeric fields from the list. The list will then display only Integer, Float and Currency data fields.
4. Adjust the **label** of each of the selected data fields as needed: click on the label and type the new one.

In Bar and Line Charts with Data Fields as input data, these labels appear under the x-axis of the chart.

Labels are used in the legend. They will be visible when the legend is enabled.

5. Select a **color** for each of the selected fields. Click on the color to open the Edit Color dialog (see "Color Picker" on page 1143).

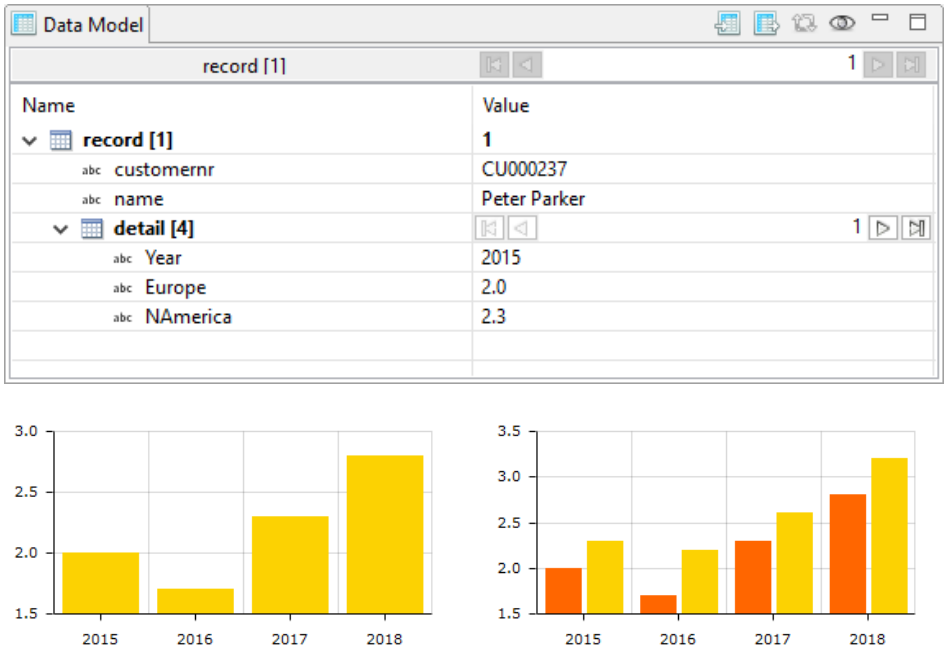
   Colors defined in the Chart Properties dialog override the colors set in the script. To open the Chart Properties dialog, right-click the chart after adding it, and select **Chart...**. See: "Enhancing a charts' design" on the facing page.

6. Use the **Move Up** and **Move Down** buttons to change the order of the fields, which is reflected in the chart.

The Preview shows what the chart looks like with the current settings, whilst the Data Preview shows which data will be passed to the amCharts library.

Preparing a data table

By default, a chart that is based on a detail table shows **one series** ("category") of bars/points **per record**. Each of the bars or points in a series has its own color and label, and represents one data field in that record.
The following image shows a detail table. Both of the Bar charts below are based on that detail table. The difference between the Bar charts is caused by the number of selected data fields (one, or two). The values of the data field 'Year' are used as Category (on the x-axis).



When creating a data mapping configuration (see "Data mapping workflow" on page 368), it is recommended to arrange data in a detail table in such a way that it matches this 'one series per record, one bar/point per data field' approach.

Occasionally you may find that data in a detail table does not match this approach, and that it would be a better fit if the chart had **one series** of bars/points for each selected **detail data field** instead, and one bar/point for each **record**.
If the number of records in a detail table is consistent throughout all records, it is recommended to modify the detail table in the data mapping configuration in order to make it match the default approach.
Otherwise this requires changing the Chart script: open the Chart script, click the Expand button, and alter the code that puts data in the `dataProvider` property.

How to create a Pie Chart that has one slice per detail record is explained in a How-to: Put one slice per detail record in a Pie Chart.

In Connect versions up to 1.8, transposing data was possible via the **Rows are series** option in the Chart script wizard. As of version 2018.1, this option is no longer available.
For a list of backward compatibility issues concerning charts, please see: "Business Graphics: Backward Compatibility Issues introduced in 2018.1" on page 136.

Enhancing a charts' design

Charts inserted with a wizard always have the same layout. The **amCharts** library that is integrated in Connect has loads of options to layout charts differently. Here's how to make use of those options.

Start by opening the Chart Properties dialog. Right-click the chart (in the template, or in the Outline pane) and select **Chart**.

Every tab menu in the Chart Properties dialog, except the last one, gives direct access to a number of layout options. For a description of the options see:

- "Bar Chart Properties dialog" on page 1130
- "Line Chart Properties dialog" on page 1161
- "Pie Chart Properties dialog" on page 1176

Adding and editing properties manually

The last tab menu in the Chart Properties dialog, the **Source** tab, reflects the choices made in the other tabs. More importantly, this tab gives you the possibility to add any amCharts configuration option that is unavailable via the other tab menus.

On the Source tab, all settings are given in JSON. For example:

```
{
"type": "pie",
"legend": {
        "enabled": false
},
"radius": "100",
"innerRadius": "30"
}
```

(The `innerRadius` option, found on the Pie tab in the Pie Chart properties, gives this Pie Chart the shape of a donut.)
Note that only properties that were modified via one of the tab menus are included in the JSON on the Source tab.

To change the chart, you can simply edit the JSON.
For example, adding `"handDrawn": "true"` (at the same level as the "type" property) will distort the lines of a Bar chart or Line chart, producing a hand-drawn effect.
All configuration options can be found in the amCharts **API** documentation: https://docs.amcharts.com/3/javascriptcharts/.

It depends on the class to which a property belongs, where in the JSON the property should be added. Chart properties should be added at the highest level; for example:

```
{
"type": "serial",
"rotate": true,
...
}
```

Properties of the Legend (listed here: https://docs.amcharts.com/3/javascriptcharts/AmLegend) should go in the `legend` section in the JSON:

```
...
"legend": {
 "position": "right"
},
```

The Source tab also lets you change properties that *are* available in either the Script Wizard or other tabs of the Chart Properties dialog. You could, for example, set the colors of the bars, lines or slices by adding an array of hexadecimal color values, like this:
`"colors": ["#FF6600", "#FCD202", "#B0DE09", "#0D8ECF"]`

Note that properties defined on the Source tab override those made in the Script Wizard or on other tabs of the Properties dialog.

For inspiration you could use amCharts' **online editor**: https://live.amcharts.com/new/. Properties can be copied directly from the Code tab in the live editor to the Source tab of the Chart properties dialog in Connect.
Note that copying the entire content of the Code tab will also carry over the sample data from the live editor (the `dataProvider` key). These will be overwritten by the chart script in Connect.

Note:

- Properties defined on the Source tab override those made in the Script Wizard or on other tabs of the Properties dialog. The only exception is the `dataProvider` property: this will be overwritten by the chart's script.
- In Connect, the implementation was tested with Pie Charts (amPieChart) and with Bar Charts and Line Charts (amSerialChart). Other variants may or may not work.
- Animations will not work in the output, even if the output is a web page. This is because the chart is generated on the server, not in the browser.

Using themes

The amCharts library supports working with themes. The default themes are: light, dark, black, patterns, and chalk. All except the 'patterns' theme can be used in Connect templates. Here's how to do that.

1. Add the theme to the top of the JSON on the Source tab of the Chart Properties diaog. For example:
   ```
   { "theme": "light",
   ...
   ```
   This setting overrides any color settings made in the Chart Script wizard and on the other tabs of the Chart Properties dialog.
2. The 'light' theme requires no other settings. For the other themes you will have to manually set the background color of the Div element that holds the chart:

A. Switch to the Design mode.

B. Right-click the chart area and select **Box...** from the contextual menu.

C. On the **Background** tab, set the **Color** to:

- #282828 for the 'dark' theme and the 'chalk' theme
- #222222 for the 'black' theme

3. Finally, the 'chalk' theme requires adding a remote stylesheet with this URL: 'https://-fonts.googleapis.com/css?family=Covered+By+Your+Grace' to your template. See "Using a remote style sheet" on page 884.

Settings for labels

By default, if labels don't fit into the column's space or go outside the plot area, the amCharts library will hide labels to improve readability.

To disable this behavior:

1. Set `ShowAllValueLabels` to `true`
2. Set `autoGridCount` to `false`
3. Set `gridCount` equal to the number of data points

To rotate the labels, add `"labelRotation": 90`.

COTG Elements

With the Designer you can create Capture OnTheGo templates. COTG templates are used to generate forms for the Capture OnTheGo mobile application. This topic is about Capture OnTheGo form elements. For more information about the application refer to these websites: Capture OnTheGo and Capture OnTheGo Help Center.

Capture OnTheGo (COTG) elements can only be added within a Form element in a Web context; see "COTG Forms" on page 708. For information about how to add and use COTG Elements, see "Using COTG Elements" on page 729.

It is also possible to add COTG Elements dynamically, to set defaults for COTG elements and to react to custom events that occur when a user interacts with a COTG element. For more information see: "Using the COTG plugin" on page 739 and "Dynamically adding COTG widgets" on page 742.

Barcode Scanner

The Barcode Scanner element adds a button to trigger the device to scan a barcode. A very large variety of barcode types are supported. Once the barcode has been scanned, the data from the barcode will be added to the COTG Form and submitted along with it.

Note: Using the Barcode Scanner element requires the installation of the ZXing Barcode Scanner application on Android devices. The application returns the barcode data after scanning.

Camera

The Camera element adds a group of buttons to capture or select an image. Once the image is selected via the camera or the device's library (aka "gallery"), it is saved within the Form data.

When the form is submitted, the image is sent in a base64-encoded string format. To learn how to add Camera data to a template, see "Adding Camera data to the template" on page 713.

The Camera element has a number of options, of which most can be set in the Design view. These options are described below.
All options, including options that cannot be set via the Design view, can be set via the Source view or in code; see "Changing default settings for widgets" on page 741.

Buttons

By default, the Camera element adds three buttons to the form:

- **Take now**: Opens the device's default Camera application to take a picture using the device's camera. Capture OnTheGo has no impact on the device's applications, so the features available (quality, orientation, filters) are dependent on the device itself. You can, however, set the format, quality and scaling for images that are submitted by the Camera element, as explained below.
- **Library**: Opens the device's default library or gallery application to select a single image that is then saved in the form data. The accessible images and navigation depend on the device itself.
- **Clear**: Removes any existing image data from the Camera element.

To omit the Take now or Library button, edit the Camera element's properties: right-click the Camera element after adding it to the form, select **Camera properties** and then use the **Source** drop-down to select which buttons will be available: Take, Pick from library, or both.

Annotations

Annotations can make a picture much more informative: an arrow, showing in which direction a car was driving; a circle, where the car has been damaged... To allow the user to make annotations, right-click the Camera element after adding it to the form, select **Camera properties**, and then check **Allow annotations**.
Clicking (or rather, touching) the image will bring up the annotation dialog. Annotations can be made in a Marker (semi-transparent) or Pencil (solid) style, in different colors and with different widths.
Annotations are submitted in SVG format by a hidden input added to the Camera element. The name of that input is the ID of the Camera element, followed by "-note-data", for example **camera1-note-data**.

Cropping/editing/deskewing

To allow the user to crop, edit and deskew the image after taking or selecting it, select **Camera properties**, and then check **Edit Image** and/or **Allow Deskew**. Which editing options the user actually gets and how they are presented to the user depends on the operating system of the device. On an Android device for example, the user may be able only to crop the image, while the user of an iOS device may select part of the image and rotate that selection.

Image format

You can set the format, quality and scaling for images that are submitted by the Camera element. Right-click the Camera element after adding it to the form, select **Camera properties** and edit the Image properties:

- **Format**: The image format can be PNG or JPG.
- **Quality**: Set the compression in a percentage.
- **Scale Image**: Check this option to enable image scaling. Then set the maximum width and height of images before they are sent to the server. Note that only the smallest of these is applied and the size ratio is always maintained.

Time stamp

A time stamp can be added to each picture taken. Right-click the Camera element after adding it to the form, select **Camera properties**, and then check **Add Time Stamp**.
The time stamp will be added to the bottom left of the picture, with medium font size, and long date format (for example: 6/15/2009 1:45 PM). These settings can only be changed via the Source tab or in code; see "Changing default settings for widgets" on page 741 and the Capture OnTheGo API: "Camera" on page 752.

Note: The Time stamp feature doesn't work in versions of the app prior to 10.6.

How to use the captured or selected image in a template

After a user has submitted the form and the data has been extracted, you may want to display the captured or selected image in a Designer template, for example in a letter or on a web page. To do this:

1. Load the data mapping configuration (or at least the data model).
2. Insert a dummy image in the template.
3. Right-click the dummy image and select **Dynamic Image**. The Text Script Wizard appears.
4. Under **Field** select the field that contains the base64-encoded string. The script puts the given string in the source (**src**) attribute of the image (**<img>**).

Instead of using the Text Script Wizard, you could also write a script yourself; see "Writing your own scripts" on page 1064.

Date and Formatted Date

The Date element and the Formatted Date element display the current date on the device when the form is first opened. When the element is touched, a date selector appears so the user can modify this date. The Formatted Date element displays dates in a format that depends on the locale of the device on which the user is viewing the form. A Date Element displays dates in the ISO 8601 format: YYYY-MM-DD.

When the form is submitted, the date data is sent as plain text. A Formatted Date element submits the date in two formats: in the format that depends on the device's regional and language settings and in the ISO format mentioned above (using a hidden field). A Date element sends the date in the ISO format only.

Device Info

The Device Info Element adds a field that contains some information about the device (phone or tablet) that is submitting the COTG Form. This includes the device's type (Android or iOS), operating system version, device model and its UUID (Universally Unique Identifier). This information can be useful for both troubleshooting, if errors occur on specific device types for example, as well as for security validation: it is possible to maintain a list of device UUIDs that are allowed access, to prevent unauthorized use even if someone has a user name and password to a repository.

Document ID

The Document ID element retrieves the Document ID of the form currently viewed by the app. You could put the Document ID in a hidden input, so that when the form is submitted, the Document ID is submitted as well. A Document ID can be used on the server side to check (in the Connect database) if the data has already been submitted.

Fields Table

The Fields Table element adds a table with two rows, a Delete button at the end of the first row and an Add button at the end of the second row. Inside the rows you can put whatever elements you need. The user can click (or rather, touch) the Add button to add a row to the table. The new row will contain the same elements as the first row.
The names of all elements in the first row will be extended with __0, while the names of the elements in the second row will be extended with __1, etc. (This means that the submitted data can be grouped; see "Grouping data using arrays" on page 701.)
Nested tables - i.e. Fields Tables used within a Fields Table - are supported as of version 2021.1. They follow the same naming pattern, for example: `level1[row_0][level2][row_1][name]`.

Geolocation

The Geolocation Element adds a button to read the device's current GPS coordinates and save them in a form field. When the button is pressed, the GPS coordinates are requested and saved. When the form is submitted, the Geolocation data is sent in plain text.

High accuracy

By default, devices attempt to retrieve a position using network-based methods. To tell the framework to use more accurate methods, such as satellite positioning, the High Accuracy setting has to be enabled on the Geolocation element.
To make this setting, right-click the Geolocation element (or select it on the Outline pane) after adding it to the form, select **Geolocation properties** and check the **High Accuracy** option.

> **Note:** The Geolocation element has several options, of which only one can be set via the user interface. All options, including those that cannot be set in Design view, can be set via the `data-params` attribute in the HTML, or in code; see "Changing default settings for widgets" on page 741.

Image & Annotation

The Image & Annotation element is meant to be used with an image that needs input from the user. When inserting an Image & Annotation element you have to select the image. The

user can simply click (or rather, touch) the image to bring up the annotation dialog. Annotations can be made in a Marker (semi-transparent) or Pencil (solid) style, in different colors and with different widths.
Annotations are submitted in SVG format by a hidden input. The name of that input is the ID of the Image & Annotation element, followed by "-note-data", for example **image_annotation1-note-data**.

Locale

The Locale Element does not have a UI element in the form. Inserting it adds a hidden input field that will contain the device's set locale when the form is submitted. This data is sent in plain text format and is available when processing the form data. The format is defined by the device.

Repository ID

The Repository ID element retrieves the repository ID (read only) from the app based on the currently logged on COTG user. You could put the Repository ID in a hidden input, so that when the form is submitted, the Repository ID is submitted as well. This information can be used on the server side to take specific actions, such as sending properly branded emails.

Signature

The Signature Element adds a signature box to a COTG form. These signatures are filled in via touch input, either with a finger or capacitive pen. Touching the signature box opens up a fullscreen box used to sign (generally more useful in Landscape mode depending on the device); after confirming, the dialog saves the data into the Form.
Signature data is transmitted in SVG plain text format. This type of data can be stored in a data field in a Data Mapping and dragged from the Data Model into a template as is. In Preview mode it will be displayed as an image because the Designer, just like web browsers, knows how to display this kind of data.

> **Note:** The Signature data returned by the COTG app (as of Android 10.6.0, iOS 10.6.0, and Windows 6.0) is formatted so that the signature will automatically be scaled to fit in the containing box in a template. With previous versions of the app, the format of returned signatures could vary.

Time and Formatted Time

The Time element and the Formatted Time element display the current time on the device when the form is first opened. When the element is touched, a time selector appears so the user can modify this time. The Formatted Time element displays times in a format that

depends on the locale of the device on which the user is viewing the form. A Time Element displays dates in the ISO 8601 format: HH:MM.

When the form is submitted, the time data is sent as plain text. A Formatted Time element submits the time in both the ISO format mentioned above and in the format that depends on the device's regional and language settings. A Time element sends the time in the ISO format only.

User Account

The User Account Element adds a read only field that contains the Capture OnTheGo user account (an email address) that was used to submit the form to the server. This is useful if a form is available to many different users, to detect who submitted it.
If desired the field can be hidden; it will still be submitted.

Date

The Date element inserts the current system date, optionally making it dynamic so that it updates whenever the template is viewed or produces output.

Adding a date

To add a Date element, use the **Insert > Date** option in the menus. A dialog appears with the following controls:

- **Language**: Use the drop-down to select which language the date should be displayed in.
- **Update Automatically**: Check to update the date automatically when the template is viewed or produces output. If the section is set to evaluate Handlebars expressions, an expression is inserted in the text (see "Date: today" on page 1011). Otherwise a placeholder is inserted in the template and a script is created to update it automatically.

  > **Note:** To make the software insert expressions instead of placeholders, right-click the section in the **Resources** pane, select **Properties** and check the option **Evaluate Handlebars expressions**.

  If the Update Automatically option is not checked, a static text with the current date is inserted.
- **Available Formats**: Select the date/time format in which to display the date.

Click OK to insert the date or Cancel to close the dialog.

Tip: If you are looking to add a date that originates from a record set, to a template, see: "Variable data in the text" on page 924. To insert a date you could use either the drag and drop method or the Wizard.

Changing the date

Once inserted, a date can be modified directly in the template (if it does not update automatically) or through the **date** script (if it does update automatically). To modify the date in the script:

1. Double-click the **date** script in the **Scripts** pane.
2. Between the round brackets after Date, enter the desired date in the following order: year, month, day, and optionally hours, minutes, seconds, milliseconds (see MDN Web Docs - Date object.) When the time is omitted, it defaults to 12:00:00 AM.

Formatting an automatically updating date

In the expression that adds the date, you can change the format by replacing the Helper (e.g. dateShort, dateMedium) with another one. For a list of date format Helpers, see: "Date and time Helpers" on page 1017.

If the date is inserted by a script, it uses the short date format. To change this:

1. Double-click the **date** script in the **Scripts** pane.
2. Delete the first line of the script.
3. On the second line, delete what comes after `format` and change `format` to `formatter` (see "formatter" on page 1486).
4. Now type a dot after `formatter`, press **Ctrl** + **space** and choose one of the functions to format a date and time; see "Date, date/time and time functions" on page 1487.

Note: The Locale, set in the **Edit > Locale** dialog, has an influence on the formatting of a date. The Locale can be the system's locale, a specific locale, or it can depend on the value of a data field or runtime parameter; see "Locale" on page 921.

Forms

Web templates can contain Forms. Capture OnTheGo templates always contain a Form.

Tip: To create a Capture OnTheGo template, preferably use a Template Wizard (see "Capture OnTheGo template wizards" on page 717). The Wizard doesn't just add the

form, it also adds the necessary Capture OnTheGo form elements (see "COTG Elements" on page 835), style sheets and JavaScript files, and extra pre-made elements.

Adding a Form

This procedure describes how to add a Form element to an existing Web context.

1. On the **Resources** pane, expand the **Web** context and double-click a Web page to open it.
2. To use the Form Wizard, select the **Insert > Form Wizard** menu option. The Form Wizard adds a Form to the Web page including the specified fields.
   Alternatively, you can select **Insert > Form** on the menu to open a dialog that lets you set the Form's properties, validation method and location, but doesn't allow you to specify fields. If you choose this method, skip step 8 and 9 of this procedure and add fields after inserting the Form (see "Adding elements to a Form" on page 698).
3. Add an **ID** and/or a **class**. ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).
4. In the **Action** field, enter the URL where the form data should be sent. The URL should be a server-side script that can accept form data. The **action** of a Capture OnTheGo form should specify the Workflow HTTP Server Input task that receives and handles the submitted data. The action will look like this: **http://127.0.0.1:8080/action** (8080 is Workflow's default port number; 'action' should be replaced by the HTTP action of that particular HTTP Server Input task).The **method** of a Capture OnTheGo form should be **POST** to ensure that it doesn't hit a data limit when submitting the form. The GET method adds the data to the URL, and the length of a URL is limited to 2048 characters. Especially forms containing one or more Camera inputs may produce a voluminous data stream that doesn't fit in the URL. GET also leaves data trails in log files, which raises privacy concerns. Therefore POST is the preferred method to use.
5. Using the the **Method** drop-down, select whether the form should be sent using the GET or POST method.
6. Using the next drop-down, select the form's Encryption Type (**enctype**):
   - **application/x-www-form-urlencoded**: Default. All characters are encoded before they are sent. Spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values.

- **multipart/form-data**: No characters are encoded. This value is required when you are using forms that have a file upload control.
- **text/plain**: Spaces are converted to "+" symbols, but no special characters are encoded.

7. Select a **validation** method:
   - The **Browser** validation method leaves it up to the browser to validate the user input. When adding fields to the Form (see the next step) you can only make fields required and set the maximum length as an additional requirement for some fields.
   - Select **JQuery Validation** to validate using JQuery scripts. This allows you to specify stricter requirements per field and type a different message for each field to display to the user if the input is not valid. This method ensures a more consistent validation as it is browser independent. The necessary JQuery files will be added to the JavaScript folder on the Resources pane when the form is inserted.
8. Under **Fields**, click the **Add** button and click on the desired field type to add a field of that type; see "Form Elements" on page 847.

   Note: A Fieldset is not available in the Form Wizard, because a Fieldset itself can contain multiple different fields. Add the Fieldset after inserting the Form; see "Adding elements to a Form" on page 698.
9. Double-click each field in the Fields list and change its settings. For an explanation of the settings, see "Adding elements to a Form" on page 698.
10. The order of the elements in the list under **Fields** determines in which order the elements will be added to the Form. Use the **Move Up** and **Move Down** buttons to change the order of the elements in the list.
11. Use the **Location** drop-down to select where to insert the element.
    - **At cursor position** inserts it where the cursor is located in the template.
    - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the <p> tag.*
    - **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*

- **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
- **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph (</p>).*
- **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

* If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

> **Note:** HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph, the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

12. Close the dialog. Now you can start adding elements to the Form (see "Using Form elements" on page 698, "Form Elements" on page 847, and "COTG Elements" on page 835).

Changing a Form's properties and validation method

Once a Form has been added, you can of course edit its HTML code directly in the Source view of the workspace. Apart from that, there are a number of dialogs to change a Form's properties and validation settings.

Changing a Form's properties

1. Select the form (see "Selecting an element" on page 766).
2. On the **Attributes** pane you can change:
   - The **ID** and/or **class**. ID's and classes are particularly useful with regard to variable data (see "Personalizing content" on page 923) and styling (see "Styling templates with CSS files" on page 881).
   - An **Action**: the URL where the form data should be sent. The URL should be a server-side script that can accept form data (a Workflow process that starts

with a Server Input task, for example).

- A **Method**: this defines whether the form should be sent using the GET or POST method.
- An Encryption Type (**enctype**):
    - **application/x-www-form-urlencoded**: Default. All characters are encoded before they are sent. Spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values.
    - **multipart/form-data**: No characters are encoded. This value is required when you are using forms that have a file upload control.
    - **text/plain**: Spaces are converted to "+" symbols, but no special characters are encoded.

Changing a Form's validation method

In Connect OL Connect, there are two ways in which a Form's input can be validated:

- The **Browser** validation method leaves it up to the browser to validate the user input. When adding fields to the Form (see the next step) you can only make fields required and set the maximum length as an additional requirement for some fields.
- **JQuery Validation** validates input using JQuery scripts. This allows for stricter requirements per field and a different message for each field to display to the user if the input is not valid. This method ensures a more consistent validation, as it is browser independent. The necessary JQuery files will be added to the JavaScript folder on the Resources pane when this option is chosen.

To change a Form's validation method:

1. **Right-click** on the Form element and choose **Validation settings**.
2. Choose a validation type (see above).
3. Double-click each field in the list to edit their validation settings:
    - **Required**: Check if the field is required to submit the form. If a field is required but contains no data, a message will be shown to the user.
    - **Minimum length**: Enter a numerical value for the minimum character length required for this field.
    - **Maximum length**: Enter a numerical value for the minimum character length accepted for this field.

- **Equal to**: Use the drop-down to select another field that is already added to the same Form. The contents of both fields must match for the data to be validated. This is useful for confirmation fields such as for passwords, email addresses etc.

Which of these options are available depends on the validation method of the form: with Browser validation you can only make a field required and set a maximum length.

Changing a Form's validation in HTML

In HTML, the validation method is stored in the data-validation-method attribute of the <form> element, with the value "browser" or "jquery-validation".
A custom message to be shown when validation of a particular Form element has failed, can be stored in the data-custom-message attribute of the Form element, for example: <input id="email1" name="email1" data-custom-message="Enter a valid email address." type="email" required="">

Validation in Connect 1.0.0

In Connect 1.0.0, the validation method of the template was stored using the names "standard" and "custom". Standard has changed to "browser" and custom is now "jquery-validation". When you open a template made with that version of the software, the template will be migrated to use the new attribute values for the data-validation-method attribute of the <form> element. The JavaScript file web-form-validation.js will not be migrated: delete that file and then change the Form's validation method to jQuery Validation, as described above. When you click OK, the new version of the web-form-validation.js file will be added.

Submitting a Form

When a form is submitted, by clicking or touching the Submit button, the **name** and **value** of form elements are sent to the address that is specified in the Form's action (see "Adding a Form" on page 843 or "Changing a Form's properties" on page 845). If the name attribute is omitted, the data of that input field will not be sent at all.

The Form's validation should ensure that the data that the user submits is valid.

Form Elements

This topic lists the elements that can be added to a form in a Web page or in a Capture OnTheGo template and explains how to add them to a Form, set a default value or change their validation. For more information about Forms and Form elements in the Designer, see "Forms" on page 842 and "Using Form elements" on page 698.

For more information about elements in Forms, see MDN Web Docs - HTML Forms.

Fieldset

A fieldset is a group of related elements in a form. The elements don't have to be of the same type. After inserting and selecting the Fieldset (see "Selecting an element" on page 766) you can add elements to it in the same way you add elements to a Form; see "Adding elements to a Form" on page 698.

Text

The Text element is a simple <input> element with the type `text`. It accepts any alphanumerical characters, including special characters. The string is HTML-encoded before it is submitted to the server.

Email

The Email element is a text input field that accepts only email addresses in a valid format.

URL

The URL element is a text input field that accepts only URLs (a web page address) in a valid format.

Password

The Password element is a password field that accepts any alphanumerical characters. When the user types in this field, characters are shown as asterisks only.

Number

The Number element is a text field accepting only numbers.

Date

The Date element is a text field accepting only dates in a valid format.

Text area

The Text area element is a text field accepting multiple paragraphs. You can set a number of rows when adding the Text Area to the Form, and change it on the Attributes pane.

Hidden field

A hidden field can contain specific data used by the server-side script. It is not visible to the user. When adding a Hidden Field you can set the value that will be sent on submit.

Label

A Label element is a text displaying informative text within the form. Labels are non-interactive.
Note that this type of label is not tied to an input element. At the same time you add an

input element, you can add a label to that element; see "Adding elements to a Form" on page 698. To change a label after inserting the Form, simply click it and start typing.

Checkbox

A Checkbox element sends information to the server whether it is checked (true) or not (false). When adding a Checkbox you can set its initial state and its value. The contents of the value property do not appear in the user interface. If a Checkbox is in checked state when the form is submitted, the `name` of the checkbox is sent along with its value.

If a Checkbox is not checked, no information is sent when the form is submitted. Fortunately, there is a workaround to submit the status of an unchecked checkbox; see "Using Form elements" on page 698.

Radio Button Group

A Radio Button Group is not an input element itself. Rather it is a group of Radio Buttons that have the same `submit name`, indicating that they belong to the same group. Multiple Radio Buttons in the same group only accept one option to be selected. Only the value of the selected Radio Button will be sent to the server on submitting the form. If more options are to be allowed at the same time you should use Checkboxes instead.

The option to add a Radio Button Group is only available in the Form Wizard; see "Forms" on page 842. For an explanation of the options in the Radio Button Group dialog, see "Adding elements to a Form" on page 698.

You could also create a Radio Button Group by specifying the same submit name for a number of Radio Buttons when adding them to a Form.

Radio Button

A radio button sends information to the server whether it is selected (true) or not (false). When adding a Radio Button you can set its initial state and its value. The contents of the value property do not appear in the user interface. If a Radio Button is in selected state when the form is submitted, the `name` of the Radio Button is sent along with its value.

If a Radio Button is not checked, no information is sent when the form is submitted. Fortunately, there is a workaround to submit the status of the unchecked radio button, see "Using Form elements" on page 698.

The `submit name` of a Radio Button indicates to which Radio Button Group the Radio Button belongs.

Select

A Select element is a drop-down list with multiple entries from which the user can select only one option. When adding a Select to a Form you can type the options to be given in the

drop-down list and the values related to them. Only the value of the selected option will be sent to the server on submitting the form.
By default, the Select element's first item is selected in the browser.

Button

The Button element is an element that can be clicked. When adding a Button to a Form you can set the button's **type**:

- A **submit** button will validate the form data and if validation is successful, send the data to the provided URL (by the Action specified for the Form; see "Changing a Form's properties" on page 845).
- A **reset** button will reset the form to its original configuration, erasing any information entered and options provided. **Note**: This cannot be undone!
- A button of the type **button** doesn't have a standard function. This is mostly used with a JavaScript to activate a script.

The button's type can also be changed on the Attributes pane.

There may be multiple submit buttons on a Form. If this is the case, specify a different `name` and/or `value` for each of the buttons.

> **Note:** When adding a Button to a Form, you can specify a value, but no name. The Button's ID will be copied to the element's `name` attribute. However, after inserting the Button you can change its name on the Attributes pane.

Hyperlink and mailto link

Links can be added to any template but they only work in electronic output (web pages, email and PDF files). They can be a regular hyperlink pointing to a web page or a mailto link that will open the default email client when clicked.

> **Note:** Hyperlinks - both external and internal - are not preserved when printing multiple pages on the same sheet (see "Imposition options" on page 1456).

HTML element: a

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file. It is possible to edit the source of the HTML file directly in the Designer; see "Editing HTML" on page 762.

The HTML tag of a hyperlink or mailto-link is <a>. This is sometimes called an anchor tag. For a list of attributes, see MDN Web Docs - Anchor element.

Adding a hyperlink or mailto link

1. Select text or an image.

   > **Note:** Although it is possible, it is not advisable to add a Hyperlink to other elements, such as a Paragraph or Div. HTML 4 specifies that hyperlinks and mailto-links may only contain inline elements. Block elements, such as a Div, may not appear inside a link. HTML 5 states that the link "may be wrapped around entire paragraphs, lists, tables, and so forth, even entire sections, so long as there is no interactive content within (e.g. buttons or other links)"; see https://www.w3.org/TR/html5/text-level-semantics.html.

2. Click the **Insert hyperlink** button on the toolbar, or on the menu, select **Format > Hyperlink > Insert**.
3. Select **URL** to create an **external** hyperlink pointing to a web page or an **internal** hyperlink pointing to another page in the document. Or select **Email** to create a **mailto**-link that will open the default email client when clicked.
4. For a **URL**:

   - **URL**:
     - **External** hyperlink: Enter a valid, well-formed URL to link to. It must start with the protocol, such as http:// or https://.

       > **Tip:** You can use a Handlebars expression in this field to make the link dynamic. (See "Variable data in text: expressions" on page 999.)

     - **Internal** hyperlink: Enter an ID, preceded by a hash; for example: `#myID`. The link will point to an element with that ID.

       > **Note:** An internal hyperlink may point to an element in another section. Just make sure that there is only one element in the document with the specified ID.

   - **Target**: Use the drop-down or type in the target for the link. When the target is `_blank` the link will open in a new browser window or tab. See the target attribute (MDN Web Docs).

   For a **mailto link**:

   - **Email**: enter a valid email address that appears by default in the To: field of the email client.

- **Subject**: type a default subject that appears in the Subject: field of the email client.
- **Message**: type a message that appears by default in the Message field of the email client.

Note that all these can be changed within the email client once the link is clicked.

Tip: To quickly change the text of a hyperlink, position the cursor on the link and click `~contents` in the Breadcrumbs. Now you can start typing the new text.

Dynamically inserting or modifying a hyperlink

You may wish to adjust a hyperlink depending on a value in a record that is merged to the template when generating output, for example, to provide a different mailto link for different customers.

Dynamically adding or modifying a hyperlink implies writing a script. For information about scripts, see "Writing your own scripts" on page 1064.

Adding a personalized link

Personalized URLs (pURLs) are links that are tailor-made for a specific purpose, generally for individual clients. Typically, a pURL in a Connect template takes the user to a personalized landing page, for example, to download an invoice or get access to specific products or services. For more information, see "Personalized URL" on page 996.

Images

Images are a powerful ingredient in all of your templates. This topic explains how to add and use them. The supported image formats are:

- BMP
- EPS
- GIF
- JPG/JPEG
- PDF
- PNG
- SVG
- TIF/TIFF

Base64 encoded images are supported; however, they are added differently: set the `src` attribute of an <img> element to the base64 code (see below, "HTML tag: img" on the facing page).

For Email and Web output, **PNG** is the preferred image format.
EPS, PDF, SVG and TIFF images in an Email or Web section are automatically converted to PNG to ensure that they can be seen in the browser or email client.

32-bit BMP files are not supported.

16-bit TIFF images with LZW compression are not supported.

> **Tip:** For better quality of images in PDF output files, use vector images (e.g. SVG, EPS) rather than bitmap images (BMP, JPG, etc.).

Ways to use images

In templates, both **imported** images and **external** images can be used (see "Adding images" on the facing page and "Resources" on page 606). Once added to the content of a template, an image can be resized (see "Resizing an image" on page 906) and alternate text can be linked to it (see "Setting an alternate text" on page 859).

> **Tip:** Using images in an Email template? See "Using images in email campaigns: tips" on page 663.

Dynamic images

Images can be switched dynamically, so that a letter, email or web page can include one image or another, depending on a value in the data set. Read "Dynamic images" on page 972 to find out how to add such switching images.

Background images

Several parts of templates, such as sections and media, and elements such as positioned boxes, can have a background image. Right-click the element and click the **Background** tab to select an image to be used as the element's background image. See "Background color and/or image" on page 908 and "Using a PDF file or other image as background" on page 634.

> **Tip:** Editing PDF files in the Designer is not possible, but when they're used as a section's background, you can add text and other elements, such as a barcode, to them.
> The quickest way to create a Print template based on a PDF file is to right-click the

PDF file in the Windows Explorer and select **Enhance with Connect**. Alternatively, start creating a new Print template with a Wizard, using the PDF-based Print template (see "Creating a Print template with a Wizard" on page 618).
To use a PDF file as background image for an existing section, see "Using a PDF file or other image as background" on page 634.

Filling optional whitespace

Conditional content and dynamic tables, when used in a Print section, may or may not leave an empty space at the bottom of the last page. To fill that space, if there is any, an image or advert can be used as a 'whitespace element'; see "Whitespace elements: using optional space at the end of the last page" on page 643.

HTML tag: img

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file. It is possible to edit the source of the HTML file directly in the Designer; see "Editing HTML" on page 762.

In the section's source file, images are <img> elements. The <img> tag has at least four attributes: `src`, `alt`, `width` and `height`. `src` specifies the URL of the image. `alt` contains the alternate text; see "Setting an alternate text" on page 859.

The value of the attributes can be set on the **Attributes** pane or changed via a script; see "Attributes" on page 764.

Adding images

This section explains the difference between imported and external images and describes a number of ways to add images to a template.

Note: The Connect image cache size is limited to 100MB. This allows most output jobs to run faster. However, if a job requires more than 100MBs of image files, then the Connect image cache size can be increased to cater for such. Please contact OL Support for instructions on how to modify the image memory cache value, if needed. (See also: "Limit of 100MB of image files within a single job" on page 138.)

Note: An image with an unknown file extension is represented by a red cross in the output, but no error is logged unless the image refers to a local file that does not exist on disk.
Image file extensions that the software recognizes are: AFP, BMP, EPS, GIF, JPG/JPEG, PCL, PDF, PNG, PS, SVG, and TIF/TIFF.

Imported or external images

In templates, both **imported** images and **external** images can be used.

Imported images are images that are saved within the template file. To import images into a template and add them to the content, you can use the drag-and-drop method or the Select Image dialog (both are explained below).

External images are either located on a specific website (URL), or in a folder on a hard drive that is accessible from your computer. Note that external images need to be available at the time the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced.
External images are updated (retrieved) when the output is generated. To refresh them at any other time, use the Refresh option in the menu (**View > Refresh**) or the Refresh button at the top of the Workspace.
External images can not be added via the drag-and-drop method. Use the Select Image dialog instead (see below).
If you add an external image via the Source tab or via a script, and the URL doesn't have a file extension, you have to add the `filetype` parameter to the URL manually. Specify the file extension as its value, for example: `?filetype=pdf`, or `&filetype=pdf` if it isn't the first parameter. Note that the ampersand character needs to be encoded as `&`.

For information about referring to images in HTML or in a script, see "Resources" on page 606.

Via drag-and-drop

The drag-and-drop method is a quick way to import one or more images into a template.

1. Look up the image file or image files on your computer using the Windows Explorer.
2. Select the image (or images, using Shift+click or Ctrl+click) and drag the image file from the Explorer to the **Images** folder on the **Resources** pane at the top left.
3. To place an image in the content, drag it from the **Images** folder on the **Resources** pane to the content and drop it. The image will be inserted in the template at the position of the cursor.

   Or, keep the **Ctrl** key pressed down while dragging to insert the image in a "Positioned Box" on page 824.

Via the Select Image dialog

To either import an image into a template or use an external image in a template, the Select Image dialog can be used:

1. Position the cursor in the content where you want the image to be inserted.
2. On the **Insert** menu, click **Image**. Or, click the **Insert Image** button on the toolbar. The Select Image dialog appears.
3. Click **Resources**, **Disk** or **Url**, depending on where the image is located.
    - **Resources** lists the images that are present in the **Images** folder on the **Resources** pane.
    - **Disk** lists image files that reside in a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder (or an image in a folder).
      As an alternative it is possible to enter the path manually. You can give a local path (e.g. C:\Images\Test.jpg) or use the "file" protocol.

      The complete syntax of a fully qualified URL with the "file" protocol is: `file://<host>/<path>`. Note: if the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: `file:///c:/resources/images/image.jpg`.
      If the file is located on another server in your network, the path must contain five slashes after "file:".

      > **Note:** Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file:////myserver/myfolder/file.txt).

    - **Url** lists image files from a specific web address. Select the protocol (**http** or **https**), and then enter a web address (for example, http://www.mysite.com/images/image.jpg).

> **Note:** If a URL doesn't have a file extension, and the option Save with template is **not** selected, the Select Image dialog automatically adds the `filetype` parameter with the file extension as its value (for example: `?filetype=pdf` (if it is the first parameter) or `&filetype=pdf`).
> The `filetype`, `page` and `nopreview` parameters are not sent to the host; they are used internally. Therefore, URLs that rely on one of these parameters cannot be used.

With an external image, you can check the option **Save with template**. If this option is checked, the file will be inserted in the **Images** folder on the **Resources** pane at the top left.
If it isn't saved with the template, the image remains external. Note that external

images need to be available when the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced. External images are updated (retrieved) at the time the output is generated.

Insertion Point group box:

Selects where the image is to be inserted.

- **Absolute** checkbox: Select to make the inserted image an Absolutely positioned one.
  Selecting **Absolute** prevents subsequent selection of **Location** and **Element**.
- **Location** pull down option: Select the Location at which to insert the image.
- **Elements** pull down option: Select the **Element** that the **Location** option is to apply to.
  Not available for **Location** selection of *At cursor position*.

4. If the image is a PDF file that consists of more than one page, select the desired page.

   Note: The number of pages in a PDF file cannot be determined via the HTTP and HTTPS protocols. If you wish to use a page other than page 1 in a remote PDF, check the option **Save with template**; then click OK and reopen the dialog. Next, on the Resources tab, select the PDF, and select a page.

5. Click **Finish**. The image will be inserted at the position of the cursor.

   If an image with the same name already existed in the template, you have the option to overwrite the existing image, use the image that was already in the template, or keep both. In the latter case, the new image is renamed.

Importing images from another template

To import images directly from another template, click **File > Import Resources...** in the menu. See: "Import Resources dialog" on page 1155.

Using one file that contains a collection of images (a 'sprite')

When a template that contains lots of images is merged with a large record set, the many file requests may slow down the process of output generation. One solution is to combine the images into a single image file (an 'image sprite') and display the part that holds the image. This reduces the number of file requests and can improve the output speed significantly.

For an explanation of how to do this, see "Creating and using image sprites" on page 1686.

Base64 encoded images

To insert a base64 encoded image into a template, you need to go to the Source view, insert the <img> tag manually and set its URL - the `src` attribute - to the base64 string, for example: `<img src=""...">`.
If the base64 string is present in one of your data fields, you can set the `src` attribute dynamically. For instructions see "How to use the captured or selected image in a template" on page 838.

Note that the image will not be visible in Design mode, but it will appear in Preview mode.

Moving an image

An image that is added to a section behaves like a character and is part of the text flow. To move it, simply click the image and drag and drop it somewhere else in the text flow. To learn how to wrap text around it, see "Wrapping text around an image" on page 907.
How to make an image stay at a certain position (like any image added to a Master Page) is explained here: "Pulling an image out of the text flow" on page 908. When an image has an 'absolute position' it can be moved around freely: hover over the border of the image to get a move pointer, click that pointer and drag and drop the image somewhere else.

Resizing and styling an image

Images can be **resized** using the handles on the sides and corners, or via the Image Formatting dialog, which opens when you right-click the image and select **Image...**, or select the **Format > Image** menu item.
Images can be styled using the same dialog, or through the CSS files; see "Styling templates with CSS files" on page 881.

A number of issues related to image styling are discussed in a separate topic: "Styling an image" on page 905.

Just like many other elements, images can be given borders and rounded corners, they can have inner and outer margins and they can be rotated. How to do this is described in general formatting topics, such as "Border" on page 910 and "Spacing" on page 922. All general formatting topics are listed under "Styling and formatting" on page 879.

Tip: If placed in a Box, images are by default resized to fit that container. To change this behavior, change the **Fit within frame** setting in the "Image Formatting dialog" on page 1153, the "Attributes pane" on page 1241 or set the image's `object-fit` property in CSS. For examples, see MDN Web Docs - object-fit.
More ways to scale and crop images are described in this how-to: Scale and crop images.

Note: For best performance, it is recommended to resize images outside of the Designer, with image editing software.

Setting an alternate text

Once an image has been inserted in the content of a template, it can have an alternate text.

The alternate text will be shown in emails and on web pages at the position of the image while the image is loading and when the image is not found. On web pages, alternate texts are also used for accessibility.

To set an alternative text, click the image and enter the alternate text in the **Alternate text** field on the **Attributes** pane at the top right.

How to handle missing images

If an image is missing, OL Connect will always display the alternate text; see "Setting an alternate text" above.

If there is no alternate text, a small image showing a white cross on a red background will be displayed to signal that the image is missing.

This default image is only visible in Design and Preview mode, not in the final output.

In addition, a `data-broken-image` attribute is added to the <img> element to specify that the image is broken. This attribute can be used in the selectors of scripts and CSS to apply custom styling or to insert a custom fallback image.

Note: If an image is missing, the alternate text will be visible and any style rules defined for the `data-broken-image` attribute will be applied in Design and Preview mode *and* in the final output.

Tip: For examples see this how-to: Handling missing images. It also explains how to cancel a job or skip items in case of missing images.

Examples

The following style rule specifies a fallback image that is located in the Images folder in the template's Resources folder.

Example: [data-broken-image] { background-image: url('../images/fallback.png'); }

The style rule could be located in the file: context_all_styles.css or in the style sheet for a specific context type. (See "Styling templates with CSS files" on page 881.)

If you use a script to set the fallback, it needs to have the selector: [data-broken-image]. The script could look like this:

Example:

```
let fallbackImgSrc = 'images/fallback.png';
results.attr('src', fallbackImgSrc);
```

Note: Relative file paths are different in scripts and CSS. A stylesheet takes it own location as starting point for relative paths, while a script takes the document location as starting point.

Using a CSS gradient to create an image

CSS gradients are a new type of image added in the CSS3 Image Module. CSS gradients let you display smooth transitions between two or more specified colors, while repeating gradients let you display patterns. This way, using image files for these effects can be avoided, thereby reducing download time and bandwidth usage. In addition, objects with gradients look better when zoomed in a browser, and you can adjust your layout with much more flexibility.

For more information about the various types of CSS gradients and how to use them, see https://developer.mozilla.org/docs/Web/CSS/CSS_Images/Using_CSS_gradients.

Note: When CSS repeating gradients are displayed in a PDF reader, artifacts, like very thin lines may occur. When this happens, try setting the gradient's position a little bit different.

Table

Tables serve two different purposes: they are a way to display data in a tabular format, and they are also a way to position elements on a page.

In HTML email, Tables are the most reliable way to position text and images; see "Designing an Email template" on page 660. In web pages, on the other hand, Inline Boxes are the preferred way to position elements. Tables should only be used to display data in a tabular format, not to position text and images. Tables used in web pages to position elements make those pages less accessible to users with disabilities and to viewers using smaller devices.
In print, Tables can be used for both purposes.

There are two types of Tables: **Standard** Tables which are static in nature, and Dynamic Tables which have a variable number of rows depending on a detail table in the record; see "Dynamic Table" on page 974.

HTML element: table

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file. It is possible to edit the source of the HTML file directly in the Designer; see "Editing HTML" on page 762.

The HTML tag of a Table is <table>. Tables are divided into table rows with the <tr> tag. Table rows are divided into table data with the <td> tag. A table row can also be divided into table headings with the <th> tag.

The tags <thead>, <tbody> and <tfoot> can be used to group the header, body, or footer content in a table, respectively.

For information about HTML tables and a list of attributes, see MDN Web Docs - HTML Table element.

Inserting a Table

1. On the toolbar, click the **Insert Table** button, or on the menu select **Insert > Table > Standard**.
2. Enter the Table's desired attributes:
    - **ID**: a unique identifier for the Table. IDs are used to access the Table from scripts and as CSS selectors for style rules.
    - **Class**: A class identifier for the Table. Classes can be shared between elements and are used to access the Table from scripts and as CSS selectors for style rules.
    - The number of **rows** for the header, body and footer of the Table.
    - The number of **columns**
    - The **width** of the Table.
3. Check the option **Absolute** to give the Table an absolute position, or use the **Location** drop-down to select where to insert the Table:
    - **At cursor position** inserts it where the cursor is located in the template.
    - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the <p> tag.*

- **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*
- **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
- **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph (</p>).*
- **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

* If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

Note: HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph, the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

Note:

- Tables with an absolute position are only useful in Print sections.
- Tables on a Master Page must have an absolute position, unless they are located inside another element with an absolute position.

4. Click **Next** and select the desired table style. The top left actually applies no styling to the table. The style can be easily changed afterwards; see "Styling a Table" on the next page.
5. Uncheck the box **Allow resizing** if the columns should not be resizable in the workspace (in Design mode). This is useful if the column size is determined in the

Source mode or in a style sheet.

6. Click **Finish** to add the Table to the section.

Header and footer

Adding a header or footer

To add a header or footer to an existing Table, right-click the Table and then select **Table > Insert thead** or **Insert tfoot**, on the shortcut menu.

Alternatively, click in one of the cells and select **Insert > Table > Insert thead** or **Insert tfoot**, on the menu.

Deleting a header or footer

To delete a header or footer, simply right-click the header or footer and select **Row > Delete** on the shortcut menu.
If the deleted element was targeted by a script, you will be asked if you want to delete the script as well.

Rows and columns

Adding a row or column

To add a row or column to an existing Table, click in a cell. Then click the black triangle next to the **Insert Row Above** button on the toolbar, and click one of the **Insert** buttons, or select one of the options in the **Insert > Table Elements** menu.

Alternatively, right-click the Table and on the shortcut menu, select **Row > Insert Above** or **Insert Below**, or select **Column > Insert Before** or **Insert After**.

Deleting a row or column

To delete a row or column, simply right-click the row or column and select **Row > Delete** or **Column > Delete** on the shortcut menu. If the deleted row was targeted by a script, you will be asked if you want to delete the script as well.

Styling a Table

The Insert Table wizard lets you select a style for the table.

Tables are styled via CSS: when the wizard adds a Table, the chosen theme's **class** is added to the <table> element, and, if it doesn't exist yet, the **default_table_styles.css** file is added to the resources of the template. This CSS file contains the CSS rules for all table themes.
(See: "Styling templates with CSS files" on page 881.)

To change the theme, you could simply select the table and change its class on the "Attributes pane" on page 1241.
The available theme classes are: table--grid, table--colgrid, table--minimalist, table--dark, table--light, table--striped, table--topbar, table--portfolio. (Note the double dash in the class name.)

The default_table_styles.css file is read-only, but of course you could overwrite styles and create your own theme, using your own **style sheets**.
Regardless, you can change the font, font size and color, the borders, the cell padding (the distance between the edge of the cell and its content), and the background color or image of the table and its cells, via the **Formatting** dialog.

Both approaches are explained in the topic: "Styling a table" on page 902.

Resizing and moving a Table

Resizing a Table

- Select the Table (see "Selecting an element" on page 766) and type the desired width and height under **Geometry** on the **Attributes** pane.
- Select the Table and select **Format > Table**, on the menu. On the **Table** tab, change the **width** and **height** of the Table.
- Click in the Table and drag the handles to resize it. Press the **Shift** key while dragging to scale the Table proportionally.
  This option only works in a Print section, with a Table that has an absolute position and for which resizing is allowed.
  - If the position of the Table isn't absolute, right-click the Table and on the shortcut menu, select **Convert to absolute**. (This option isn't available for Tables on a Master Page, as they must always have an absolute position, or be located inside another element with an absolute position.)
  - Select the Table (see "Selecting an element" on page 766) and then, on the **Attributes** pane, check the option **Allow resizing**.

Moving a Table

The easiest way to move a Table in relation to other content is this:

1. Open the **Outline** pane (next to the Resources pane).
2. Select the Table on the Outline pane.
3. Drag and drop it somewhere else in the outline.

To move a Table with an absolute position, you can also:

- Click in the Table and then drag the border to move the Table.
- Select the Table (see "Selecting an element" on page 766) and type the desired values in the Top and Left fields on the **Attributes** pane.
- Select the Table and select **Format > Table**, on the menu. On the **Table** tab, change the **Positioning** values.

Hiding the border

When using a Table to position other elements, you will want to hide the borders of the Table. To do this, set the width of the border to 0; see "Border" on page 910.

Text and special characters

The vast majority of templates for personalized customer communications contain, of course, text. While the most common text element is a <p> or paragraph, other elements such as Headings (<h1> through <h6>) are also considered text elements. Text elements can be present within other types of elements such as table cells (<td>), boxes ( ), etc.

Adding text

To add text, simply type in the workspace in the middle.

- Press **Enter** to insert a new paragraph.
- Press **Shift+Enter** to insert a line break.

Alternatively, use the **Insert Lorem Ipsum** toolbar button to insert dummy text, or copy-paste text into the template.
Select **Paste as Plain Text** from the Edit menu or the contextual menu to insert the text without any HTML styles or formatting.

Text that precedes or follows the value of a **data field** can be added by the Handlebars Helper Wizard (in the case of expressions; see "Using the Helper Wizard" on page 1003) and the Text Script Wizard (see "Using the Text Script Wizard" on page 948).

Note: It is not possible to open a Word file in the Designer. When you copy text from a Word document, however, basic style characteristics travel with the content to OL Connect Designer. Formatting options like bold, italic and formats like Heading 1, Heading 2 are maintained.

Extra spaces

When you add spaces in Design or Preview mode the editor automatically preserves any extra spaces by converting them to non-breaking spaces (" " in HTML). It does this because in HTML extra spaces are generally removed. Take this into account when you edit the template in Source mode (i.e. in HTML) or add text via a script.

Adding special characters

To add special characters:

1. Position the cursor where the character should be inserted.
2. On the **Insert** menu, point to **Special Characters** click **Symbols**, **Dashes and Spaces**, **Arrows**, or **Geometric Shapes**, and click one of the available special characters.

Adding page numbers

Page numbers can only be used in the Print context. See "Page numbers " on page 644.

HTML element: p, h, li and others

When adding elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file, which you can see and edit on the Source tab (see also: "Editing HTML" on page 762).

In HTML, text can be contained in many different elements: paragraphs, span elements, line items and table cells, for example.

The HTML tag of a paragraph is . The paragraph should be followed by a closing tag: .

A line break looks like this in HTML: <br>.

Only the Source tab requires that you write HTML. In Design or Preview mode the editor adds HTML tags as needed.

Formatting text

Text can be styled, colored, centered, indented etc. It can even be displayed so that it reads from right to left. See "Styling text and paragraphs" on page 892.

In all templates you can use the fonts that are provided with the Designer, as well as imported fonts; see "Fonts" on page 917.

Translating text

OL Connect templates can be **multilingual**. For information about multilingual templates and how to create them, see "Translating templates" on page 1120.

Snippets

A snippet is a small, ready-to-use piece of content in a file. Snippets can be re-used within the same template, in all contexts, sections and scripts.
There are three types of snippets:

- **HTML** snippets can contain any contents that a section can have, such as text, images, variable data, dynamic tables, etc., see "HTML snippets" on page 870.
- **JSON** files contain data; see "JSON snippets" on page 872
- **Handlebars** templates can contain HTML and Handlebars expressions; see "Handlebars templates" on page 1023.

The information in this topic applies to all or at least two types of snippets.

Note: Regarding **styling**, when a snippet is added to different sections or contexts, it is displayed according to the section's or context's style sheet. This means that the same content can look different depending on the styles applied to the section or context, without changing the content.

Tip: It is possible to open and edit any external HTML or JSON file in the Designer: select **File > Open**, select **All files (*.*)** as the file type and then select an HTML or JSON file.

Adding a snippet to the Resources

Here's how to add a snippet to the Resources of a template.

1. Before adding a snippet, import any resource files that are related to the snippet, such as image files and CSS files, into the template file. Drag and drop the files to the corresponding folders (**Images** and **Stylesheets**, respectively) on the **Resources** pane. If you want to use external images, see "Images" on page 852.
2. Drag the file that contains the snippet into the **Snippets** folder on the **Resources** pane. If the added snippet is not UTF-8 encoded you will be asked to select its encoding. The snippet will then be converted and saved in the template as UTF-8 encoded file.

Tip: To **export** a snippet from your template, drag or copy/paste it out of the Snippets folder to a folder on the local hard drive.

Remote snippets

Remote HTML or JSON

A remote snippet is either an HTML or JSON file that is not located within your template file but is hosted on a Content Management System or other external location.

To add a remote snippet:

1. Right-click the **Snippets** folder on the **Resources** pane, and click **New Snippet** > **Remote HTML file** or **Remote JSON file**.
2. Enter a name for the file as it appears in the Snippets folder. This name is shown in the Snippets folder with the **.rhtml** file extension for HTML, or **.rjson** for a JSON file.
3. Enter the URL for the remote resource. This must be a full URL, including the http:// or https:// prefix, domain name, path and file name.

Note: Remote snippets may contain other resources, such as images. There is one limitation though: only **absolute** paths are supported inside remote snippets. Images and other resources referred to with a relative path (for example: images/img.gif) or root-relative path (any path starting with a slash, for example: /base/images/img.gif) won't be available in the template.

Note: Remote snippets are expected to be UTF-8 encoded.

Remote Handlebars partial

While there is no Remote Handlebars snippet option yet, you can use a **remote partial**.

Example: {{+http://localhost:1880/cms/policies/03921-travel.hbs}}

The **concat** helper can be used to create a dynamic path. In the following example, the `policy` field holds the name of a snippet. (See "Using functions in expressions: Helpers" on page 1009.)

Example: {{+ (concat 'http://localhost:1880/cms/policies/' policy)}}

If the name of a partial is stored in one of the template's runtime parameters, it can be accessed in a Handlebars expression as follows. (See "Data in expressions" on page 1005.)

Example: {{+ (concat @parameters.cmsHost policy)}}

Creating a snippet

To create a snippet that is similar to one already in your template, right-click the snippet and select **Duplicate**.

To start creating a snippet from scratch:

1. On the Resources pane, right-click the Snippets folder and select New.
2. Select the type of snippet that you want to create: **HTML** file, **JSON** file or **Handlebars template**.
3. Give the snippet a name.

Tip: It is also possible to create a new HTML snippet from an existing piece of content in the template; see "Using content to create an HTML snippet" on the facing page.

Adding a snippet to a section

Each type of snippets is used differently; see "Adding an HTML snippet to a section" on the facing page, "Using a JSON snippet" on page 872 and "Using Handlebars templates" on page 1025.

Editing a snippet

To edit a snippet, open the snippet file by double-clicking it on the **Resources** pane.

By default, an **HTML snippet** that is being used as shared content can also be modified in a section where it is used. This behaviour can be changed; see "Editing a snippet" on page 871

Renaming a snippet

To rename a snippet, right-click it on the **Resources** pane in the **Snippets** folder and select **Rename**.

If you rename an **HTML snippet** that was inserted into a section as shared content, you need to update the reference to the snippet manually; see "Renaming a snippet" on page 871.

Translating a snippet

Snippets in a multilingual template get translated at the moment they are inserted in the output, if the text is tagged for translation. For more information see "Translating

templates" on page 1120 and "Tagging text in snippets" on page 1123.

HTML snippets

HTML snippets can contain any contents that a section can have, such as text, images, variable data, dynamic tables, etc. Just like other types of snippets, HTML snippets are stored in the **Snippets** folder on the **Resources** pane, but their file name ends in **.html**.

This topic contains information that only applies to HTML snippets. If you want to know how to add an existing (remote) snippet to the resources, or how to create an empty HTML snippet, please see "Snippets" on page 867.

Tip: It is possible to open and edit any external HTML file in the Designer: select **File > Open**, select **All files (*.*)** as the file type and then select an HTML file.

Using content to create an HTML snippet

To turn a part of a letter, email or web page into an **HTML** snippet for reuse in the content of a template:

1. Open the section and select the part or parts that should be saved to a snippet.
2. Right-click the selection, point to **Snippet** and click **Create** to **copy** the selected part to a new snippet, or **Create Shared Content** to create the new snippet and replace the selected part with a **reference** to the new snippet.
3. Give the snippet a name.

Note: Elements in a Print section that have a fixed or absolute position can be anchored to a specific page. It is recommended not to use these in a snippet, since they will appear on the page they are anchored to, even if the snippet is inserted on another page.

Adding an HTML snippet to a section

Drag-and-drop

To add an HTML snippet to the content of a section, drag the snippet from the **Snippets** folder on the **Resources** pane to the desired location in a section.
Check the option **Insert as shared content** to insert a **reference** to the original snippet in the template, rather than a **copy** of the original snippet.

Via a script

In addition to the drag-and-drop method, it is possible, and often useful, to insert a snippet or part of it, using a **script**; for remote snippets this is normal practice. See "Loading a snippet via a script" on page 1091.

> **Note:** If an HTML snippet with **expressions** is loaded through a script, the expressions won't get replaced with values. Use a Handlebars template instead. (See "Handlebars templates" on page 1023).

Editing a snippet

To edit a snippet, open the snippet file by double-clicking it on the **Resources** pane.

Editing shared content

By default, an HTML snippet that is being used as shared content can also be modified in a section where it is used. That way you actually adjust its source. The snippet will be changed at all locations where it is used.

The option to modify shared content snippets from within a section can be turned off:

- In the menu, select **View > Shared Content Editing** to enable or disable editing of all shared content.
- To enable or disable editing on a case by case basis, you can also manually set the `contenteditable` attribute on the Article element of the shared content to `true` or `false`. This overrides the menu setting.

> **Note:** Remote snippets inserted as shared content cannot be changed from within the Designer. Of course, their source file can be edited outside of the Designer. When that happens, the changes will become visible in remote snippets that are inserted as shared content.

Renaming a snippet

To rename a snippet, right-click it on the **Resources** pane in the **Snippets** folder and select **Rename**.

If you rename a snippet that was inserted into a section as shared content, you need to update the reference to the snippet manually:

1. Open the section in which the snippet is used.
2. Switch to Source view.

3. Look for the <article> element in the HTML and replace the old snippet name with the new name in the source attribute.

Translating a snippet

Snippets in a multilingual template get translated at the moment they are inserted in the output, if the text is tagged for translation. For more information see "Translating templates" on page 1120 and "Tagging text in snippets" on page 1123.

JSON snippets

JSON snippets are snippets that contain **JSON data** instead of text. Just like other types of snippets, JSON snippets are stored in the **Snippets** folder on the **Resources** pane, but their file name ends in **.json**.

This topic contains information that only applies to JSON snippets. If you want to know how to add an existing (remote) snippet to the resources, or how to create an empty JSON snippet, please see "Snippets" on page 867.

> **Tip:** It is possible to open and edit any external JSON file in the Designer: select **File > Open**, select **All files (*.*)** as the file type and then select an HTML or JSON file.

Using a JSON snippet

Unlike HTML snippets, JSON Snippets cannot be inserted into the content directly, but they can be accessed via a script using the `loadjson()` function.

To load a JSON snippet in script, use: `loadjson('snippets/nameofthesnippet.json')`.

For example:

```
var json_data = loadjson("snippets/snippet.json");
results.html(json_data.field1);
```

See also: "Writing your own scripts" on page 1064 and "loadjson()" on page 1497.

Handlebars templates

OL Connect has a special kind of snippets called "Handlebars templates". Just like other types of snippets (see "Snippets" on page 867), Handlebars templates are stored in the **Snippets** folder on the **Resources** pane, but their file name ends in **.hbs**.

Handlebars templates look like regular text with embedded Handlebars expressions. A Handlebars expression is some contents enclosed by double curly braces: **{{** ... **}}**.

Example: `<p>Hello {{firstname}}!</p>`

More about expressions and the functions that you can use in expressions can be found in the topics "Handlebars expressions" on page 1005 and "Using functions in expressions: Helpers" on page 1009.

In OL Connect, a Handlebars template may be included in a section (which itself functions as a Handlebars template) or in another Handlebars template.
In the Handlebars library, such Handlebars templates are referred to as ***partials***.

Why use a Handlebars template?

You can use Handlebars expressions in all **sections** (see "Handlebars in OL Connect" on page 998 and "Variable data in text: expressions" on page 999). Nevertheless, in a number of cases it is better or even unavoidable to work with Handlebars snippets.

- With Handlebars templates you can easily load content with expressions into your template dynamically, based on data.
- HTML mixed with Handlebars expressions is not necessarily valid HTML. Processing it with an HTML parser - like the editor in which sections are written - might break both the Handlebars expressions and the HTML. This happens, for example, when `#each` (a Handlebars Helper) is used to create table rows.
  Consider creating a Handlebars template whenever you find yourself switching to Source view to insert expressions and HTML. The Handlebars template editor is a plain text editor, not an HTML parser.

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Creating a Handlebars template

To create a new, empty Handlebars template:

1. On the **Resources** pane, right-click the **Snippets** folder and select **New Snippet**.
2. Select the type of snippet that you want to create: **Handlebars template**.
3. Give the snippet a name.
4. Double-click the new file to open it in the Designer and fill it with HTML text and Handlebars expressions.

Handlebars templates can include other Handlebars templates - see "Using Handlebars templates" below.

Inline partials are supported as well. See the two examples at https://docs.w3cub.com/handlebars/partials#inline-partials.

Tip: To keep an overview you could group the Handlebars templates in a subfolder of the Snippets folder on the Resources pane.

Note: The editor for Handlebars snippets does not have a Design view. HTML with Handlebars expressions is not necessarily valid HTML. Processing it with an HTML parser might break both the Handlebars expressions and the HTML.

Note: There is no need to register a Handlebars template as a partial, like in the original Handlebars library. OL Connect automatically registers all Handlebars templates in the Snippets folder, so that they can be referenced in expressions by name, without path and extension.

Using Handlebars templates

To include a Handlebars template - a 'partial' - in either a section or another Handlebars template, write an expression that refers to it by name:

{{+partialname}}

Note: If the name of the partial contains a space, the space needs to be escaped. Either surround the name with quotes or replace the space with %20. For example, a partial called “par tial.hbs” can be referenced as follows: {{+par%20tial}} or {{+('par tial')}}.

The name can be:

- The name of an .hbs snippet in the template's **Snippets** folder.

  Example: `{{+Snippet1}}`

  {{+snippets/Snippet1.hbs}} would also work, but the path and extension can be omitted.

- The name of an .hbs snippet on **disk**.

  The complete syntax is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in **file:///<path>** - note the **three** forward slashes after **file:**.
  In the remainder of the path you can either use escaped backward slashes:

Example: "file:///C:\\Users\\Administrator\\Desktop\\Partial1.hbs"

or forward slashes:

Example: "file:///C:/Users/Administrator/Desktop/Partial1.hbs"

- The name of a **remote** .hbs snippet (starting with http:// or https://). It could also be a URL that retrieves a Handlebars template from an endpoint.

  Example: {{+http://localhost:1880/policies?p=travel}}

Passing data

You could pass data to the partial using one or more **hash arguments**.

Example: `{{+myPartial test=123}}`

See Hash arguments.

Expressions inside a partial are resolved using the record, unless a **custom context** is passed to the partial.

For example, if a field in your data has an `address` object, that object could be passed as follows:

Example: `{{+myPartial address}}`

In **section clones**, the current detail record could be passed to a partial. The following expression uses the `lookup` helper (see lookup in the Handlebars documentation) to find the current detail record in a detail table called 'policies' by the index which is set in a clone's `meta` property.

Example: {{+myPartial (lookup @root.policies @meta.detail-table-record)}}

See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

When a partial is missing

If there is no partial with the specified name, Handlebars can generate some other content, called **failover content**. Failover content can be specified using block syntax. The expression that calls the partial is the start of a block. An expression with a closing tag and the name of the partial signals the end of the block. The content between these expressions will be used if the called partial is missing.

Example: {{#+ myPartial }} Failover content {{/myPartial}}

The expression in this example will render "Failover content" if no "myPartial" partial is found.

Loading Handlebars templates dynamically

There are several ways to load Handlebars templates - aka 'partials' - dynamically, based on data.

1. You could use a **Block Helper** (see "Using functions in expressions: Helpers" on page 1009).

   Tip: Partials can include other partials. To keep the template clean and clutter-free it's best to put the Block Helper itself in a partial as well.

2. You could dynamically select a partial by using a **sub expression** between parentheses. The sub expression should evaluate to the name of a partial.

   Example: `{{+ (whichPartial) }}`

   - If `whichPartial` is a data field that returns the name of a partial, the template renders the partial whose name is found in the data field.
   - `whichPartial` could also be the name of a function. Note that to use a function in this way, it must be registered as a *Helper*. See "Creating custom Helpers" on page 1018.

3. Handlebars templates can also be loaded dynamically in a **script**. This requires the Handlebars templates to be rendered before they are added to a section. See "Using Handlebars templates in a script" on page 1032.

4. Instead of loading a different partial you could load the same partial and pass different data to it using one or more **hash arguments**. See Hash arguments.

Partials

Partials are normal Handlebars templates that may be called directly by other templates. This topic explains how to work with Handlebars partials in OL Connect.

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use

it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Creating a partial

Since partials are normal Handlebars templates, you can create them just like any other Handlebars template; see "Creating a Handlebars template" on page 1024.

Tip: To keep an overview you could group the partials in a subfolder of the Snippets folder on the Resources pane.

Using partials

To include a partial in either a section or a Handlebars template (i.e. another partial), write an expression that refers to the partial by name:

{{+partialname}}

".hbs" can be omitted from the name of the partial.

Note: If the name of the partial contains a space, the space needs to be escaped. Either surround the name with quotes or replace the space with %20. For example, a partial called “par tial.hbs” can be referenced as follows: {{+par%20tial}} or {{+('par tial')}}.

Partials don't have to be located in the template. They can also be loaded from a disk or from an endpoint (which could facilitate a search in a database or CMS). For instance, you can load a partial from a URL like this:

{{+http://localhost:1880/policies?p=travel}}

To include a partial in a **Handlebars template** (not a section), you can also:

- Drag the partial from the **Snippets** folder on the **Resources** pane to the desired location in the Handlebars template. This copies the contents of the partial into the other.
- Use the expression: {{**>** partialname}}. In sections this does not work, because the greather-than-sign will then be interpreted as HTML.

Custom context

Expressions inside a partial are resolved using the record, unless a **custom context** is passed to the partial. For example, if a field in your data has an `address` object, that object could be passed as follows:

Example: `{{+myPartial address}}`

In **section clones**, the current detail record could be passed to a partial. The following expression uses the `lookup` helper (see lookup in the Handlebars documentation) to find the current detail record in a detail table called 'policies' by the index which is set in the clone's `meta` property.

Example: `{{+myPartial (lookup @root.policies @meta.detail-table-record)}}`

See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Hash arguments

You could pass data to a partial using one or more **hash arguments**.

Example: `{{+myPartial test=123}}`

See Hash arguments.

When a partial is missing

If there is no partial with the specified name, Handlebars can generate some other content, called **failover content**. Failover content can be specified using block syntax. The expression that calls the partial is the start of a block. An expression with a closing tag and the name of the partial signals the end of the block. The content between these expressions will be used if the called partial is missing.

Example: {{#+ myPartial }} Failover content {{/myPartial}}

The expression in this example will render "Failover content" if no "myPartial" partial is found.

Dynamic partials

There are several ways to load partials dynamically, based on data.

1. You could use a **Block Helper** (see "Using functions in expressions: Helpers" on page 1009 and "Using Handlebars templates" on page 1025).

   Tip: To keep the template clean and clutter-free it's best to put the Block Helper itself in a partial.

2. It is also possible to dynamically select a partial by using a **sub expression** between parentheses. The sub expression should evaluate to the name of a partial (with or without the extension: ".hbs").

   Example: `{{+ (whichPartial) }}`

   If `whichPartial` is a data field that returns the name of a partial, the template renders the partial whose name is found in the data field.
   `whichPartial` could also be the name of a function. Note that to use a function in this way, it must be registered as a *Helper*. See "Creating custom Helpers" on page 1018.

3. Finally, instead of loading a different partial you could load the same partial and pass different data to it. This could be either a **custom context** (see "Custom context" on the previous page) or one or more **hash arguments**. See Hash arguments.

Inline partials

Inline partials are supported as well. See the two examples at https://docs.w3cub.com/handlebars/partials#inline-partials.

Styling and formatting

In the Designer you have everything at hand to make your templates look good: colors, fonts and all the tools to position, align and embellish elements in your designs. This topic informs about the ways to style a template.

Local formatting versus style sheets

There are in general two ways to style elements:

- Using **local formatting**. Local formatting means styling an element directly, using a toolbar button or one of the formatting dialogs.
- Using **Cascading Style Sheets** (CSS). Style sheets can determine the appearance of individual elements, as well as the appearance of elements that have the same class or HTML tag.

Whether applied through style sheets or through local formatting, behind the scenes all layout properties in the Designer are CSS properties. When you format an element locally, an **inline** style rule is added to the element.

Note that where local formatting conflicts with a formatting rule for the same element in one of the style sheets, the local formatting rule gets priority; the rule in the style sheet will be ignored.

It is highly recommended to use style sheets in templates right from the start. Even more so if the communications are going to be output to different output channels, or if they consist of different sections (for example, a covering letter followed by a policy). Using CSS with templates allows a consistent look and feel to be applied. A style sheet can change the look of multiple elements, making it unnecessary to format each and every element in the template, if the company's layout preferences change. See "Styling templates with CSS files" on the next page.

Layout properties

Colors and fonts make an important contribution to the look and feel of your template. See "Colors" on page 913 and "Fonts" on page 917.

Text and elements that contain text such as paragraphs and boxes have a number of formatting options that are not available for other elements: font styles and line height, for example. See "Styling text and paragraphs" on page 892.

Boxes and a number of other elements can have a background color and/or background image; see "Background color and/or image" on page 908.

Several elements, such as boxes, images, paragraphs, and tables, can have a border; see "Border" on page 910.

Boxes, images, tables, text and other elements can be rotated; see "Rotating elements" on page 901.

Spacing (padding and margin) helps to position elements relative to other elements in the template; see "Spacing" on page 922.

The best way to position elements depends on the output channel for which the template is intended; see "How to position elements" on page 897.

The locale setting influences how dates, numbers and amounts of money are displayed; see "Locale" on page 921.

Styling templates with CSS files

The Layout toolbar and the Format menu offer many possibilities to style every piece of a template. However, styling every single element, one after another, is a lot of work and, more importantly, can result in a template with a messy mix of styles that isn't easy to maintain and lacks consistent design. Therefore the preferred way to style templates is with CSS files: Cascading Style Sheets. This topic explains how to do that.

Why use CSS files

The basic idea behind CSS is to separate the structure and contents of a (HTML) document as much as possible from the presentation of that document.

Cascading Style Sheets were originally designed for use with web pages, or HTML files. Since every template in the Designer is constructed in HTML, CSS files can also be used in the Designer.
Instead of setting the font size, line height, color etc. for each and every paragraph in the template itself, you can define a layout for all paragraphs, and for all output channels, in a CSS file.

The benefit of this is that you can quickly and easily change the look and feel of all contexts in one template, without having to change the contents. In the event that your company chooses to use another font or to adjust its corporate colors, you only have to change the style sheets.

You are writing HTML

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file.

To see this, toggle to the **Design** tab in the workspace. Click anywhere in the content. Take a look at the *breadcrumbs* at the top of the workspace or select the Outline pane. The breadcrumbs show the HTML tag of the clicked element, as well as the HTML tags of other elements to which the clicked element belongs. The clicked element is at the end of the line.

To edit the HTML text directly:

- In the workspace, toggle to the **Source** tab.

On this tab you can view and edit the content of the template in the form of plain text with HTML tags (note the angle brackets: <>). You may add and edit the text and the HTML tags, classes, ID's and other attributes.

To learn more about HTML, see for example:

- MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction
- W3schools: http://www.w3schools.com/html/default.asp

Many video courses and hands-on courses about HTML (and CSS) are offered on the Internet as well, some for free. Go, for example, to www.codeschool.com or www.codeacademy.com and look for HTML (and CSS) courses.

Emmet

Emmet is a plugin that enables the lightning-fast creation of HTML code though the use of a simple and effective shortcut language. The Emmet functionality is available in the HTML and CSS source editors of Connect Designer. Emmet transforms abbreviations for HTML elements and CSS properties to the respective source code. The expansion of abbreviations is invoked with the **Tab** key.

In the Source tab of the Workspace, you could for example type `div.row`. This is the abbreviation for a element with the class row.
On pressing the Tab key, this abbreviation is transformed to:

```
<div class="row"></div>
```

To quickly enter a table with the ID 'green', one row, and two cells in that row, type:

```
table#green>tr>td*2
```

On pressing the Tab key, this is transformed to:

```
<table id="green">
	<tr>
		<td></td>
		<td></td>
	</tr>
</table>
```

> **Tip:** For more examples, see this how-to: Streamline coding with emmet and custom abbreviations.

All standard abbreviations can be found in Emmet's documentation: Abbreviations.

To learn more about Emmet, please see their website: Emmet.io and the Emmet.io documentation: http://docs.emmet.io/.

Preferences

To change the way Emmet works in the Designer, select **Window > Preferences**, and in the Preferences dialog, select **Emmet**; see "Emmet preferences" on page 1049.

What you can't do with CSS

In OL Connect, it depends on the output channel what can and cannot be done with CSS. CSS can only be used to its full potential with HTML output. As might be expected, animation and transition features don't with Print output.

Included Cascading Style Sheets

When you create a template, a number of style sheets is automatically included:

- One style sheet that applies to all document types: context_all_styles.css.
- One or more style sheets specific to the context (Print, Email, Web). For example, when you create an action email using the Wizard, the files context_htmlemail_styles.ccs and basic_email_action.css are automatically added to the **Stylesheets** folder on the **Resources** pane.
- A style sheet that defines default styles for tables: default.css. It contains the styles that you can choose from when you insert a table via the **Insert** menu or the **Insert table** toolbar button.

  Note: Do not change the default.css style sheet. Use the global style sheet or the style sheet for the relevant context to define your own styles for tables.

Adding CSS files

To add a CSS file of your own, open an Explorer window, **drag** the file to the **Resources** pane and drop it on the **Stylesheets** folder.
In case the CSS file has references to specific images, you can drag/drop or copy/paste those images into the Stylesheets folder as well.

To create a new CSS file, right-click the **Stylesheet** folder on the **Resources** pane and select **New Stylesheet**.

Or, right-click a CSS file already in the Resources and select **Duplicate**.

You can also **import** one or more CSS file from another template using the "Import Resources dialog" on page 1155.
Remember to add or import any files the CSS file refers to.

Note: The order in which style sheets are executed, can affect the actual output. This sequence can be set per section; see "Applying a style sheet to a section" on page 614.

Tip: To export a CSS file from your template, drag or copy/paste it out of the Stylesheets folder to a folder on the local hard drive.

Tip: To open and edit any CSS file in the Designer, select **File > Open**, select **All files (*.*)** as the file type and then select a CSS file.

Using a remote style sheet

A remote style sheet is not located within your template but is located in a network folder or hosted on an external web server (generally called a **CDN**). When generating Web output,

files on an external server are referenced in the web page's header and are served by the remote server, not by the OL Connect Server module.

To add a remote style sheet:

1. Right-click the **Stylesheet** folder on the **Resources** pane, and click **New Remote Stylesheet**.
2. Enter a name for the file as it appears in the Stylesheet resources. For better management, it's best to use the same filename as the remote resource.
3. Enter the **URL** for the remote resource.

   If the file is located on an external web server, this must be a full URL, including the http:// or https:// prefix, domain name, path and file name.

   If the file is located on your network you can click the Browse button or enter the path and file name manually.

   The complete syntax with the "file" protocol is: `file://<host>/<path>`. If the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: file:///c:/resources/images/image.jpg.
   If the file is located on another server in your network, the path must contain five slashes after "file:".

   > **Note:** Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file://///myserver/myfolder/file.txt).

4. Optionally, for a Capture OnTheGo Form, you can check **Use cached Capture OnTheGo resource**, to prevent downloading a remote style sheet again if it has been downloaded before. The file should be available on a publicly accessible location, for example: a folder location on a corporate website, hosted by a CDN (Content Delivery Network) or shared via a Workflow process.

> **Tip:** After adding the remote file, you may right-click it and select **Download Resource**. This allows you to maintain a central file, from which you can quickly download a copy to your template without having to copy & paste.

Note that a local copy of a remote resource is a snapshot; it is not automatically kept in sync with its remote content. You can download the remote resource again to overwrite the local copy with updated content. If you don't want a local copy to be overwritten you should rename it before downloading the remote resource again.

There are a few advantages to resources hosted on a CDN:

- These resources are not served by your server, saving on space, bandwidth and processing.
- Using a popular CDN takes advantage of caching - a client having visited another website using that same CDN will have the file in cache and not re-download it making for faster load times for the client.

Tip: To refresh the remote resources in a Designer view, use the Refresh option in the menu (**View > Refresh**) or the Refresh button at the top of the Workspace.

Using a Sass file

A CSS preprocessor is a CSS extension language that allows you to enhance CSS with code (variables, for example) and then compile it into plain CSS. CSS Preprocessor **Sass** is integrated in Connect.
For more information about Sass, see: https://sass-lang.com/.

For information about working with Sass in the Designer, see "Using a Sass file" on page 891.

Styling your templates with CSS files

Note: Email clients do not read CSS files and some even remove a <style> tag when it is present in the email's header. Nevertheless, CSS files can be used with the Email context in the Designer.
When generating output from the Email context, all CSS rules that apply to the content of the email can be processed and added either to the header of the email or to inline style properties as if local formatting was applied, depending on the Email section properties. See "Email section properties" on page 1199.

Step 1: edit CSS

Editing CSS using a property sheet

1. Select **Edit > Stylesheets**.
2. Click the downward pointing arrow next to **Global** and select the scope that you want to edit styles for, or select the Global CSS file to edit CSS rules that apply to all contexts.
3. Click **New**, or click one of the selectors that are already listed and click **Edit**.
4. Type a CSS selector. This can be:

- A class: `.class`. Class rules apply to all HTML elements with that class. When you create a class, choose a name that indicates what the class is used for, e.g. 'small' for a class that gives elements the font size 'small'. The class name has to be preceded by a dot, e.g. `.small`.

  Tip: Consider using a standard naming convention, such as BEM (Block, Element, Modifier). See https://getbem.com/introduction/.

- An ID: `#id`. An ID is always preceded by #, e.g. `#sender`. When you create an ID, choose a name that indicates what the ID is used for, e.g. #sender would refer to the HTML element with information about the sender.

  Note: Each ID should be unique and can only be used once in each section.

  Note: Do not give an element the ID 'pages' or the class name 'dynamic'. These are reserved words. Using them as an ID or class name leads to undesirable effects.

- An HTML element: p, h1, table, etc. Type the tag name without the angle brackets.
- A combination of HTML elements, separated by a comma. The CSS rule will apply to all HTML elements that are listed in the selector. For instance, a CSS rule with the selector `h1, p` applies to first level headings as well as paragraphs.
- HTML elements inside other HTML elements. For instance, a rule for all paragraphs inside a div element has the selector: `div p`.
- Etcetera. See MDN Web Docs - CSS selectors for more CSS selectors and combinations of CSS selectors.

5. Select the layout options that should apply to selected elements; see "Styling and formatting" on page 879. Note: where a width can be set as a percentage, it is a percentage of the space between the left and right margin.
6. Click the **Apply** button to see how a setting affects any elements that are subject to the selector. (You may have to move and resize the Stylesheet dialog before opening the Edit Rule dialog, in order to be able to see the template that you are working on.) If all is well, click **OK**.
7. In the Stylesheets dialog, click the selector that you chose. All CSS rules for that selector will become visible in a box below the list of selectors.

Editing plain CSS

- Click the button **Advanced** in any property sheet to open a CSS property editor. Type CSS properties at the left and values at the right.
- In the **Resources** pane at the left, double-click the global stylesheet or the stylesheet for the relevant context. The file opens in the workspace in the middle.

A list of all CSS properties and their possible values can be found here: MDN Web Docs - CSS selectors.
Custom CSS properties can be used as well; for an explanation and examples of these see: MDN Web Docs - Custom properties.

Caution: It is recommended not to change the stylesheet that contains OL Connect's default table styles: default.css.

Note: Block comments (/* ... */) are allowed in CSS, but single-line comments (// ...) are not standard and might not work as expected in all browsers.
In the Designer, syntax highlighting doesn't work on single-line comments in CSS files.

Tip: Right-click and choose **Format**, or press Ctrl + Shift + F to "pretty-print" the CSS and make it easier to read and edit.

Step 2: apply CSS to the content

After editing the CSS file(s), make sure that the CSS rules actually apply to one or more elements in the template.

- CSS rules for HTML elements, such as paragraphs, are automatically applied to all elements with the corresponding HTML tag.
- To make a CSS rule for a certain class or ID work for an element in your document, you have to add the class or ID to that HTML element (as described below).

  Note: Classes may be reused throughout one section, but a specific ID should not be used more than once in each section. CSS layout rules for an element with a certain ID only apply to the first element with that ID in each section. If you have two sections inside of a Print context, then you can have the same ID on two sections; they will both be affected by the CSS rules for the element with that ID.
- Style sheets only apply to sections in which they are included; see "Applying a style sheet to a section" below.

Adding a class or ID to an HTML element

1. Select the element (see "Selecting an element" on page 766).
2. On the **Attributes** pane, type the **ID** and/or **class**. Type the ID **without** the preceding # and class names **without** a dot.

   Note: Elements can have multiple classes. Separate the class names with a space (eg. “red small”).

Alternatively, after selecting an element, you can click the **Source** tab at the bottom of the workspace. The selected element will be highlighted in the source. Add the class or classes and/or the ID to the opening tag of the HTML element, for example: `<p class=”intro”>`.

Tip: Would you like to know which elements have a certain class or ID?

Use the **Text Filter** box at the top of the **Resources** pane. It allows to look for text in the source of any text-based files, including sections, master pages, and snippets.

Applying a style sheet to a section

In order for a style sheet to be applied to a specific section, it needs to be included in that section. There are two ways to do this.

Drag & drop a style sheet

1. Click and hold the mouse button on the style sheet on the **Resources** pane.
2. Move the mouse cursor within the **Resources** pane to the section to which the style sheet should be applied.
3. Release the mouse button.

Using the Includes dialog

1. On the Resources pane, right-click the section, then click **Includes**.
2. From the **File types** dropdown, select **Stylesheets**.
3. Choose which CSS files should be applied to this section. The available files are listed at the left. Use the arrow buttons to move the files that should be included to the list at the right.
4. You can also change the order in which the CSS files are read: click one of the included CSS files and use the **Up** and **Down** buttons. Note that moving a style sheet up in the list gives it **less** weight. In case of conflicting rules, style sheets read later will override previous ones.

Note: Style sheets are applied in the order in which they are included in a section. The styles in each following style sheet add up to the styles found in previously read style sheets. When style sheets have a conflicting rule for the same element, class or ID, the **last** style sheet 'wins' and overrides the rule found in the previous style sheet.

Note: Style sheets that are linked to (i.e. included in) a section show a chain icon in the Resources pane (see "Resources pane" on page 1250).

Note: Which style sheets are included can also be set for the **Web context** as a whole: in step 1, right-click the Web context, instead of a section.

How to determine which styles are applied

To see which styles are applied to an element, select the element (see "Selecting an element" on page 766) and take a look at the "Styles pane" on page 1260 that sits next to the Attributes pane.

Tip: Content added by a script isn't visible in Design mode, but is visible and can be inspected in Preview mode.

The Styles pane shows which CSS style rules apply to the currently selected element. Hovering over elements, parent elements, styles and selector parts in the Styles pane highlights the affected elements in the main editor.
A link next to a style rule will open the file where that particular style is defined. This can be either a CSS file or the source file of a section if local formatting was used (see "Styling and formatting" on page 879).

A crossed-out style rule signals that it was overruled by another style rule. This happens when:

- A more specific, and therefore more important rule, is encountered for the same element. See "Using a more specific CSS rule" below to learn more about the specificity of style rules.
- A rule with the same importance is read after the first rule. Not only is the order of the rules in a CSS file important, but also the order in which the style sheets are read. The style sheets that are included with a section are read in the specified order; see "Applying a style sheet to a section" on page 614.

Using a more specific CSS rule

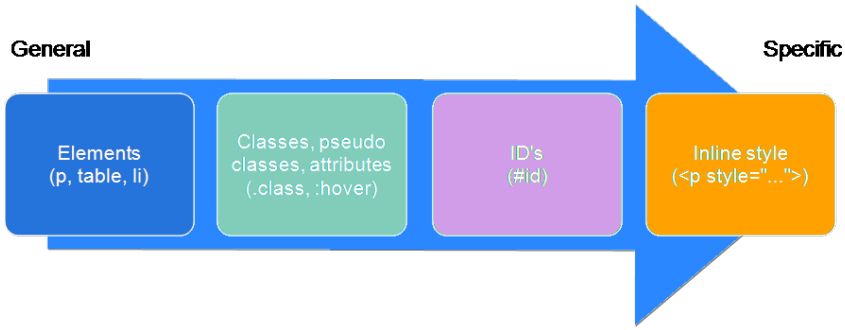
By default, many CSS properties of an HTML element also apply to the elements inside that element. For example, a CSS rule that specifies a certain font-type for a box also applies to

paragraphs in that box. In this example the box is the 'parent' element and the paragraphs are the 'child' elements that inherit the font-type property of the box.

Note: Although the background color property seems to be inherited, it isn't. Most elements are transparent; therefore the background color of the parent element shines through.

To replace inherited style properties, you need to add a more specific CSS rule for that (type of) element. In case of a conflict between a general rule and a more specific rule, the more specific rule will be applied.

The following diagram shows the order of specificity.



Rules for HTML elements (p, table, li etc.) are general rules. Rules for classes, pseudo classes, and elements with a certain attribute (.class, :hover, [target]) are more specific. Rules for elements with a certain ID are even more specific. The most specific are inline styles.

Example:
Assuming that a table has the CSS property "color: red" (which colors text in the cells red), a more specific rule for cells in that table could be, for example:

- A rule for the text color of all table cells (td elements), for example: `td { color: green; }`.
- A rule for the text color of table cells with a certain class, for example `.green { color: green; }`.
- A rule for the text color of a table cell with a certain ID, for example: `#greentext { color: green; }`.
- An inline style rule (local formatting) added to the HTML tag of a particular table cell, for example: `<td style="color: green;">...</td>`.

Each of these rules is more specific than the previous rules. All of these rules are more specific than the rule that applies to the table as a whole.

> **Note:** When `!important` is added to a style rule (e.g. `color: red !important;`), this rule overrides any other style rules, even inline style rules.

Using a Sass file

A CSS preprocessor is a CSS extension language that allows you to enhance CSS with code (variables, for example) and then compile it into plain CSS. CSS Preprocessor **Sass** is integrated in Connect.
For more information about Sass, see: https://sass-lang.com/.

Adding a Sass file

To add a Sass file:

1. Right-click the **Stylesheet** folder on the **Resources** pane, and click **New Stylesheet > SCSS file**.
2. Enter a name for the file as it appears in the Stylesheet resources.
   When the name of Sass file begins with an underscore, it is considered a partial .scss file (e.g. _mySass.scss). Partial files are typically imported in a base .scss file. They may include Sass variables or other directives declared in the base file, and they cannot be compiled.

Compiling a Sass file

A Sass file needs to be compiled into a plain CSS file before it can be applied to a section. To compile it:

1. Open the **Stylesheet** folder on the **Resources** pane.
2. Right-click the .scss file and select **Compile**.

The compiled style sheet will have the same name as the Sass file, but with the extension .css.
Compiled CSS files can be recognized by their first line: /* Compiled by https://sass-lang.com/libsass */.
Compiler options can be set in the Preferences; see "Editing preferences" on page 1044.

Note:

- Re-compiling a .scss file overwrites any manual changes made to the .css file.
- Partial .scss files cannot be compiled.
- Single line comments (//...) are not added to the compiled .css file, whereas multi-line comments (/* ... */) are maintained. Multi-line comments should be added within the brackets of the definition they apply to, to make sure they appear in the correct place in the compiled css.

Styling text and paragraphs

There are numerous ways to format text in a template. You can apply a certain font, make text bold, transform it to uppercase, center it, color it, etc.

This topic explains how to apply local formatting to text. It is recommended though, to format text using style sheets; see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881.

Here are some helpful how-tos related to this topic:

- Bottom-align text.
- Creating custom bullets for lists.

Tip: With the **Copy fit** feature, text can automatically be scaled to the available space in a Box or Div. See "Copy Fit" on page 896.

Formatting text and paragraphs locally

An intuitive way of formatting text locally is by using the toolbar buttons: select some text, or an element that contains text (see: "Selecting an element" on page 766) and click one of the toolbar buttons to make it bold, center it, create a numbered or bulleted list, etc.

Tip: To quickly change a paragraph into a Heading, place the cursor inside of it, or select the paragraph (see: "Selecting an element" on page 766). Then select the appropriate element, either on the **Format** menu, or from the 'Element type' drop-down on the toolbar.

More local formatting options are available in the Formatting dialogs; see below.

Formatting text

To open the Text Formatting dialog, select some text, or an element that contains text, such as a paragraph or a box. Then select **Format > Text**. In the Text Formatting dialog you can set:

- The font, font size, color and background color:
  - **Font**: Select the font used to display text. See also: "Fonts" on page 917. This is equivalent to setting the `font-family` property in CSS.
  - **Font size**. Enter the size in a measure, named size or percentage. This is equivalent to setting the `font-size` property in CSS.
  - **Color**: Specify the color of the text: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    This setting is equivalent to the `color` property in CSS.
  - **Background color**: Specify the background color of the text: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    This setting is equivalent to the `background-color` property in CSS.
- The spacing between letters and words and the way the text is wrapped:
  - **Letter Spacing**: The space between characters in a text in measure or percentage. This is equivalent to the letter-spacing property in CSS.
  - **Word Spacing**: Set the space between each word in a text in measure or percentage. This is equivalent to the `word-spacing` property in CSS.
  - **Whitespace**: Specify how the text wraps. See MDN Web Docs: white-space for details. This is equivalent to the `white-space` property in CSS.
- The style of the text. Check any option to apply the selected style to text within the element. This list shows the CSS property and value for each of the options:

- **Bold**: Sets the `font-weight` to `700`.
- **Italic**: Sets the `font-style` to `italic`.
- **Underline**: Sets the `text-decoration` to `underline`.
- **Strikethrough**: Sets the `text-decoration` to `line-through`.
- **Subscript**: Sets the `vertical-align` to `super`.
- **Superscript**: Sets the `vertical-align` to `sub`.
- **Capitalize**: Sets the `text-transform` to `capitalize`.
- **Uppercase**: Sets the`text-transform` to `uppercase`.
- **Lowercase**: Sets the`text-transform` to `lowercase`.
- **Small-caps**: Sets the `font-variant` to `small-caps`.

Note: All settings in the Text Formatting dialog are in fact CSS style rules. If you selected some text and then change one or more settings, the selected text gets wrapped in a Span element that has an **inline** style tag containing the selected setting (s).
For more information about CSS, see "Styling and formatting" on page 879.

Formatting a paragraph

Through the Paragraph Formatting dialog you can choose a font and style, set the line height and first indent of a paragraph, and specify how to handle page breaks before, in and after the paragraph. It also lets you add spacing and a border; see "Spacing" on page 922 and "Border" on page 910.

To open the Paragraph Formatting dialog, select a paragraph (see: "Selecting an element" on page 766) or place the cursor in a paragraph, and then select **Format > Paragraph**.
For an explanation of all options in this dialog see: "Paragraph Formatting dialog" on page 1172.

Formatting all paragraphs in a Box

Settings that can be made for a paragraph can also be made for a Box that contains one or more paragraphs. The Box settings are applied to all paragraphs in the Box.

To open the Box Formatting dialog, select a Box (see: "Selecting an element" on page 766) or place the cursor in a paragraph in the box, and then select **Format > Box** from the menu.
For an explanation of all options in this dialog see: "Box Formatting dialog" on page 1138.

Removing local formatting from text

Layout buttons and options on the Format menu add **inline** style tags to the text. Style tags can look like this: **...** or like this: **<p style= "color: red;" >**.

Inline style tags have priority over styles defined in a CSS file because they are considered more *specific* (see "Using a more specific CSS rule" on page 890). For example, when a formatting rule in a style sheet colors all paragraphs green, a paragraph with an inline style tag to color it red would still stay red. So, when a rule in a style sheet doesn't seem to work, an inline style tag may be the culprit. In that case you might want to remove the local formatting.

To remove local formatting:

- Select the formatted text and click the toolbar button **Remove Formatting**. Doing this removes inline style tags from the selection.
- Alternatively, click the **Source** tab at the bottom of the workspace (or select **View > Source View**) to manually remove style tags.

  Tip: When you select an element in the template, the **Styles** pane will show which styles are applied to that element. The link behind the style will take you to the place (the Source tab, or a CSS file) where that style is defined.

Aligning text

To align text **horizontally** - to the left, center, or right - there are toolbar buttons (see "Alignment" on page 1269).

There are no buttons to align text vertically, but there are techniques for doing so.

To be able to align a paragraph **vertically**, it must be in a fixed-sized element: a table or a positioned box.

- Table cells have the option to have their content aligned to the top, middle, baseline or bottom. See "Styling a table" on page 902.
- The contents of a positioned box can be aligned vertically using CSS properties. To bottom-align text, you can use the following CSS rules:

  ```
  display: flex;
  flex-direction: column;
  justify-content: flex-end;
  ```

  You can either add these rules via the Box Formatting dialog (click the Advanced button; see "Box Formatting dialog" on page 1138) or directly in a style sheet (see "Styling templates with CSS files" on page 881).

Tip: This approach is especially useful if the text consists of variable data, because with these rules, any empty paragraphs are automatically omitted from the output.

Tip: Here is a how-to that demonstrates this technique: Bottom-align text.

Copy Fit

Copy Fit is a feature to automatically scale text to the available space: the name of a person on a greeting card for example, or the name of a product on a shelf talker.
This feature is only available with Box and Div elements in Print sections.

Activating the Copy Fit feature

After adding a Box or Div element to a Print section (see "Boxes" on page 823), you can activate the Copy Fit feature on that element. Text inside that Box or Div or text in an element inside it, will then be scaled to fit the available space. This is how it's done:

1. Right-click the Box or Div element and click the respective element on the shortcut menu.
   Alternatively, select the element (see "Selecting an element" on page 766) and on the **Format** menu click the respective element.
2. Click the **Content** tab.
3. Check the **Copy Fit** option.
4. Enter the **Min font size and/or Max font size** using a valid font measurement unit (pt, px, in, cm or mm). Do not put a space between the number and the unit.
   - The **minimum font size** is 1pt. The default minimal font size is 4pt.
     When the minimum font size is left **blank**, the font size in Design view becomes the minimum font size. This means that the text can only be made bigger than its initial size.
   - The **maximum font size** is 1048pt. By default this is 48pt.
     When the maximum font size is left **blank**, the font size in Design view becomes the maximum font size, so that the text can only be made smaller than its initial size.
5. When the option **Fit to width only** is checked, no line breaks will be added to the text.
6. Optionally, you can specify a **child** (an element inside the Box or Div) by giving its ID, for example: **#product**, or class, for example: **.product** - note the dot. The Copy Fit

feature will only be applied to this child element.

> **Tip:** To give an element a class or ID, select it and add the class or ID on the Attributes pane. An ID is meant to be used once in each section, while a class can be shared between several elements.

7. Click **Apply** or **OK**. Note that the effect of the Copy Fit feature can only be seen in Preview mode.

If it is impossible to make a text fit within the box with the given minimum and maximum font size, this will be reported as an error during a preflight (see "Testing scripts" on page 1074).

How it works

When the Copy Fit feature is activated, the font size is calculated for the entire Box. Elements inside that Box get a font size relative to the Box. This means that their relative proportions are maintained.

How to position elements

To position elements in relation to each other in a template, you can wrap those elements in a Table or Box (see "Table" on page 860 and "Boxes" on page 823), and/or use the Spacing property of the elements. The Spacing property can also be used to indent elements or create a hanging paragraph or image; see "Spacing" on page 922.

Aligning objects with an absolute position is easy with the Alignment buttons. Guides help to align elements as well; see "Aligning objects" on the facing page.

The top, right, bottom, and left properties specify offsets from the edges of the element's containing block.

Where to use Tables and Boxes

Tables, Positioned Boxes and Inline Boxes can help position elements in relation to other elements. It depends on the context which element is best to use.

In the **Email** context, Tables are the most reliable way to position text and images; see "Designing an Email template" on page 660 and "Table" on page 860.

In the **Web** context, Inline Boxes are the preferred way to position elements; see "Boxes" on page 823.

In Web pages, tables should only be used to display data in a tabular format, not to position text and images. Tables used in web pages to position elements (and often, Positioned

Boxes) make those pages less accessible to users with disabilities and to viewers using smaller devices.

In the **Print** context, Tables can be used to position elements, as well as Positioned Boxes and Inline Boxes; see "Table" on page 860 and "Boxes" on page 823.

Spacing

Boxes, tables, paragraphs and many other elements have a **margin** and **padding**.
The margin is the white space around an element, outside the border. It is used to position an element in relation to the other elements, by putting more space between the element and its surrounding elements.
The padding is the space between an element's content and its border. It is used to position the content of the element inside the border.

To learn how to set an element's spacing properties, see "Spacing" on page 922.

> **Tip:** Use a negative left margin to create a hanging paragraph or image.

Aligning objects

In Print sections, objects with an 'absolute position', such as a Positioned Box, have a fixed position in relation to the page (see also: "Using the CSS position property" on page 900). Objects with an absolute position can be aligned easily:

1. Press the **Ctrl** key and hold it down while clicking on the objects that you want to align. The **last** selected object is the reference object; this object will not be moved.
2. On the menu, select **Format > Align Objects**, or use the respective Toolbar buttons to align the objects.

Align a box to the bottom of the last page

An Inline Box can be aligned to the bottom of the last page by setting its `position` to `absolute` and `bottom` to 0 (zero). This only works in Print sections. Here's how to do that.

1. Insert an Inline Box using the Inline Box icon on the toolbar.
2. Select the box, using the breadcrumbs for example. (See "Selecting an element" on page 766.)
3. Select **Format > Box...** from the menu to open the Box Formatting dialog.
4. On the Box tab, under Positioning,

a. set **Position** to **Absolute**

b. set **Bottom** to **0** (zero).

Note that the same can be achieved with a style sheet. Give the box an ID, for example: bottom-box, and then add the following style rules to your style sheet.

```
#bottom-box {
    position: absolute;
    bottom: 0;
}
```

(See: "Styling templates with CSS files" on page 881).

Guides

Guides are horizontal and vertical lines used to help in designing templates, for example when positioning absolute positions boxes over a PDF background. They can only be used in Print sections.

- Select **View > Guides > Show guides** to show or hide the guides and margins.

To **add** a guide, press the **Insert Horizontal Guide** or **Insert Vertical Guide** buttons on the Toolbar.

To **move** a guide, click and drag it to a new location.
Click the **Shift** key while dragging to make the guide snap to the closest ruler tick.

Double-clicking a guide brings up its Edit dialog where its exact position can be adjusted.

- Select **View > Guides > Lock guides** to lock the guides in their current position.
- Select **View > Guides > Snap to guides** to make Positioned Boxes (and any other objects that have their **position** set to **absolute**) snap to guides when moved within a few pixels of them.

To **delete** a guide, double-click on it and press the **Delete** button.

Using the CSS `position` property

An element can be positioned independently of the text flow by changing its `position` property to `absolute` or to `relative` (that is, relative to the 'parent', its container).
When an element is placed inside another element, such as a Box, changing its `position` property to `absolute` positions the element absolutely inside its parent.
With the `position` property of an element set to `absolute`, the `top` or `bottom` and `left` or `right` properties position the element inside its parent with exact values: pixels (px), centimeters (cm), etc. Negative values are allowed.

Tip: You can quickly change the `position` property of an element in a Print or Web section by right-clicking it and selecting **Convert to Absolute** or **Convert to Inline**.

For an explanation of all values that the `position` property can possibly have, see MDN Web Docs - position.

Where to use it

In Print sections, setting the `position` property to `absolute` can be very useful. It takes the element out of the text flow, so that the element stays where it is on the page. On Master Pages (which are only used in Print sections) elements are always positioned absolutely; if not, they must be located inside an element that has an absolute position.

In Web sections, setting the `position` property to `absolute` may sometimes be useful for elements inside a Div element, but in general, elements should not be positioned absolutely. Designs for the Web should be flexible so that they display nicely on a variety of devices and screen sizes.

In Email sections, do not use this property. Use Tables instead (see "Designing an Email template" on page 660 and "Table" on page 860).

How to use it

In the Formatting dialog the `position` property can often be found on the first tab, under **Positioning**. To open the Formatting dialog, right-click the element and click the respective element on the shortcut menu. Alternatively, select the element (see "Selecting an element" on page 766) and on the **Format** menu click the respective element.

This property isn't present in one of the tab menus of the style rule editor, but you can add it and specify a value after clicking the Advanced button in the style rule editor (see "Styling templates with CSS files" on page 881).

About the CSS `display` property

The `display` property is one of the most important CSS properties for controlling layout. Yet it is unlikely that you will use it often to position elements in a template. In most cases, the initial value of the `display` property for an element in a template will be the right one.

You can use this property in style sheets and scripts to hide certain elements, by setting the value of this property to `none` (`display: none;`). (See "Styling templates with CSS files" on page 881 and "Writing your own scripts" on page 1064.)

Rotating elements

In any type of template, boxes, images, tables, text and other elements can be rotated.

The toolbar buttons **Rotate Clockwise** and **Rotate Counter Clockwise** rotate the element in which the cursor is located 90 degrees at a time.

To rotate an element into another angle position, use the 'angle' CSS property of the element. In most cases, this can be done in the element's Formatting dialog. In other cases, such as with text, you have to enter the CSS property and value manually. Both methods are explained in the following procedure.

1. Right-click the element and click the respective element on the shortcut menu. Alternatively, select the element (see "Selecting an element" on page 766) and click the respective element on the **Format** menu.
2. On the first tab, look for the **angle** property. If it is available, type the number of degrees the element should be rotated. A positive number will rotate the element clockwise, a negative number rotates it counter-clockwise. Skip steps 3 to 6.
   If the angle property is not available, proceed with the following step.
3. Click the **Advanced** button to open the Advanced Formatting dialog.
4. Click in the first blank field under **Property** and type **transform**.
5. Click in the field next to it, under **Value** and type **rotate(**, followed by the number of degrees the element should rotate, and then **deg)**, for example: rotate(20deg). A positive number will rotate the element clockwise, a negative number rotates it counter-clockwise.
6. Close the Advanced Formatting dialog.
7. Close the Formatting dialog, or click the Apply button to see the effect without closing the dialog.

Note: It is also possible to rotate elements by creating a style rule in a style sheet; see "Styling templates with CSS files" on page 881.

Styling a table

Just as other elements, tables can be styled in two ways:

- With **local formatting**. This means styling the table directly, using the Formatting dialog.
- Via **Cascading Style Sheets** (CSS). In a style sheet, style rules are declared for elements with different HTML tags, ID's and classes.

These two methods are described below. See "Styling and formatting" on page 879 for background information about these two methods.

Selecting a table, row or cell

There are several ways to select a table or row:

- Click in the table or row. Then, in the **breadcrumbs** (see "Selecting an element" on page 766) click **table** to select the table, or **tr** to select the row.
- Select the table (table) or row (tr) on the **Outline** pane.
- Right-click a cell and from the shortcut menu, choose **Table > Select** or **Row > Select**.
- Click in a cell and then use the toolbar: click the **Select Table** button or click the black triangle next to that button and then click **Select Table** or **Select Row**.

Selecting one cell is easy: just click in it. Or select the **td** in the breadcrumbs or on the Outline pane.

> **Tip:** Use the Styles pane to see which styles apply to the currently selected table, row or cell.

Styling a column

In HTML, column elements don't exist, only tables, rows and cells.
To make cells look like a column, make sure that they are positioned underneath each other - insert empty cells if needed -, and then style the cells as usual.

Via the Formatting dialog

The Formatting dialog allows you to change the font, font size and color (see "Fonts" on page 917), the borders (see "Border" on page 910), the margins or cell padding (the distance between the edge of the cell and its content, see "Spacing" on page 922), and the background color or image of the table and its cells ("Background color and/or image" on page 908).

To open the Formatting dialog for **one cell** or for the **table as a whole**:

- Click in a cell and choose **Format > Table** or **Format > Table Cell**.
- Right-click it and choose **Cell...** or **Table...** from the shortcut menu.

Note that in this case **Table** styles the table as a whole. When you choose **Table** and change the border, for example, the borders of the cells inside it will not be changed.

To style **all cells** in a table or row at the same time via the Formatting dialog, you have to select the table or row first; see "Selecting a table, row or cell" on the previous page. Next, to open the Formatting dialog, choose **Format > Table Cell**. The settings that you make now will be applied to all cells in the selected row or table.

For information about specific options in the formatting dialogs, see "Table Formatting dialog" on page 1215 and "Table Cell Formatting dialog" on page 1219.

Via a style sheet

Cascading Style Sheets (CSS) offer more ways to style a table and its contents, than the Formatting dialog does. This is especially true for Dynamic Tables. With local formatting, all rows that are added on the fly (in Preview mode and in output) will look exactly the same as the first one. Alternating row colors, for example, in dynamically added rows can only be done via CSS. How to do this is described below.

Another good reason to prefer style sheets over local formatting for Dynamic Tables, is that the output from a Dynamic Table is created slightly faster when it's styled via Cascading Style Sheets than when it's styled with local formatting.

How to use style sheets is explained in another topic; see "Styling templates with CSS files" on page 881.

Note that to make a style rule apply to a **specific** table, row or cell, you have to add an ID or class to that table, row or cell.

A style sheet contains a bunch of style rules for different elements, that are identified via a CSS **selector**. This can be the element's HTML tag (without angle brackets), an attribute, its ID, or a class.

When used as a CSS selector, the HTML tag for a table is **table**. For a row, it is **tr** and for a cell, **td**. A style rule that uses one of these, however, would apply to **all** tables, rows, or cells. For a rule to be more specific you need to add an ID (for a unique element) or a class (for a set of similar elements) to the table, row or cell, and use that as the style rule's selector.

Adding an ID or class to a table, row or cell

Before you can add an ID or class to a table, row or cell, you have to select that table, row or cell (see "Selecting a table, row or cell" on page 902). After selecting the cell, row or table, type the ID or class in the respective field on the **Attributes** pane.

In CSS, refer to the table, row or cell with `#ID` (where ID should be replaced with the actual ID) or with `.class` (where class should be replaced with the actual class).

Styling the first, last and nth rows

The CSS pseudo-classes `:first-child`, `:last-child` and `:nth-child()` are very useful for styling table rows, especially in Dynamic Tables.

A CSS **pseudo-class** follows a selector to specify a special state of that selector. It always starts with a colon.

The pseudo-classes `:first-child`, `:last-child` and `:nth-child()` select an element only if it is the first, last or nth child element respectively. (In HTML and CSS, the word **child** refers to an element inside another element.)

The following CSS style rule selects the table row (tr) that comes first (:first-child) in its parent (which naturally is a table), and colors its background red:

```
tr:first-child {
        background: red;
}
```

> **Tip:** In a Dynamic Table, data are in the body of the table (selector: tbody) and subtotals are in the footer (selector: tfoot).

Selecting a specific row, odd or even rows, or every nth row

The pseudo-class `:nth-child()` lets you select a specific row, all odd or even rows, or every **n**th row.

Between the round brackets in `:nth-child()` you can fill in a number, `odd` or `even`, or a formula: `an+b`. In the formula, `a` represents a cycle size (every...), `n` is a counter (for the child elements), and `b` is an offset value ('start at b'). The following examples will make this clear.

`:nth-child(3)` matches just one element: the third child element.

`:nth-child(odd)` matches child elements 1, 3, 5, 7, etc. The keyword `odd` substitutes the expression `2n+1`, which in other words says: 'take every second element, starting at 1'.

`:nth-child(even)` matches child elements 2, 4, 6, 8, etc. The keyword `even` substitutes the expression `2n+0`, or simply `2n`.

`:nth-child(3n)` matches child elements 3, 6, 9, 12 etc.

`:nth-child(3n+1)` matches child elements 1, 4, 7, 10 etc., so every third element, starting at 1.

Via script (based on a data field value)

To style a table, row or cell based on a data field value, you have to write a script (see "Writing your own scripts" on page 1064).

First add an ID or class to the table, row or cell that needs to be styled: select the element (see "Selecting a table, row or cell" on page 902) and add an ID on the **Attributes** pane. Then create a script, using that ID or class as the script's selector. The script can be very simple:

```
if (record.fields.COUNTRY == 'CANADA') {
```

```
    results.css('color','green');
}
```

The Designer Scripts API provides several functions to style elements, for example `css()`, `hasClass()` and `addClass()` (see "Standard Script API" on page 1481).

For information regarding scripts for **Dynamic Tables**, see "Using scripts in Dynamic Tables" on page 1095.

Tip: Styling Dynamic Tables with scripts is demonstrated in this howto: How to script the appearance of data in a Dynamic Table

Styling an image

Just like many other elements, images can be given borders and rounded corners, and they can be rotated. How to do this isn't any different from the way it is done with other elements, so it isn't described in this topic, but in general formatting topics; see "Styling and formatting" on page 879.
This topic discusses specific image formatting issues.

Note that image characteristics like brightness and contrast can not be changed within the Designer.

Local formatting vs. style sheets

Just as other elements, images can be styled in two ways:

- With **local formatting**. This means styling the image directly, using the Formatting dialog.
- Via **Cascading Style Sheets** (CSS). In a style sheet, style rules are declared for elements with different HTML tags, ID's and classes.

See "Styling and formatting" on page 879 for background information about these two methods.

Applying local formatting to an image

To apply **local** formatting to an image, either:

- right-click the image and select **Image...** from the contextual menu
- click the image and select **Format > Image...** from the menu.

For an explanation of the available options, see "Image Formatting dialog" on page 1153.

Applying style rules to an image

To format an image via a **style sheet**, first give the image an ID or class: select the image, and enter the ID or class on the Attributes pane. This makes it possible to make the CSS style rule target this image specifically, or a set of images with the same class. A style rule with the selector `img` (the HTML image tag) would apply to all images.
Next, create the style rule; see "Styling templates with CSS files" on page 881. Note that when a property isn't present in the style rule editor, it can still be used: click the Advanced button in the style rule editor; enter the property under Property, and its value under Value.

Resizing an image

There are several ways to resize an image after inserting it in the content of a template.

- Click the image and drag the handles to resize it. Press the **Shift** key while dragging, to scale the image proportionally.
- Select the image (see "Selecting an element" on page 766) and type the desired width and height in the respective fields on the **Attributes** pane.
- Select the image and select **Format > Image**, on the menu. On the **Image** tab, change the **width** and **height** of the image.
- Set the size of the image in a style sheet (see "Styling templates with CSS files" on page 881).

The size can be set in a measure or as a percentage of the containing element.

> **Tip:** If placed in a Box, images are by default resized to fit that container. To change this behavior, change the **Fit within frame** setting in the "Image Formatting dialog" on page 1153, the "Attributes pane" on page 1241 or set the image's `object-fit` property in CSS. For examples, see MDN Web Docs - object-fit.
> More ways to scale and crop images are described in this how-to: Scale and crop images.

Reset the image size

To reset an image to its original size, select the image and click the **Reset Image Size** button on the Attributes pane, under Geometry.

Positioning an image

Wrapping text around an image

Initially, when an image is inserted into a paragraph, it behaves as if it were a character. Text isn't wrapped around an image automatically. To make that happen, you have to

change the `float` property of the **image** to `left` or `right`. This anchors the image to the left or right, allowing text to be wrapped around it.

Select the image (see "Selecting an element" on page 766) and use the Float left and Float right icons on the toolbar to change the position of an image within the text.

- The **Float left** button aligns the image to the left. The text is positioned to the right of it and is wrapped around the box.
- The **Float right** button aligns the image to the right, with the text wrapped around it to the left.
- The **No float** button positions the image where it occurs in the text, as if it were a character. Text is not wrapped around it.

To position an image using the menu, select the image and then select one of the options in **Format > Float**.

Alternatively, open the Formatting dialog (see "Applying local formatting to an image" on the previous page): select the image; on the menu, select **Format > Image** and on the **Image** tab, under **Text Wrap**, set the **Float** property.

The `float` property could also be changed via a style sheet. This property isn't present in one of the tab menus of the style rule editor directly, but you can add it and specify its value after clicking the Advanced button in the style rule editor (see "Applying style rules to an image" on the previous page).

Pulling an image out of the text flow

When dragged into a template, an image is automatically integrated in the text flow. This means that it will move up or down, depending on the preceding text.
In a Print section, to position the image independently of the text flow, you can change its `position` property to `absolute`. (For an explanation of all values that the `position` property can possibly have, see MDN Web Docs - position.)
When an image is placed inside a Box (or Div element), changing its `position` property to `absolute` positions the image absolutely inside that Box.
Note that `float`, the property that can make an image float to the right or left (see "Wrapping text around an image" on the previous page), is a relative positioning property, since it specifies the position of the element relative to its container. This means it is incompatible with the `position:absolute` property.

In the Formatting dialog (see "Applying local formatting to an image" on page 906) the `position` property can be found on the **Image** tab, under **Positioning**.
The `position` property isn't present in one of the tab menus of the style rule editor directly,

but you can add it after clicking the Advanced button in the style rule editor (see "Applying style rules to an image" on page 906).

When the `position` property of an element is set to `absolute`, the `top` or `bottom` and `left` or `right` properties can be used to position the element inside its parent with exact values (pixels (px), centimeters (cm), etc). Negative values are allowed.

Note: In Web sections, the `position` property may sometimes be useful for images inside a Div element, but generally elements should not be positioned absolutely. Designs for the Web should be flexible so that they display nicely on a variety of devices and screen sizes.

Background color and/or image

In any type of template, boxes, tables and table cells can have a background color and/or a background image.

To select a background image or color:

1. Right-click the element and select the respective element on the shortcut menu.

   Alternatively, select the element (see "Selecting an element" on page 766; note that a Box is a element) and on the **Format** menu click the respective element.

2. Click the **Background** tab.

Note: Print section backgrounds are set differently. See "Using a PDF file or other image as background" on page 634.

To define a **background color**:

Click the downward pointing arrow next to **Color** to select a color from the list of predefined colors (see "Background color and/or image" on the previous page).
Alternatively, click the small rectangle to the right of the color list to open the Color Picker dialog. In this dialog you can select a color from the color wheel. You can also choose the color mode: RGB or CMYK. For an explanation of these two modes, see "Background color and/or image" on the previous page; for an explanation of the other options in this dialog, see "Color Picker" on page 1143.
You could also type a name or value in the Color field directly. It must be a valid color name (see MDN Web Docs - named color), a hexadecimal color code (see w3school's color picker), RGB color value, for example `rgb(216,255,170)` or CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.

Use the slider to set the color saturation.

To select a **background image**:

1. Click the **Select Image** button.
2. Click **Resources**, **Disk** or **Url**, depending on where the image is located.
    - **Resources** lists the images that are present in the **Images** folder on the **Resources** pane.
    - **Disk** lists image files that reside in a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder (or an image in a folder).
      As an alternative it is possible to enter the path manually. The complete syntax is: file://<host>/<path>.

      Note: If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/resources/images/image.jpg.
    - **Url** lists image files from a specific web address. Select the protocol (**http** or **https**), and then enter a web address (for example, http://www.mysite.com/images/image.jpg).
3. With an external image, you can check the option **Save with template**. If this option is checked, the file will be inserted in the **Images** folder on the **Resources** pane.

   If not saved with the template, the image will remain external. Note that external images need to be available when the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced. External images are updated (retrieved) at the time the output is generated.
4. Select an image from the list.
5. If the image is contained in a PDF file that consists of more than one page, select the desired page.
6. Click **OK**.
7. Set the size of the image. The options are explained here: MDN Web Docs - background-size.
8. Set the position of the image in the element.
9. Finally, click **OK**.

Note: It is also possible to set an element's background in a style sheet; see "Styling templates with CSS files" on page 881. When referring to images or fonts from a CSS

file, refer to a path that is relative to the current path, which is css/. For example: **#header { background-image: url('../images/image.jpg'); }**.

Border

In any type of template, boxes, tables and table cells, paragraphs, images and other elements can have a border.

Elements have a rectangular shape, so their border has four sides. Each side of the border can have a different layout.

Adding a border

1. Right-click the element and click the respective element on the shortcut menu. Alternatively, select the element (see "Selecting an element" on page 766) and on the **Format** menu click the respective element.
2. Click the **Border** tab.
3. Uncheck the option **Same for all sides** to be able to style each side of the border separately.
4. Specify the width of the border (side). This is equivalent to the `border-width` property in CSS.
5. Specify the style of the border (side), such as solid, dashed or dotted. This is equivalent to the `border-style` property in CSS.
6. Specify the color of the border (side): select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`. This setting is equivalent to the `border-color` property in CSS.

Note: It is also possible to set an element's border in a style sheet; see "Styling templates with CSS files" on page 881.

Rounding corners

Any element in a template can have rounded corners. For boxes and images, this option is available in the Formatting dialog. For other elements, you have to create a CSS rule to set the `border-radius` of the element (or class of elements).

Boxes, images and tables

To round the corners of a box, image or table:

1. Select a Box, Image or Table element (see "Selecting an element" on page 766) and on the **Format** menu click the respective element. Alternatively, right-click the element and click the respective element on the shortcut menu.
2. On the first tab in the Formatting dialog (the **Box**, **Image** or **Table** tab respectively) specify the **corner radius** in a measure (10mm, 5px, 0.5in) or percentage (0 - 90%).
3. For a Box or Image, click **Apply** to see the effect without closing the dialog or **OK** to close the dialog.

For a Table, you have to take yet another step. Tables can't have rounded corners and collapsed borders at the same time. All built-in table styles in the Designer have collapsed borders. For the rounded corners to show, you must create a CSS rule that sets the table's `border-collapse` property to `separate` instead of `collapse`.

1. Click the **Advanced** button at the bottom of the Formatting dialog.
2. Under **Property**, type **border-collapse**.
3. Under **Value**, type **separate**.
4. Add a padding to keep the table cells from sticking out of the rounded corners: under **Property** type **padding** and under **Value** type a measure for the padding.
5. Click OK, and click OK again to close the Formatting dialog.

If the table's rounded corners are still not (fully) visible, check the styles for table cells. Table cells can have their own background color and by that, hide the table's background color - including the rounded corners. Table cells can have rounded corners as well, just as any other elements; see below.

Other elements

To round the corners of elements other than boxes and images, or to have different roundings on different corners, you have to make use of the CSS property: `border-radius`; see MDN Web Docs - border-radius.

This is, for example, how you could round the corners of a paragraph:

1. Select the paragraph (see "Selecting an element" on page 766) and then select **Format > Paragraph** on the menu, or right-click the paragraph and select **Paragraph** on the shortcut menu.
2. Click the **Advanced** button at the bottom of the Formatting dialog.

3. Under **Property**, type **border-radius**.
4. Under **Value**, type the value of the corner radius in a measure (10mm, 5px, 0.5in) or percentage (0 - 90%).
5. Click OK, and click OK again to close the Formatting dialog.

Using a CSS file

Of course you could also add this rule to a CSS file; see "Styling templates with CSS files" on page 881. The following rule sets the border-radius of the corners of all paragraphs to 5 pixels:

```
p { border-radius: 5px; }.
```

To make this rule apply to one specific paragraph, first give the paragraph an ID (select the paragraph and type the **ID**, for example **rounded**, on the **Attributes** pane). Then add the ID to the selector of the CSS rule, for example

```
p#rounded { border-radius: 5px; }
```

To make the CSS rule apply to a set of paragraphs with the same class, first give the paragraphs the same **class** (for example **rounded**). Then add that class to the selector of the CSS rule, for example

```
p.rounded { border-radius: 5px; }.
```

Colors

Colors make an important contribution to the look and feel of your templates. This topic explains how to define and apply colors and how to keep them consistent in different output channels.

Defining colors, spot colors and tints

Color selectors, such as the drop-down list on the toolbar, initially contain a small set of colors. Add your own colors so that they can be used throughout the templates, in all contexts and in color selector dialogs as well as with their names in style rules (see "Styling and formatting" on page 879).

Defining colors

To define colors:

1. Select **Edit > Colors** on the menu.
2. Add a color. There are two ways to do this:
   - Click the **New** button (the green plus).
   - Select an existing color from the list and copy it using the **Duplicate** button. (The Filter drop-down limits the list to colors of a certain type.) Select the new color and click the **Edit** button.
3. In the Edit color dialog, type a name for the color (or let the Designer create a name based on the values that you select). The color's name can be used in style sheets. It should not contain spaces or special characters.

   Tip: Working with style sheets? Choose a name that reflects the purpose of the color, rather than a name that describes the color. This way you won't have to change the color's name in the style sheets when you change the color.
4. Click **Color**. (Tint is used for transparent colors.)
5. Select the color type: **CMYK** or **RGB**.

   The letters **CMYK** stand for Cyan (a greenish-blue color), Magenta (reddish-purple), Yellow and Key (black). In color printing, these are the usual primary colors.

   Note: CMYK colors may look different in Designer when compared to the output, but the colors in the output match the colors configured in the template.

   RGB stands for Red, Green and Blue. In the RGB color model, red, green, and blue light are added together in various ways to reproduce a wide range of colors. This model is typically used for electronic devices.

   In order to define a **PANTONE** color, set up a Spot Color and set its name to the PANTONE swatch name.

   For information about the **Spot color** and **Overprint** options see "Defining a spot color" below.
6. Drag the slider bars to set the values for the color and click OK or Apply.

Defining a spot color

A **spot color** is any color generated by an ink (pure or mixed). Note that spot colors can only be used on certain printers.

If your printer can use spot colors and you want a spot color to be used in a Print context, define the color as described above, making sure to:

- Match the color's **name** to that of the spot color used in the printer.
- Check the option **Spot color**.
- If applicable, check the **Overprint** option for this spot color. Overprinting refers to the process of printing one color on top of other colors. This is sometimes required, for example to deal with special print applications, such as applying UV ink or varnish to a certain area, or to avoid mis-registration when printing black on top of colored areas.

  Note: Black overprint can be enabled for text smaller than a given size; see "Overprint and black overprint" on page 628.

Defining a tint

A tint is a transparent color, based on another color in the template. To define a tint:

1. Select **Edit > Colors** on the menu.
2. Click the **New** button (the green plus) to add the tint.
3. From the Type drop-down, select **Tint**.
4. In the Edit color dialog, type a name for the color (or let the Designer create a name based on the values that you select). This name should not contain special characters. The color's name can be used in style sheets; see "Using colors in style sheets" below.
5. Select one of the existing colors in the template as the **Source** of the color. The tint or opacity will be applied to this color.
6. Check **Use opacity** if you want to set the Tint slider to use Opacity instead.
7. Use the slider to set the percentage of the tint or opacity, or type the percentage directly in the input box and finally click OK.

Applying a color

Colors can be applied to elements in your templates locally or through style sheets.

Using colors in style sheets

It is highly recommended to use style sheets in templates right from the start. Even more so if the communications are going to be output to different output channels, or if they consist of different sections (for example, a covering letter followed by a policy). Using CSS with templates allows a consistent look and feel to be applied. A style sheet can change the look

of multiple elements, making it unnecessary to format each and every element in the template, if the company's layout preferences change. See "Styling templates with CSS files" on page 881.

In style sheets, you can color every type of element that has a CSS color property, such as **color**, **background-color** or **border-color**.

Use the color's name as it is defined in the Designer, or any legal color value: a valid color name (see MDN Web Docs - named color), hexadecimal color code (see w3school's color picker), RGB color value, for example rgb(216,255,170) or CMYK color value, for example cmyk(15%, 0%, 33%, 0%).

The following CSS rule applies MyColor, which is a custom color (see "Defining colors, spot colors and tints" on page 913), to the text of all paragraphs:

```
p {
color: MyColor;
}
```

Note: Spaces need to be escaped.

```
p {
color: Empire\ Green;
}
```

CMYK() function

You may use the custom cmyk() CSS function to assign a CMYK color to any element, or a series of elements. The following example assigns a steel blue color as a background for all H1 elements:

```
h1 {
background-color: cmyk(33%, 17%, 0%, 20%);
}
```

Note: CMYK colors may look different in Designer when compared to the output, but the colors in the output match the colors configured in the template.

Coloring text

Instead of using a style sheet (see above), you can color text locally:

1. Select text or an HTML element that contains text (see "Selecting an element" on page 766).

2. On the menu, select **Format > Color**, or click the black triangle on the **Text color** toolbar button.
3. Select one of the colors in the list, or click **Other** to set all aspects of the text style, including text color and/or background color.

Coloring backgrounds and borders

Instead of using a style sheet (see above), you can color a background or border locally. This is how:

1. Select an HTML element (see "Selecting an element" on page 766).
2. On the **Format** menu, click the element. For a **div** element, click **Box**. The Formatting dialog opens up.
3. Click the **Border** or **Background** tab.
4. Click the downward pointing arrow next to **Color** to select a color from the list of predefined colors (see "Defining colors, spot colors and tints" on page 913).
   Alternatively, click the small rectangle to the right of the color list to open the Color Picker dialog. In this dialog you can select a color from the color wheel. You can also choose the color mode: RGB or CMYK. For an explanation of these two modes, see "Defining colors, spot colors and tints" on page 913; for an explanation of the other options in this dialog, see "Color Picker" on page 1143.
   You could also type a name or value in the Color field directly. It must be a valid color name (see MDN Web Docs - named color), a hexadecimal color code (see w3school's color picker), RGB color value, for example `rgb(216,255,170)` or CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.

   Use the slider to set the color saturation.
5. Click **OK** or **Apply**.

Color management

Color profiles can keep colors consistent across different outputs. To manage color profiles, select **Edit > Color settings**; for an explanation of the options in the Color settings dialog, see "Color Settings" on page 1145.

Fonts

In templates for personalized customer communications you can use the operating system's fonts, including imported fonts. There are two ways to incorporate remote or "web" fonts, as well; see "Using remote fonts" on page 920.

When you are using a font that is not installed on your machine (for example, the bold or italic variant of a regular font) Windows tries to simulate the font in the Designer. Likewise, OL Connect tries to simulate the font in the output. It is however not guaranteed that the output will be exactly as shown in the Designer.

It is strongly advised to make sure that all used fonts are available and to always test the output before running production jobs.

The safest option is to import the fonts into the template itself; see "Importing a font" on the facing page.

If fonts used in a template are not imported into the template, they will need to be installed on both the computer where the template was designed and on the computer where the output will be created.

Note: Hosting non-standard fonts on the operating system in a **server** environment (as opposed to importing them into the template) is not recommended.
If output is produced on the server whilst running under a different account, that account might not have access to the font.

If you do add a font to a server, select the option "Install for all users", and do not forget to **restart** the machine, as otherwise the font might not be available, due to the way certain Windows versions handle fonts.

Note: Variable fonts - officially known as OpenType Font Variations - are not supported. In order to convert a variable font to a non-variable font you could try https://fontforge.org/en-US/ (but note that this doesn't work for CCF2 fonts).

Applying a font

To apply a particular font to a piece of text, you can:

- Select some text, or an element that contains text (see: "Selecting an element" on page 766) and select a font from the **Fonts** drop-down on the toolbar.
- Use the name of the font in a CSS rule, for example:

```
body {
font-family: Verdana, Arial, sans-serif;
}
```

  Instead of the body tag, any element that can have the CSS property 'font-family' can be used.
  Make sure that the rule is applied to the text that you wanted to apply the font to; see "Step 2: apply CSS to the content" on page 887.

Note: The reason for specifying more than one font in a style sheet for web pages and emails is that the font might not be available on the device on which they are viewed.
Order the font names by preference. The last one should be the generic font family (either serif or sans-serif).

Importing a font

To import a font into a template:

- Drag the appropriate font files into the **Fonts** folder on the **Resources** pane.

Note: Font software may have specific restrictions for copying and redistribution. Please consult the license agreement for each font vendor before using it in a template. It is your responsibility to comply with the requirements of third-party agreements.

Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

Using separate font files for font effects

Fonts normally come with separate files for the **bold**, *italic* and any other versions of the font.
When you style text, Connect will use the appropriate font file to display the text, if that font file is linked to the respective font effect.
Here's how to combine a font file with a font effect.

1. Import the files for the bold, italic and any other versions of the font into the Fonts folder.
   Initially, imported fonts appear as separate entries in the Fonts drop-down on the toolbar. They are not used automatically when you style text, for example using the Bold and/or Italic toolbar button. For that, they have to be linked to a font effect.
2. Open the **Font Manager**: from the menu, select **Edit > Fonts**.
3. Combine each of the styled fonts with a font effect:

a. Select the font and click the **Edit** button.

b. Select the appropriate font effect (e.g. Font Weight: Bold, or Font Style: Italic).

c. Change the name of the styled font. It should have the **same** name as the regular font.

4. Close the Font Manager.

Style simulation

When it can't use a separate font file for a particular font effect, Connect will **mimic** the bold and italic versions of that font. This is called style simulation.
There are some major drawbacks to style simulation. Firstly, it is not reliable: it does not work for all fonts. Secondly, style simulation is costly in terms of performance (output is created slower) and results in bigger files. Finally, because of the plain machine transformation of the font, the result will probably not look as good as when using a carefully designed, properly styled, font.
It is much better, if you have separate files for the **bold**, *italic* and any other versions of a font, to use those.

Font types

The Designer currently supports 5 font types: TTF, OTF, WOFF, EOT and SVG.

When you are creating a **Web** template, keep in mind that the different font types are not supported by all clients; for instance, EOT and SVG are used only by Explorer and Safari, respectively.

When creating an **Email** template, it's better to import several types of the same font, in order for any client to see the appropriate fonts.

In the case of a **Print** context you do not need to provide alternative fonts, because the output is not displayed using a font from the device on which the output is read.

Applying an imported font

Once a font is imported, it is automatically added to the Fonts drop-down on the toolbar.

It can also be used in the style sheets, even in combination with other fonts, for example:

```
body {
font-family: 'MyWebFont', Arial, sans-serif;
}
```

Using remote fonts

Adding remote web fonts directly

The most straightforward method for using web fonts in Designer is this:

1. With Designer open, select the **Fonts** folder from the Resources tab.
2. Right-click to display the context menu and select **Web Font**.
3. Provide a name for the font, and enter the URL.

This adds a link to the font in the Fonts folder, and the font can be used in your templates.

Note that some email clients are not able to display web fonts used in email messages. In those scenarios the email will use the fall back font of the client (unless specified differently in a stylesheet).

When choosing fonts, we suggest adding multiple styles (regular, italic, bold weights etc). When dedicated font styles are not available, the browser has to synthesize styles such as bold and italic. You can pick multiple font families and still get one stylesheet URL. For example, fonts from Google Fonts include all the chosen fonts.

Using a style sheet to point to the font style sheet

Instead of adding web fonts directly, you can add a remote style sheet that points to a web font style sheet, for example https://fonts.googleapis.com/css?family=Roboto+Slab. For instructions see "Using a remote style sheet" on page 884.

These remote fonts can be applied to content in a Master page, section, or Snippet. They may be used in a style sheet and they are automatically added to the Fonts drop-down on the toolbar. Note that the list of font names is based on the style sheets that are included in the active section (see "Applying a style sheet to a section" on page 888) or, when editing a Snippet, in the section that was active when the Snippet was opened.

> **Note:** Support for remote fonts in email clients cannot be relied upon, and not all remote fonts are supported by all browsers. It is therefore recommended to add fallback fonts to the specific style rules whenever using remote fonts in a **Web** or **Email** section (see "Applying a font" on page 918).

Locale

The locale is a setting that can affect date, time and currency output, and other formatting that depends on location and language. This setting is specific to each template, so changing it for one template will not affect other templates.

Assume that a record set has a `Date` field that contains the following date: `4/11/12`, and that this field has been added to the template using the Text Script Wizard with the Long Date format (see "Using the Text Script Wizard" on page 948 and "Formatting variable data" on page 952). If the locale is set to `en-US`, the date appears on the page as **April 1, 2016**. Setting the locale to `fr-CA` makes this text appear as **1 avril 2016**. Setting it to `zh-CN` will print **2016年4月1日**.

The locale can also be used in scripts; see "Writing your own scripts" on page 1064 and "Standard Script API" on page 1481.

Changing the locale

By default, the locale is the same as the operating system's locale setting. To change this setting for the currently open template:

1. On the menu, select **Edit > Locale**.
2. Use the drop-down to select how the locale is to be set for the current template:
    - Select **System Locale** to use the operating system's locale settings. The operating system's locale is set in the **Region** settings of the control panel. Note that when output is generated on a different operating system, that operating system's locale will be used.
    - Select **Explicit Locale** to specify a static locale which will remain static for this template, whichever server the template is used on. Use the **Locale** drop-down to select a specific locale. The locales comprise a language code followed by a 2-letter country code (`de-DE`, `zh-CN`, `fr-CA`, `fr-FR`, etc), as defined by the international standards ISO-639-1 and ISO 3166.
    - Select **Data Field** to use a data field from the record. The locale will be record-specific in this case. Use the drop-down to select a field within the current Data Model that contains the locale.
    - Select **Parameter** to use the value of a runtime parameter. Runtime parameters are defined in the template, but their value is set at runtime; see "Runtime parameters" on page 608.

    Note: The Data Field or Runtime Parameter value must be a string that contains the exact locale to be used, such as "en" or "fr-CA". It cannot be an alias such as "english" or "french". The locale supports language codes (`en`, `fr`, etc), as well as language codes followed by a hyphen and a 2-letter country code (`de-DE`, `zh-CN`, `fr-CA`, `fr-FR`, etc). The language codes are defined by ISO-639-1. The 2-letter country code is defined by ISO 3166. It is allowed to use an underscore instead of a hyphen (`de_DE`, `zh_CN`, `fr_CA`, `fr_FR`, etc).
3. Click **OK** to apply the setting. The setting will be saved with the template.

Spacing

Boxes, tables, paragraphs and many other elements have a **margin** and **padding**.
The margin is the white space around an element, outside the border. It is used to position an element in relation to the other elements, by putting more space between the element and its surrounding elements.
The padding is the space between an element's content and its border. It is used to position the content of the element inside the border.

Elements have a rectangular shape, so they have four sides. The margin and padding have be different on all sides.

Tip: Use a negative left margin to create a hanging paragraph or image.

To set the spacing:

1. Right-click the element and click the respective element on the shortcut menu. Alternatively, select the element (see "Selecting an element" on page 766) and on the **Format** menu click the respective element.
2. Click the **Spacing** tab.

   Note: All settings in the Formatting dialog are in fact CSS style rules. Click the **Advanced** button to manually add CSS properties (at the left) and values (at the right). For more information about CSS, see "Styling and formatting" on page 879.

   It is also possible to change an element's formatting via a style sheet; see "Styling templates with CSS files" on page 881.
3. Set the value for the **padding** in measure or percentage. You can do this for each side separately, which is equivalent to the **padding-top**, **padding-bottom**, **padding-left** or **padding-right** property in CSS. To set the same padding for all sides, check the option **Same for all sides**. This is equivalent to the **padding** property in CSS.
4. Set the value for the margin in measure or percentage. You can do this for each side separately, which is equivalent to the **margin-top**, **margin-bottom**, **margin-left** or **margin-right** property in CSS. To set the same margin for all sides, check the option **Same for all sides**. This is equivalent to the **margin** property in CSS.
5. Click **OK**, or click **Apply** to apply the changes without closing the dialog.

Personalizing content

Variable-data printing is a form of digital printing in which elements such as text and graphics may be changed using information from a database or data file. It prints unique documents with customized messages for each customer. This is exactly what you can do

with OL Connect: using variable data you can personalize your company's communications (including but not limited to printed matter).

You can only start personalizing the content of a template if there is an open data mapping configuration, Data Model, data file or database; see: "Loading data" on page 926.

Alternatively, you could create a Data Model, without extracting data, within the Designer; see "Changing the Data Model" on page 1246.

The most common ways to personalize templates are listed below.

Variable data in the text

Variable data can be inserted in a template directly. For example, if a data field in your data holds a person's last name, the name of that data field can be put in the template, in an *expression* or a *placeholder*. When the template is merged with the data, the expression or placeholder gets replaced with a value, in this case, a person's last name.

Expressions look like this: *{{FieldName}}*. They get replaced with data from the current record by the Handlebars library which is integrated in OL Connect. The advantage of expressions is that fewer scripts and sometimes no scripts at all are needed in a template. See "Variable data in text: expressions" on page 999.

Placeholders require a script to replace them. See "Variable data in text: scripts and placeholders" on page 945.

Conditional content

In a template you may want to reveal content - text or images - to one group of recipients, but hide it from others. You can use a Conditional Script Wizard to achieve this, if you have a data field in your data on the basis of which a condition can be set. See "Showing content conditionally" on page 958.

Conditional Print sections

Entire Print sections can be included in or omitted from the output on the basis of one or more values in variable data. See "Conditional Print sections" on page 962.

Dynamic images and Print section backgrounds

Dynamic Images are dynamic in the sense that they are replaced by another image when a data field contains a certain value. Think of a signature image being swapped based on the

sender's name, for example. You can use the Dynamic Image Script Wizard to make this happen; see "Dynamic images" on page 972.

To swap a Print section's background dynamically, you can use the Dynamic Background Script Wizard; see "Dynamic Print section backgrounds" on page 994.

Dynamic tables

A Dynamic Table is a table with a variable number of rows that can overflow on one or more pages. It can also display subtotals on transport lines. In invoices, a Dynamic Table is an essential element. Read "Dynamic Table" on page 974 to learn how to insert a dynamic table.

Section clones

If documents in the output need to have a variable number of sections, based on data, Print sections can be cloned (copied) on the fly. See "Cloning a section using the wizard" on page 966.

Snippets

Snippets are pieces of content that can be re-used within the same template, in all contexts and sections. Snippets can contain any contents that a section can have, such as text, images, variable data, dynamic tables, etc. They are often very useful to personalize content, especially in combination with variable data and scripts. See "Snippets" on page 867 and "Loading a snippet via a script" on page 1091.

Handlebars snippets

Handlebars templates are a special kind of snippets in which you can use Handlebars expressions. The expressions get replaced with values when the snippet is merged with data. You determine - in a very small script - with which data the snippet will be merged. See "Handlebars templates" on page 1023 for instructions.

Handlebars is the name of a JavaScript library that implements a templating language (see https://handlebarsjs.com/). It uses a template and an input object to generate output. The Handlebars library is **integrated** in OL Connect Designer. See: "Handlebars in OL Connect" on page 998.

Scripts

As soon as you want to do more than what can be done with the available Script Wizards, self-made scripts are the solution. You could, for example, combine data of two or more data fields in a condition for conditional text. Or you could load a part of a snippet depending on the value of a data field. With a self-made script you can achieve anything that can be done by any of the Script Wizards, and much more.

The basics of script-writing in the Designer are explained in the following topic: "Writing your own scripts" on page 1064.

Control Scripts

When output is generated from a template, Control Scripts run **before** all other scripts, when a record is merged with a context. They determine how different sections of the context are handled. They can, for example, make the page numbering continue over all Print sections, split Email attachments, or omit Print sections from the output.

Some functionality is provided as a Scripting Wizard, for example Conditional Print Sections.

See "Control Scripts" on page 1098.

Post Pagination Scripts

Post Pagination Scripts are run in a **Print** context **after** the content has been paginated. Because they can search through the output of all Print sections, and modify Print sections (one at a time), they may be used to create a Table Of Contents (TOC), as explained in the topic: "Creating a Table Of Contents" on page 1116.

See "Post Pagination Scripts" on page 1114.

Loading data

In order to personalize the content of a template in the Designer, you need to have a Data Model and a sample of customer data. At the design stage the Designer doesn't need to have access to all data; it just needs to know which data fields exist in your data and it needs some data to be able to display a preview of the output.

To get access to a Data Model **and** data, you can open:

- A data mapping configuration, see "Loading a data mapping configuration" on page 928. A Data Model and sample data are part of a data mapping configuration.
- A data file, see "Adding data from a data file" on page 929.

- A database, see "Adding data from a database" on page 933.
- JSON sample data, see "Adding JSON sample data" on page 937.

When you open a data file or a database, the Data Model will be derived from it. That is, **unless** there already is an open data mapping configuration; in that case, the current data mapping configuration will try to retrieve data from the file or database, using its own Data Model and extraction logic.

After opening a data mapping configuration, data file or database, the **Data Model** pane at the right hand bottom shows the data fields that occur in the data.

The **Value** column displays data from the first record in the data file. Use the **First**, **Previous**, **Next** and **Last** buttons to browse through the records, or use the Page Up, Page Down, Home and End keys.

Note: When loading data from a CSV or Excel file, the data type of each field (String, Integer, Boolean, Currency, Float or Date) is auto-detected based on the first 10 rows. With other data files and when data is loaded directly from a database, all values are initially strings.

In order to set conditions based on a data field, or to apply formatting to a value, the data type may be important. To change the type of a field, right-click it and select **Set Type**.

Designing a template without sample data

Strictly speaking, you don't need sample data to design a template; the only thing you really need in order to add variable data fields to a template is a Data Model.
You can open a Data Model without data by importing a Data Model file, a JSON file or a Connect template file from within the Data Model pane, using the Import icon at the top right. The file's data model structure will be displayed in the Data Model pane, but the data is not included.
You can also add fields and tables and set default values for fields in the Data Model pane; see "Changing the Data Model" on page 1246. This way you could even build a Data Model in the Designer from scratch.
However, without sample data you won't be able to preview the template with actual data in the Designer.

Tip: It isn't possible to enter values directly in the Data Model pane, but there is a workaround: click the **JSON Sample Data** toolbar button on the Data Model pane. You will see a JSON string that represents a single record based on the current Data

Model, with dummy values (like empty strings and zeros). Edit that JSON string and click OK to insert the data in the Data Model.

Designing a template with sample data

Generally, the best way to access data is by creating a **data mapping configuration**. With a data mapping configuration you can, among other things:

- Use Workflow to automate the extraction of data from this kind of data file.
- Load transactional or structured data. If there are detail lines, transactions, or any variable number of items to put into the template, you need a data mapping configuration to extract them.
- Format, transform, conditionally include/exclude and enhance data from the source file.
- Use other field types. All fields contain strings when loaded from a database or file (except CSV and Microsoft Excel files).
- Use the same data file with different templates, or use different kinds of data files with the same template.

Associated data file or data mapping configuration

When you save a template, any data file or mapping configuration that is currently open will be associated with the template by saving the path to the data file or data mapping configuration in the template file.
The next time you open the template the Designer will attempt to reopen the data file or data mapping configuration as well.

To change which data file or data mapping configuration is linked to the template, open both the template and the data file or data mapping configuration that should be linked to it; then save the template.

Loading a data mapping configuration

If you already have an open data mapping configuration, its Data Model and sample data will automatically be used when you start creating a template. You might have to click the **Synchronize Model** button on the **Data Model** pane, to update the fields.

To open a data mapping configuration:

1. Open the Welcome screen: click the Home icon at the top right or select **Help > Welcome** on the menu.

2. Click **Open**.
3. Select the data mapping configuration and open it.
4. At the top of the workspace, click the tab with the name of the template's section to go back to the template.
5. Click the button **Synchronize model** at the top of the **Data Model** pane to reload the data model.

Note:

- If the View ExtraData option is checked in the preferences (Window > Preferences > Data Model), you can click the Show/Hide ExtraData fields icon in the Data Model toolbar to show an ExtraData field as the first field in each record and in each detail table. These fields are automatically added to the Data Model by the DataMapper. They were originally intended to be used in a Workflow configuration to add data to the Data Model. Nowadays, **runtime parameters** are the preferred way to pass data at runtime to a data mapping configuration (see "Properties and runtime parameters" on page 387) or template (see "Runtime parameters" on page 608).
  ExtraData fields and runtime parameters can be used in a template just like any other data field (see "Variable data in the text" on page 924).
- When generating output with just a data mapping configuration, the template is merged with the complete sample data file that is part of the data mapping configuration. The output is **not** limited to the number of records shown in the Data Model pane (which is one of the settings in the DataMapper).

Tip: A data mapping configuration can have more than one sample data file. Switching between sample data files while you are working on the template is easy. Click the down arrow in the Data Model toolbar and select the desired sample data file.

Adding data from a data file

Note: Note that all values are initially strings when data is loaded directly from a database or file (except CSV and Microsoft Excel files).

In order to set conditions based on a data field, the data type may be important. To change the type of a field, right-click it and select **Set Type**.

1. Click **File**, select **Data** and then click **Open Data File...**. Browse to the location of the file and select it.
   The Designer can open the following types of data files:
   - Tabular files: CSV files (.csv, .txt) and Microsoft Excel files (.xls, .xlsx)

     Note: Excel files saved in "Strict Open XML" format are not supported yet.
   - Microsoft Access Database (.mdb, .accddb)
   - JSON files (JSON)
   - XML files (.XML)
   - PDF/VT files
   - AFP files (Enterprise edition only)
2. Review the options presented, to ensure that the data will be interpreted correctly. The options available depend on the type of data file (see below).
3. Click OK and verify that the data are read properly.

   With **CSV** and Microsoft **Excel** files, the first 10 rows are analyzed to detect a field's data type. If the data in a column is not consistent throughout the file, an exception may be thrown if a value cannot be interpreted, and data loss may occur if the detected data type is more general than the data type in a subsequent row. For example, if an Integer is detected in the analyzed rows, but a subsequent row contains a Float value, the value of the Float value is truncated to an Integer.
   To change the detected type of a field, right-click it and select **Set Type**.

Once loaded, a link to this data file is saved with the template. The next time the template is opened, the data file is loaded as well.

To remove the link to the data file: click the arrow next to the Data file button in the Data Model toolbar, and select **Close data file**.

Microsoft Excel (XLS/XLSX) file options

- **First row contains field names**: Check this option to use the first line of the Excel file as headers. This option automatically names all extracted fields.

- **Sheet**: Only one sheet can be selected as the data source.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

CSV file options

- **Encoding**: The Designer can not infer from a CSV file what encoding it is in. The default is right in the large majority of cases, but when it isn't, it can be very difficult to figure out the correct encoding. Ask your source what the encoding of the file is.
- **Field separator**: Choose the character that separates the fields in the file.
- **Comment delimiter**: If there are comment lines in the file, type the character that starts a comment line.
- **Text Delimiter**: Type the character that surrounds text fields in the file. Other delimiters will not be interpreted within these text delimiters.
- **Ignore unparsable lines**: When checked, any line that does not correspond to the above settings will be ignored.
- **First row contains field names**: Check this option to use the first line of the CSV as headers. This option automatically names all extracted fields.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

Microsoft Access MDB file options

- **File**: Include the full path to the file.
- **Password**: If the file isn't password protected, you can click **Next** without filling out this field.
- **Table name**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
- **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

JSON file options

After selecting a JSON file, specify if and how the JSON file must be split into multiple records.
This is done by selecting an object or array as **parent element**. Its direct child elements - objects and arrays, *not* key-value pairs - can be seen as individual source records. If the root is selected, there will be only one source record.

Whether source records are output as individual records depends on the **trigger**. Either:

- Select **On element** to create a new record in the output for each object or array in the parent element.
- Select **On change** to create a new record each time the value in a certain key-value pair changes. Only key-value pairs that exist at the root of a child element can be evaluated.

All data found in child elements of the selected parent element are extracted to fields at the root of the Data Model. If a value consists of an object or array, the entire object or array is extracted to one data field.

Field names are derived from keys, objects and arrays in the *first* record, but those aren't necessarily the same in a subsequent record. If following records have a different structure, for example if a record has more child elements compared to the first record, some data may not get extracted.

> **Tip:** JSON data can also be imported directly into the Data Model; see "Adding JSON sample data" on page 937.

XML file options

Select what level of XML elements defines a record.

The **Trigger** is what triggers the creation of a new record. It can be set to:

- **On element**: This defines a new record when a new element occurs on the selected XML level.
- **On change**: This defines a new record when a specific field under the chosen XML level has a new value. After selecting this option, you have to select the field that triggers the creation of a new record.

PDF/VT file options

After selecting a file, use the drop-down to select what level in the PDF/VT file defines a record in your data. The names of the levels are taken from the PDF/VT file itself. (See

"About PDF/VT files" below.)
All metadata fields that belong to the chosen level and levels higher up in the tree structure will be listed. The lower the chosen level is in the tree structure, the more records you will get and the more metadata fields will appear in the list.
Select metadata fields to add them to your data. Their property names will be used as field names in the Designer's data model.

About PDF/VT files

The pages in PDF/VT files can be grouped on several levels. PDF/VT files can have a variable number of levels in their tree structure. The level's names are variable as well, with the exception of the lowest level, which is always called the **page** level. Metadata can be attached to each level in the structure.

AFP file options (Enterprise edition only)

After selecting a file, use the drop-down to select what level in the AFP file defines a record in your data. The levels are defined in the AFP file itself. (See "About AFP files" below.)
All metadata fields that belong to the chosen level and higher levels in the tree structure will be listed. The lower the chosen level is in the tree structure, the more records you will get and the more metadata fields will appear in the list.
Select metadata fields to add them to your data. Their property names will be used as field names in the Designer's data model.

About AFP files

Pages in AFP files are arranged in a tree structure, comprising one **document** at the top of the structure, **pages** at the bottom of the structure and one or more levels of **page groups** in between. (Unlike in PDF/VT files, the names of levels in AFP files cannot be chosen freely.) In other words, an AFP file always consists of one document, that can contain page groups, of which each can be divided in page groups, and so on; the page groups at the lowest level contain pages.
Metadata can be attached to each level in the structure.

Adding data from a database

1. Click **File**, select **Data** and then click **Connect with Database...**. Browse to the location of the file and select it.
   The Designer can open databases from the following types of data sources:
   - MariaDB
   - MySQL

- Oracle
- Microsoft SQL Server
- Microsoft Access Database (.mdb, .accddb)
- ODBC DataSource
- JDBC

2. Review the options presented. The options available depend on the type of database data source; see below.

MariaDB \ MySQL

1. Enter the appropriate information to connect to the database:
    - **Server**: Enter the server address for the MariaDB or MySQL database.
    - **Port**: Enter the port to communicate with the MariaDB or MySQL server. The default port is *330*.
    - **Database name**: Enter the exact name of the database from where the data should be extracted.
    - **User name**: Enter a user name that has access to the MariaDB or MySQL server and specified database. The user only requires *Read* access to the database.
    - **Password**: Enter the password that matches the username above.
2. Click **Next** and enter the information for the source table.
    - **Connection string**: Displays the full path to the database.
    - **Table**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
    - **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.
        - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.
3. Click **Finish** to open the database.

Oracle

1. Enter the appropriate information to connect to the database:
    - **Server**: Enter the server address for the Oracle database.
    - **Port**: Enter the port to communicate with the Oracle server.
    - **Database name**: Enter the exact name of the database from where the data should be extracted.
    - **User name**: Enter a username that has access to the Oracle server and specified database. The user only requires *Read* access to the database.
    - **Password**: Enter the password that matches the username above.
2. Click **Next** and enter the information for the source table.
    - **Connection string**: Displays the full path to the database.
    - **Table**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
    - **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.
        - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.
3. Click **Finish** to open the database.

Microsoft SQL Server

1. Enter the appropriate information to connect to the database:
    - **Server**: Enter the server address for the Microsoft SQLServer database.
    - **Port**: Enter the port to communicate with the SQLServer. The default port is *1433*.
    - **Database name**: Enter the exact name of the database from where the data should be extracted.
    - **User name**: Enter a user name that has access to the SQLServer and specified database. The user only requires *Read* access to the database.
    - **Password**: Enter the password that matches the user name above.
2. Click **Next** and enter the information for the source table.

- **Connection string**: Displays the full path to the database.
- **Table**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
- **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.
    - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

3. Click **Finish** to open the database.

Microsoft Access

1. Enter the appropriate information to connect to the database:
    - **File name**: Browse to your Microsoft Access database file (.mdb, .accddb).
    - **Password**: Enter a password if one is required.
2. Click **Next** and enter the information for the source table.
    - **Connection string**: Displays the full path to the database.
    - **Table**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
    - **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.
        - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.
3. Click **Finish** to open the database.

ODBC DataSource

1. Select the ODBC system data source. Note: Only 64-bit data sources can be used.
2. Click **Next** and enter the information for the source table.
    - **Connection string**: Displays the full path to the database.
    - **Table**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
    - **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.

- **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the selected column has numbers, it will be sorted as a text.

3. Click **Finish** to open the database.

JDBC

1. Enter the appropriate information to connect to the database:
    - **JDBC Driver**: Use the drop-down to select which JDBC Driver to use for the database connection.
    - **JAR file path**: Enter a path to the JAR file that contains the appropriate driver for the database below.
    - **Server**: Enter the server address for the database server.
    - **Port**: Enter the port to communicate with the server.
    - **Database name**: Enter the exact name of the database from where the data should be extracted.
    - **User name**: Enter a username that has access to the server and specified database. The user only requires *Read* access to the database.
    - **Password**: Enter the password that matches the username above.
    - **Advanced mode**: check to enable the Connection String to manually enter the database connection string.
    - **Connection string**: Type or copy in your connection string.
2. Click Next and enter the information for the source table.
    - **Connection string**: Displays the full path to the database.

      Note: By default, the connection is attempted with encryption enabled. To instruct the driver to not use encryption, add the ";encrypt=false" parameter to the connection string.

    - **Table**: Use the drop-down to select the appropriate table or stored query to retrieve the appropriate data set.
    - **Encoding**: Use the drop-down to select the encoding with which to read the data in the table.
        - **Sort on**: Select a field on which to sort the data, in ascending (A-Z) or descending (Z-A) order. Note that sorting is always textual. Even if the

selected column has numbers, it will be sorted as a text.

3. Click **Finish** to open the database.

After adding data from a database, the **Data Model** pane at the right hand bottom shows the data fields that occur in the data.

The **Value** column displays data from the first record in the data file. Use the **First**, **Previous**, **Next** and **Last** buttons to browse through the records.

Adding JSON sample data

JSON data can either be **added to**, or **replace** the Data Model in a template.

Importing JSON data into the Data Model makes it easier to test templates that are meant to be merged with JSON data in a Workflow configuration.

> **Tip:** Workflow's OL Connect Create Content tasks can use JSON as their data source; see Create Email Content, Create Print Content, and Create Web Content.)

To add JSON sample data to the Data Model:

1. Select **File > Data > Open JSON Sample Data**, from the menu. Alternatively, click the **JSON Sample Data** toolbar button on the Data Model pane.
2. Either browse to the location of the file and select it, or paste or write the JSON directly in the box below the file name.

   > **Tip:** If a data field contains JSON data you could use that data as sample data: right-click the field, select **Copy** to copy the JSON data to the clipboard; then open the JSON Sample Data dialog and paste the data there.

   You can add the following types of JSON data:

   - **A JSON object or an array of JSON objects** representing records. The data type is derived from the data:
     - Any value surrounded with quotes is converted to a field of type String.
     - Any numeric value containing a period is converted to a field of type Float.
     - Any numeric value that does not contain a period is converted to a field of type Integer.
     - A value "true" or "false" is converted to a field of type Boolean.
   - **Typed JSON**. This JSON follows the structure of a JSON Record Data List (see

the REST API Cookbook). Field types are determined by the `schema` object in the JSON.

Any nested JSON objects are displayed on successive levels in the Data Model.

3. Review the JSON; you may edit it if you like.
4. Select the **Replace Data Model** option if you want the JSON to replace the existing Data Model. Otherwise, the JSON data will be mapped to corresponding fields in the existing Data Model, and data that cannot be mapped to any field will be discarded.

   Note: If a data mapping configuration is open at the same time, the loaded JSON will always replace the Data Model and no values will be imported.

5. Click **Finish**.

Add a counter using the Generate Counter Wizard

Generating a counter is useful for numbered tickets or any other template requiring sequential numbers but no variable data.

The Generate Counter Wizard creates a **record set** with a Counter field and in that field, the current counter value for each record. The Counter starts and stops at set values and is incremented by a set value as well.

1. To open the Generate Counter Wizard, select **File > Add data > Generate counters**.
2. Adjust the settings:
   - **Starting value**: The starting number for the counter. Defaults to 1.
   - **Increment value**: The value by which to increment the counter for each record. For example, an increment value of 3 and starting value of 1 would give the counter values of 1, 4, 7, 10, [...]
   - **Number of records**: The total number of counter records to generate. This is not the end value but rather the total number of actual records to generate.
   - **Padding character**: Which character to add if the counter's value is smaller than the width.
   - **Width**: The number of digits the counter will have (prefix and suffix not included). If the width is larger than the current counter value, the padding character will be used on the left of the counter value, until the width is equal to the set value. For example, for a counter value of "15", a width of "4" and padding character of "0", the value will become "0015".

- **Prefix**: String to add before the counter, for example, adding # to get #00001. The prefix length is not counted in the width.
- **Suffix**: String to add after the counter. The suffix length is not counted in the width.

3. Click **Finish** to generate the Counter record set.

While the Generate Counter script is really useful for things like raffle tickets, it's unusable in combination with a data file or database, as it cannot complement that data automatically.

Tip: The Handlebars helper: "counter" on page 1010 generates sequential numbers that update for each record in the main data set.

Variable data in text: expressions

As of OL Connect 2022.2, variable data can be inserted in a template via **expressions**. Expressions look like this: *{{FieldName}}*. When the template is merged with the data, the expressions get replaced with values.

In previous versions, variable data used to be inserted in a template via **scripts and placeholders** (see "Variable data in text: scripts and placeholders" on page 945). The advantage of expressions is that fewer scripts and sometimes no scripts at all are needed in a template.

Powered by Handlebars

Expressions get replaced with data from the current record by the Handlebars library which is integrated in OL Connect. For more information about Handlebars, see: "Handlebars in OL Connect" on page 998.

Note: Before you can add variable data fields to the contents of your template you need to load - or create - a Data Model (see "Data Model pane" on page 1244), a data mapping configuration, or data from a data file or database (see "Loading data" on page 926).

Note: Web templates are personalized just like any other template. There are a few extra possibilities, though; see "Using variable data on a Web page" on page 690.

Note: Make sure to include the curly brackets when selecting an expression if you want to style the results. HTML tags inside the curly brackets break the expression.

Writing expressions

To insert an expression that gets replaced with the value of a field in your data, you can simply **write** the field name between double curly braces: *{{FieldName}}*.

- A whitespace before and after the field name is accepted, e.g. *{{ FirstName }}*.
- If the field name contains a space you will also need to enclose it in square brackets: *{{[Field Name]}}*.

This works in Design view and in Source view. You can even use expressions inside HTML tags and attributes.

> **Tip:** An easy way to add a dynamic image or hyperlink is to use an expression in the image's `src` attribute or in the hyperlink's `url` attribute.

In expressions, you can use functions called *Helpers*, such as "Format Helpers" on page 1015 and Block Helpers (see "Using functions in expressions: Helpers" on page 1009).

> **Tip:** If you miss a function, or if an expression becomes very long, consider writing your own Helper; see "Creating custom Helpers" on page 1018.

Inserting expressions via drag-and-drop / double-click

The **drag-and-drop** or **double-click** method is best for data that do not need to be preceded or followed by a space, line break or text. Otherwise it is better to use the wizard (see "Using the Helper Wizard" on page 943).

1. Open the section or Master Page you want to add the data field to.
2. Either:
    - **Drag and drop** a data field from the **Data Model** pane at the bottom right into the content of your template.
      To select and insert **multiple** data fields at the same time, you could press **Shift** or **Ctrl**, whilst selecting fields in the Data Model pane, but it would be better to use the wizard (see "Using the Helper Wizard" on page 943).
    - Place the cursor in the template where you want the data to be inserted and then **double-click** the data field in the Data Model pane.
    - Or, keep the **Ctrl** key pressed down while dragging to insert a "Positioned Box" on page 824 with the expression in it.

The first two methods work in Design view and in Source view, the latter only works in Design view.

Tip: Press the **Ctrl** key *while dragging* to wrap the expression(s) in an absolute positioned box (a **div**) at the cursor position.
Absolute positioned boxes are particularly useful when the document mainly consists of a PDF used as the background image of a section (see "Using a PDF file or other image as background" on page 634).

What happens is that:

- The name of the data field appears in the template, enclosed in double curly braces: **{{data field name}}**. This is called an **expression**.

  Example: Hello {{FirstName}}!

  Note: If the data field is of the type **HTML String**, the expression gets three curly brackets to ensure that the output is interpreted as HTML. In expressions with double curly braces, characters that have a special meaning in HTML are *escaped*, e.g. "&" is written out as "&". (See also: "HTML values" on page 1008.)

  Note: Does a **placeholder** appear instead of an expression? Placeholders look like this: *@FieldName@*. This happens by default when a template was made with an OL Connect version prior to 2022.2.

  To make the software insert expressions instead of placeholders, right-click the section in the **Resources** pane, select **Properties** and check the option **Evaluate Handlebars expressions**.

  See "Variable data in text: scripts and placeholders" on page 945 for an explanation of how placeholders work.

- Depending on the data type of the dragged field, a function that formats the value may be added to the expression automatically. For example, dragging a currency field will produce: *{{currency FieldName}}* which outputs a number formatted as an amount of money.

  Example: The amount due is {{currency Total}}.

  You can remove the function or change it (see "Format Helpers" on page 1015).

To see the expression replaced with the value of the data field in the current record, switch to the **Preview** mode by clicking the Preview tab at the bottom of the workspace. The value will be refreshed when you browse through the records in the Data Model pane.

The same happens when the output is generated. With each record, expressions that hold the name of a data field get replaced with the value of that data field.

> **Note:** Whether expressions in a Master Page are evaluated when output is generated, depends on the Print section that the Master Page is used with. See "Print section properties" on page 1200.

Detail data in expressions

When you drag and drop detail data in a template, the expression includes the path to the data field. For example, suppose there is a detail table called Products that has a Description data field. Dragging and dropping the Description field from the first record of the detail table inserts the following expression:
{{Products[0].Description}}
The **[0]** refers to the first record. Remember that detail tables can have a variable number of records.

> **Note:** Detail data can only be dragged into a row in a Dynamic Table that is based on the same (nested) detail table (see "Dynamic Table" on page 974).

> **Note:** Another way to insert detail data is to use a Block Helper, preferably in a Handlebars template (see "Block Helpers" on page 1012 and "Handlebars templates" on page 1023).

> **Tip:** Double-click the name of a detail table, or drag and drop it from the Data Model into the template, to start the Dynamic Table wizard. See "Dynamic Table" on page 974.

Using the Helper Wizard

Should the value of a data field be preceded or followed by a space, line break or text in the template, like in an **address block**? Then it is best to use the Helper Wizard. Especially if there are any **empty** fields in the data. Otherwise, empty data fields will cause empty lines and superfluous white spaces to show up in the text.

The Helper Wizard is also a handy tool to use if a field name, and hence the expression, is too long to fit in a table cell, for example.

> **Tip:** The following how-to shows how to create an address block using the Helper Wizard: Creating an address block. However, OL Connect also has a special wizard to help you insert an address block: "Address Block" on page 955.

The Helper Wizard creates a "Helper" - a function that can be called in an expression. When called, the Helper inserts one or more data fields into your template, each with an optional prefix and suffix.
The wizard also makes it easier to **format** data differently, for example, display a number as a currency.

1. Open the Handlebars Text Helper Wizard: on the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Handlebars Text Helper**.
2. Change the name of the Helper to make clear what it does. The name must be unique.

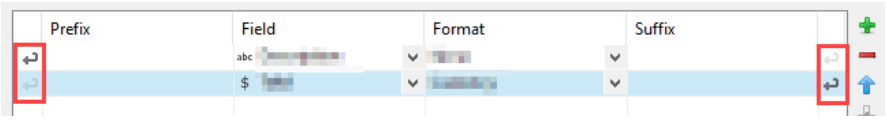
   The name of the Helper is what you use in an expression. For example, if its name is MyHelper, then the expression that calls the Helper is {{MyHelper}}.

   Tip: To make it easy to identify your own Helpers, you can start the name with a special character such as an underscore.

   Tip: Use a short name to make the expression fit in a small space.
3. Click the downward pointing arrow to select the **Scope**: the record or a detail table. The fields you choose in the Helper Wizard must all be on the root of the record, or in the same detail table.
4. Click the downward pointing arrow in the first row in the column **Field**. Select a data field from the list that appears.
5. Add a **Prefix** and/or a **Suffix**. The prefix and suffix can contain text and/or HTML tags, such as:

   - with a Number field, Prefix: *Your invoice* (one space at the end), Suffix: *is now ready to be viewed!*
   - with a field State, Prefix: , (comma then space).

   To add a single **line break**, activate the Line break button in either the first or last column.

   Alternatively, or to add an *extra* line break, you can add *
*, which is the HTML tag for a line break, to the Suffix. For example:

   - with a field LastName, Suffix: *
*

Note: If a field is empty, the prefix and suffix will be ignored.

Tip: For a comma between fields, use the Prefix of the second field if you don't want a comma when the second field has no value.

6. The Wizard allows you to format the data (for example, display a text in uppercase, apply thousands separators to numbers, etc.). Click the column **Format**, then click the downward pointing arrow and select one of the formats. For an explanation of the formats see "Formatting variable data" on page 952.
7. Add as many data fields as you need, following the same procedure.
8. Optionally, you can click **Options** at the bottom to specify how the result is inserted:
   - As **HTML**. HTML elements in the result are processed and displayed as HTML elements. For instance, `<b>this is bold</b>` will be displayed as **this is bold**. This is the default setting.
   - As **text**. This inserts the result as-is, meaning HTML tags and elements are displayed as text in the output. In this scenario, "<br>" shows up in the text and does not insert a line break. This is the preferred setting if the Helper produces plain text. It will be slightly faster than HTML since it avoids processing with an HTML parser.
9. Close the Wizard. The new Helper appears on the Scripts pane under Control Scripts.
10. Now all you have to do is **drag the Helper** from the Scripts pane into the template. Alternatively you can write the name of the Helper enclosed in double curly brackets: e.g. *{{MyHelper}}*. Do this anywhere where you want the result to be inserted.

Tip: When one of the included data fields is empty, the respective line, including the prefix and suffix, is skipped. The result will be shorter, causing the rest of the content to move up. If, in a Print context, you don't want the result of a Helper to be part of the text flow (for example, when a letter is going to be sent in an envelope with a window), put the expression that calls the Helper in a positioned box (see "Boxes" on page 823 and "How to position elements" on page 897).

Variable data in text: scripts and placeholders

OL Connect versions prior to 2022.2 used to insert variable data in a template via **scripts and placeholders** instead of expressions. Expressions look like this: *{{FieldName}}*. An expression works without a script. The advantage of expressions is that fewer scripts and sometimes no scripts at all are needed in a template.

Placeholders look like this: *@FieldName@*. If your template was made with a previous version, you will normally see a placeholder - and a script - appear when you drag a field from the Data Model into your template.

Do you prefer to work with expressions? Right-click the section in the **Resources** pane, select **Properties** and check the option **Evaluate Handlebars expressions**. For further instructions see "Variable data in text: expressions" on page 999.

However, if you want to get familiar with scripting in OL Connect, inserting variable data placeholders may be a good starting point. This will give you an understanding of how placeholders and scripts work.

Note: Before you can add variable data fields to the contents of your template you need to load - or create - a Data Model (see "Data Model pane" on page 1244), a data mapping configuration, ordata from a data file or database (see "Loading data" on page 926).

Note: Web templates are personalized just like any other template. There are a few extra possibilities, though; see "Using variable data on a Web page" on page 690.

How to prevent empty lines and extra spaces

You can insert placeholders and create the accompanying scripts via the **drag-and-drop** or **double-click** method, or using a **wizard**.

The **drag-and-drop** or **double-click** method is best for simple fields that do not need to be preceded or followed by a space, line break or text.
Are there empty fields in the data? And is the value of a data field preceded or followed by a space, line break or text in the template? Then it is better to use a **wizard**. Otherwise, empty data fields will cause empty lines and superfluous white spaces to show up in the text. You should also use this method for blocks of data, such as address blocks. The wizard makes it easy to format data differently, for example, display a number as a currency.

Both methods are described in detail below.

Tip: The address block wizard helps you insert an address block in which empty lines are automatically removed. See "Address Block" on page 955.

Inserting placeholders via drag-and-drop / double-click

An easy, quick and direct way to insert placeholders for variable data in the content is via drag-and-drop or a double-click.

1. Open the section you want to add the data field to.
2. Either:
    - **Drag and drop** a data field from the **Data Model** pane at the bottom right into the content of your template. To select and insert **multiple** data fields at the same time, press **Shift** or **Ctrl**, whilst selecting fields in the Data Model pane.

      Tip: Press the **Ctrl** key while dragging to wrap the placeholder(s) in an absolute positioned box (a **div**) at the cursor position. Each placeholder inside the box is wrapped in a Span, and each Span gets its own class. This method is particularly useful when the document mainly consists of a PDF used as the background image of a section (see "Using a PDF file or other image as background" on page 634).
    - Place the cursor in the template where you want the data to be inserted and then **double-click** the data field in the Data Model pane.

Note: This doesn't work for data in a detail table. Data in detail tables can be inserted in a Dynamic Table using the Dynamic Table wizard (see "Dynamic Table" on page 974).

What happens is that:

- A **placeholder** for the value of the data field shows up in the text. It looks as follows: @FIELDNAME@.
- A **script** appears in the **Scripts** pane at the bottom left.

Tip: Double-click the script to open the Text Script Wizard and add more fields to the same script. See "Using the Text Script Wizard" on the facing page.

The script replaces the placeholders in the content with the value of a data field in the current record.

Switch to the **Preview** tab at the bottom of the workspace to see the script in operation. The value of the corresponding data field in the first record appears instead of the placeholder, everywhere where the placeholder is found in the text. This value will be refreshed when you browse through the records in the Data Model pane.

When the output (the letter, email, etc.) is generated, the text script executes for each record in the record set, and each time it replaces the placeholders with the value of the field in the current record.

In the **Scripts** pane you can see that the script has a **name** and a **selector**. The **selector** is what a script looks for in the template.

- The drag-and-drop method automatically generates a script that is named after the data field (see the **Scripts** pane).
- By default the drag-and-drop method wraps the placeholder text in a Span. (Select the placeholder and switch to the **Source** tab to see the Span element.) The Span's **class** is used as selector. Note that any spaces that occur in a data field name will be removed from the script selector.

When you drag the same field from the Data Model to the content again, a second placeholder appears in the text, but no new script is added. The existing script will replace all placeholders that match its selector.

You could also drag a script with an ID and/or class selector from the Scripts pane into the template to insert an additional placeholder.

Tip: Drag the data field directly to the **Scripts** pane to create a script with a class selector, without adding a placeholder to the template. Holding the Alt key while dragging creates a script with a text selector.

Note: Connect versions older than 2020.2 used the placeholder text as the script's selector. Looking for text in a text is a less optimized operation and may impact output speeds in longer documents. For that reason placeholders are now by default wrapped in a Span and the class of that Span is used as the script's selector.

If you would like the drag-and-drop method to behave like it did in older Connect versions, you can change its behavior via the Preferences (**Window > Preferences > Editing**), or hold the **ALT** key while dragging to change its behavior on the fly.

For more tips to make a template generate output faster, see "Optimizing scripts" on page 1078.

Note: In placeholders, special characters (" \ / ' £ $ & @ ~ #) are not supported. If a placeholder with one of these characters is used as a selector, the script that is supposed to replace the placeholder will not work.

Using the Text Script Wizard

The Text Script Wizard can insert one or more data fields into your template, each with an optional prefix and suffix. It is recommended to use the Text Script Wizard for blocks of

data, such as address blocks, and when data fields can be empty or need to be formatted differently.

1. Create a new text script and open the Text Script Wizard. There are three ways to do this:
   - On the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **New Text Script**. A new script appears in the list. Double-click the new script to open it.
   - Select a word in the content. Right-click the selection and on the shortcut menu, choose **Text Script**.
   - Drag a data field into the template as described above, and then double-click the new script.

   The Text Script Wizard appears.
2. Change the name of the script to make clear what it does.

   > **Note:** Scripts can only have the same name when they are not in the same folder.

3. The **selector** defines what to look for in the template. The results can be replaced by the script.

   > **Tip:** Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

   - **Text**, for example: @lastname@, or {sender}. The text doesn't have to have any special characters, but special characters do make it easier to recognize the text for yourself. In the Text Script Wizard, click **Text** and type the text to find.

     > **Note:** A script made with the Text Script Wizard for a block of data already runs faster than a series of individual scripts, because it only has one selector. However, searching for text can be a lengthy operation, compared to searching for an element with an ID. When speed matters, select one of the two remaining options: **Selector** or **Selector and Text**. See also: "Testing scripts" on page 1074 and "Optimizing scripts" on page 1078.

   - An **HTML/CSS selector**:
     - HTML elements, such as a paragraph. In the Text Script Wizard, click **Selector** and type the HTML tag without the angle brackets, for example: **p**.

- HTML elements with a specific class. In the Text Script Wizard, click **Selector** and type the class name, including the preceding dot, for example: **p.green** for all paragraphs with the class 'green' or **.green** for all kinds of HTML elements that have the class 'green'. See "Styling and formatting" on page 879 for an explanation about CSS (Cascading Style Sheets).
- An HTML element with a specific ID. In the Script Wizard, click **Selector** and type the ID, including the preceding #, for example: **#intro**.

  > **Note:** Each ID should be unique. An ID can be used once in each section.

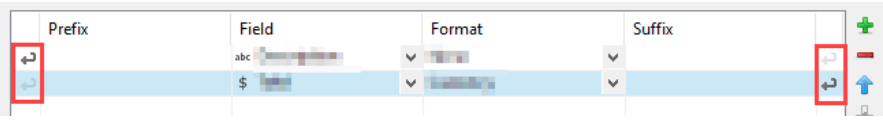
- Etcetera. See: MDN Web Docs - CSS Selectors for more selectors and combinations of selectors.

- A **selector and text**. This is text inside an HTML element (or several HTML elements) with a specific HTML tag, class or ID. In the Text Script Wizard, click **Selector and text** and type the selector and the text in the respective fields.

4. Click the downward pointing arrow in the first row in the column **Field**. Select a data field from the list that appears.
5. Add a **Prefix** and/or a **Suffix**. The prefix and suffix can contain text and/or HTML tags. If a field is empty, the prefix and suffix will be ignored, which means you can add line returns and static text, such as:

   - with a Number field, Prefix: *Your invoice* (one space at the end), Suffix: *is now ready to be viewed!*
   - with a field State, Prefix: *,* (comma then space).

   To add a single **line break**, activate the Line break button in either the first or last column.



   Alternatively, or to add an *extra* line break, you can add *
*, which is the HTML tag for a line break, to the Suffix. For example:

   - with a field LastName, Suffix: *
*

   > **Tip:** For a comma between fields, use the Prefix of the second field, if you don't want a comma when the second field has no value.

6. The Wizard allows you to reformat the data (for example, apply uppercase, apply thousand separators to numbers, etc.). Click the column **Format**, click the downward pointing arrow and select one of the formats. For an explanation of the formats see "Formatting variable data" on the facing page.
7. Add as many data fields as you need, following the same procedure.
8. Optionally, you can click **Options** to specify where and how the script inserts its results:
    - As **HTML**. HTML elements in the results are processed and displayed as HTML elements. For instance, `<b>this is bold</b>` will be displayed as **this is bold**. This is the default setting.
    - As **text**. This inserts the results as-is, meaning HTML tags and elements are displayed as text in the output. In this scenario, "<br>" shows up in the text and does not insert a line break. This is the preferred setting if the script produces plain text. It will be slightly faster than HTML since it avoids processing with an HTML parser.
    - As the value of an **attribute** of an HTML element. The selector of the script should be an HTML element. Which attributes are available depends on the selected HTML element. If the script's selector is an image (<img> element) for example, and the attribute is `src`, the script will modify the image's source. The script's results should be a valid value for the chosen attribute.

      Note: When checked, the option **Convert fields to JSON string** writes the results from the script into an attribute or text as a JSON string. This is useful for web contexts where a front-end script can read this value easily.
9. Close the Text Script Wizard and type the placeholder for the results of the script in the content of your template, or make sure that there is at least one element that matches the selector of the script.
10. Hover over the name of the script in the **Scripts** pane. In the workspace you will see which parts of the template are affected by the script. If the script produces an error, the error message will be displayed in a hint on the **Scripts** pane.

    Tip: When one of the included data fields is empty, the respective line, including the prefix and suffix, is skipped. The result of the script will be shorter, causing the rest of the content to move up or down. If, in a Print context, you don't want the result of the script to be part of the text flow (for example, when a letter is going to be sent in an envelope with a window), put the placeholder for the script

in a positioned box (see "Boxes" on page 823 and "How to position elements" on page 897).

Tip:

- OL Connect has a special wizard to help you insert an address block: "Address Block" on page 955.
  An alternative is to create an address block using a Handlebars text helper. This is described in a how-to: Creating an Address Block.
- To use only part of a data field, or to split the data, you will have to write a script.
- Are you curious to see the actual code of the script? Double-click the script to open the Script Wizard, and then click the **Expand** button to see the code.

  Caution: When you change an expanded text script and save it, it becomes impossible to edit the script using the Script Wizard again.

Formatting variable data

The way variable data are displayed in a template depends on two things. Firstly, the **locale** influences the way dates, times, numbers and currencies are formatted; see "Locale" on page 921. Secondly, there are functions to display a text in uppercase, apply thousands separators to numbers, etc.

The **Text Helper Wizard** (see "Using the Helper Wizard" on page 1003) and the **Text Script Wizard** (see "Using the Text Script Wizard" on page 948) let you select a format or enter a format mask for each field that the script adds to the template. This topic describes the options that the wizards offer, depending on the type of field.
Reopen the Helper or script by double-clicking it in the Scripts pane.

In an **expression**, e.g. {{LastName}} (see "Variable data in text: expressions" on page 999), you can type the desired format function before the field name, e.g. *{{upperCase LastName}}*. Functions in an expression are called "Helpers". For an overview of the available Format Helpers see "Format Helpers" on page 1015.

The available formatting functions depend on the **data type** of the corresponding field in the Data Model. In a data mapping configuration you can set the data type of each field. When you open a data file or database without a data mapping configuration, all fields are initially text fields (fields of the type `string`). To change the type of such a data field, right-click it and select **Set Type**.

When the software warns about a mismatch between the type of a field in the wizard and in the current data model, click the Refresh Types button to update field types in the wizard to match the types in the data model.

To learn how formatting can be applied to **values in a Dynamic Table**, see: "Formatting values" on page 987.

You could also format data in a **script** using the `formatter`; see "Standard Script API" on page 1481.

Date

Dates and strings that follow the ISO (2024-04-02) or UTC data string format can be displayed as long, medium and short dates with different time displays. There are quite a few presets, but you can also enter a custom format mask.

- **Custom Pattern** opens a dialog in which you can enter a custom pattern the date (and, optionally, the time). The dialog shows an example of a date formatted according to the given pattern.
  Do not put the pattern in quotes. For possible patterns see "Date and time patterns" on page 1491.

  > **Note:** The Custom Pattern option is only available when the type of the field has been set to Date in the data mapping configuration and the field contains a valid date.

- **Default Date** displays the date with the default pattern for the current Locale.
- **Short Date** displays the day, month and year in two digits each, for example **01.04.16**.
- **Medium Date** displays the day and month in two digits each and the year in four digits, for example **01.04.2016**. (This is also the value of the Default Date.)
- **Long Date** displays the day as a number, the month's full name and the year in four digits, for example **1. April 2016**.
- **Short Time** displays a time in hours and minutes in two digits each, for example **00:00**.
- **Medium Time** displays a time in hours, minutes and seconds in two digits each, for example **00:00:00**. (This is also the value of the Default Time.)

- **Long Time** displays a time in hours, minutes and seconds in two digits each, and adds a time zone, for example **00:00:00 EDT**.
- **Short Date/Time** displays the date as a short date and the time as a short time, for example **01.04.16 00:00**.
- **Medium Date/Time** displays the date as a medium date and the time as a medium time, for example **01.04.2016 00:00:00** (This is also the value of the Default Date/Time.)
- **Long Date/Time** displays the date as a medium date and the time as a medium time, for example **1. April 2016 00:00:00 EDT**.

Font style

Text originating from variable data can be displayed in uppercase, lowercase or proper case.

- **Uppercase** transforms all characters to uppercase.
- **Lowercase** displays transforms all characters to lowercase.
- **Propercase** transforms the first character of each word to uppercase and all other characters to lowercase.

Numbers and currencies

Numbers, and strings existing of digits, can be displayed as a number with a certain formatting or as an amount of money. There are a few presets, but you can also type a format mask.

Spaces in a number or amount of money are transformed into non-breaking spaces to ensure that it is not split onto different lines.

- **Custom Pattern**: allows you to enter a custom format mask. For example, the pattern `000000` means that the number should count six digits; leading zeros are added to numbers shorter than six digits. For an overview of pattern symbols see "Number patterns" on page 1511 and http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html. In the data mapping configuration, set the field type to Integer, Float or Currency. When you open a data file or database, all fields are text fields (fields of the type `string`).
- **Standard**: displays the number with the default pattern and symbols for the current Locale.
- **Grouped** displays a number with three decimal places and sets the thousands separator for the value based on the current locale; see "Locale" on page 921.

- **Currency** displays a number as an amount of money, with a thousands separator and rounded to two decimal places, based on the current locale; see "Locale" on page 921.
- **Currency no symbol** does the same as Currency, but omits the currency symbol.
- **Leading zero** adds a leading zero to a floating value between 0 and 1. This format is only available for fields that contain a `float` value. Note that when you open a data file or database without a data mapping configuration, all fields are of the type `string`.
- **∑ (Sum)** and **∑↑ (Sum Up)** are used in Dynamic Tables in a Print context. **∑** is for transport rules at the end of a page and **∑↑** shows the subtotal of the previous page.

Address Block

When you create a template, the first piece of content you add is likely to be the address. Address blocks are essential in a variety of documents, such as business letters, cover letters, invoices, quotes, postcards, policy documents, and contracts.

The Address Block wizard makes it easy to create an address block built according to country standards. It allows you to map fields from the data model to elements in an address block format of your choice.

If data is missing because a record is incomplete, empty lines or white space in the final address block are automatically removed.

For specific countries the wizard provides options to render postal barcodes, for example the Dutch KIX barcode.

Before inserting an address block

You can only insert an address block if there is an open data mapping configuration, Data Model, data file or database; see: "Loading data" on page 926.

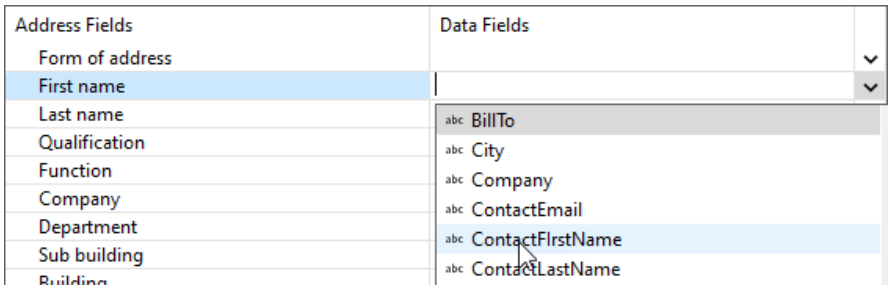
How to use the Address Block wizard

To insert an address block with the wizard:

1. Select **Insert > Address Block**, from the menu, or click the Address Block icon on the toolbar.
2. Select an **Address format**. Currently, address block formats are provided for the following countries: Australia, Belgium, Canada, France, Germany, Italy, Malaysia, Netherlands, Spain, United Kingdom, and United States. Additional address formats for the United States support the use of the USPS Ancillary Service Endorsement (ASE)

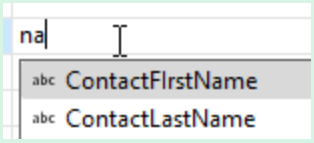
and can place an IMb barcode below or above the address.
If you wanted a region or postal service specific format that is not in the list, you can select the **Generic** format.
Another alternative is to create your own address format. (See "Creating an address format" on page 957.)

3. The fields in the **Address Fields** column are organized roughly in the same way they will appear in the address block.



To **connect** a data field to an address field, click the **Data Fields** column, then select the field that provides the data for that address field.

Tip: To shorten the list, start typing (part of) the name of the data field you are looking for.



The example of the address block at the bottom left will be populated with data from the current record in the currently open dataset.

To **remove** the connection between a data field and an address field, click the field in the Data Fields column, then press the Delete or Backspace key and navigate out of the field.

4. Click the checkbox **Align text to bottom** if you want the address to be aligned to the bottom of the address block.
5. Finally click **OK** to insert the address block.

The Address Block wizard remembers your settings - per address format - and will attempt to link the same address fields and data fields the next time you create an address block.

Changing the address block

Once an address block is added to the template, you can preview it, move it, resize it, and change the data fields it is filled with.

- To see the address block filled with data, toggle to the **Preview** tab at the bottom of the workspace (or select **View > Preview View** on the menu).
- To **reopen** an address block with the Address Block wizard, right-click it and select **Address Block** from the popup menu.
- To **move** the box that contains the address block, click the edge of the box and drag it.

  Alternatively, click the address block and set its top and left on the Attributes pane.
- By default, the box that contains the address block is 95x38 mm. To **resize** it, click and drag one of the handles of the box.

  Alternatively, click the address block and set its width and height on the Attributes pane.

Styling an address block

To style an address block that you inserted with the wizard, right-click the address block box, select **Box...** and make the appropriate settings in the "Box Formatting dialog" on page 1138.

Alternatively, select the address block box, enter an ID on the Attributes pane, and use that ID to create CSS rules for this address block. See "Styling templates with CSS files" on page 881.

Creating an address format

If an address format is not yet available for your country, region, or postal service specific format, you could create a custom address format. This topic explains how to do that.

1. Create a text file and save it in the folder:

   `C:\ProgramData\Objectif Lune\OL Connect\Address Blocks`

   Change the file's extension into **.hbs**. This essentially turns the file into a Handlebars template.

The name of the file, without the extension, will now appear in the Address Block wizard, in the address format drop-down.

2. Open the file you created and write a Handlebars expression for each address field you want to appear in the address block.

   The expression for an address field must contain:

   - The prefix: **OLC**, followed by two digits: **00 - 99**. The digits determine the order of the address fields in the Address Fields column. OLC00 comes first, OLC99 comes last.
   - The address field name. If the name contains a space, surround the name with square brackets.

   Example: {{OLC10firstname}} {{[OLC10last name]}}

   If the address field name (without prefix and digits) happens to be the same as one of the standard address field names, it will be translated. For example, **firstname** will appear in the address field list as **First name** if the display language of the Designer is English.

3. Put the address fields that should be on one line in the address block into a paragraph by surrounding them with HTML paragraph tags: … .

   Example: {{OLC10firstname}} {{OLC11middlename}} {{OLC12lastname}}

4. To make sure the address block looks good even if data are missing, you will probably need to add some logic. You can do this with Handlebars Helpers; see "Using functions in expressions: Helpers" on page 1009.

   For example, you may want to have two spaces between the first and last name, but only if both names are present in a record. The line in the following example has that logic.

   Example: {{OLC10firstname}}#if (and {{OLC10firstname}} {{OLC13lastname}}) {{OLC12lastname}}

   Note: If none of the expressions in a paragraph return results, and the paragraph has no other content, the address line is empty and is automatically deleted. Empty lines do not appear in the output.

Showing content conditionally

One way to personalize content is to show or hide one or more elements depending on the value of a data field or runtime parameter. For example, a paragraph written for Canadian customers could be hidden when the recipient of the letter is not living in Canada, if that can be derived from the data.

Showing or hiding elements using the Conditional Content Script wizard

The Conditional Script Wizard helps you to show or hide an element – a paragraph, image or other HTML element - based on the value of one or more data fields and runtime parameters. For example, you could check whether the data field 'State' is 'Equal To' the value 'British Columbia' or 'Québec', to include a paragraph only for recipients in those states.

To open the Conditional Script wizard:

- Select one or more elements, right-click and choose **Make Conditional**.
- Alternatively, click the black triangle on the **New** button on the **Scripts** pane at the bottom left of the window, and click **Conditional Content Script**.

In the wizard, first make the following **settings**.

1. **Rename** the script so that it reflects what the script does.

   > **Note:** Scripts can only have the same name when they are not in the same folder. (See "Managing scripts" on page 1071.)

2. Type a **Selector**. The selector selects one or more pieces of text or elements from the template, so that the conditional content script can hide or show those pieces.
   An **ID** (for example: #conditional-script) is best if you want to show or hide one element only.
   Use a **class** selector (for example: .conditional) if the script should show or hide more than one element. See "Selectors in OL Connect" on page 1083 for further explanation on selectors.

   When you start the Conditional Script Wizard by right-clicking an element, the Selector field is pre-populated with the element's ID and class (or classes). If the element didn't have an ID or class, it gets a new ID (or class, in case multiple elements were selected) that is generated automatically.
   If you'd want to change the selector later, you can do that after reopening the script (double-click the name of the script in the **Scripts** pane).

3. Set the **Action**: use the drop-down to select whether to **Show** or **Hide** the element when the condition below is **true**. When the condition evaluates to false, the opposite action is performed.

Once these settings have been made, the **condition** can be build.

The ✚ **Add** button adds a rule that evaluates a **data field** to a group.
To add either a **group**, or a rule that evaluates a **runtime parameter**, click the downward pointing arrow next to this button and select **Group** or **Parameter Rule**.

> **Note:** Creating a parameter rule is only possible if the template contains runtime parameters; see "Runtime parameters" on page 608.

The rule(s) apply at a Group level. There may be one rule or many rules at the same level, and there may be groups within groups, providing the ability to create quite complex nested logical structures.

Group

A group consists of one or more rules with a logic operator. Four logic choices are available at the Group level. These are:

- **All of the following**.
  This equates to the logical operator (... AND ...).
  If *all* of the associated criteria are met, then this group resolves to TRUE.
- **Any of the following**.
  This equates to the logical operator (... OR ...).
  If *any* of the associated criteria are met, then this group resolves to TRUE.
- **Not all of the following**.
  This equates to the logical operator (NOT(... AND ...)).
  If *any (but not all)* of the associated criteria are met, then this group resolves to TRUE.
- **Not any of the following**.
  This equates to the logical operator (NOT (... OR ...)).
  If *none* of the associated criteria are met, then this group resolves to TRUE.

The top level is always a group. A group can contain one or more Rules and/or Groups.

Rule

A single rule evaluates one data field or runtime parameter using a single operator or function and a value (the "operand").

To construct a rule:

1. Click the downward pointing arrow next to **Field**, then select the data field or runtime parameter that should be evaluated.
2. Select the **operator**. The downward pointing arrow expands the list of operators with which the data can be evaluated.

   The options available will depend upon the Data Field or Parameter type (they are "type aware"), but most have at least "*is equal to*" and "*is not equal to*" as an option. For the complete list of options per data type see "Conditional Content script dialog" on page 1197.

   > **Tip:** Data fields in data loaded from a file - i.e. not via a data mapping configuration - are of type String (text) by default. To change the data type of such a field, right-click it in the Data Model and select **Set Type**.

3. Enter the **Value** that should be used for the conditional check.
   - Values in Strings are **case sensitive** by default; you can click the button next to the value to make them case insensitive.
   - **Dates** should be entered in ISO standard notation (yyyy-mm-dd) (see ISO 8601). It is best to select the date using the date selection option.

It isn't possible to enter more than one value in one rule. Comparisons to multiple values can be made by combining rules in a group.
For example, an "*after*" and "*before*" rule combined in one "all of the following" group can be used to check whether a date falls *between* two dates.

To save the script, click **Apply** or **OK**.

The selected action will be performed if the joint groups and rules evaluate to **true**.
If, conversely, the condition evaluates to false, the opposite action will be performed.

To see the result, toggle to the **Preview** tab at the bottom of the workspace (or select **View > Preview View** on the menu).

> **Note:** If a condition cannot be made with the wizard - for example, one that compares a data field to a runtime parameter value - you can click **Expand** and edit the code of the script (see "Writing your own scripts" on page 1064). But be careful: once the script has been altered, there is no going back to the wizard.

Showing or hiding several elements with one conditional script

To apply one conditional content script to several elements, a CSS **class** or HTML element needs to be used as the selector of the script. When you select multiple elements and

create a new conditional content script following the actions described in "Showing or hiding elements using the Conditional Content Script wizard" on page 959, a CSS class will be applied automatically.

If you have created the script yourself, or if you want a script to apply to more elements later on, you have to do this manually.

1. Double-click the conditional script in the **Scripts** pane to reopen it.
2. Make sure that the selector is a CSS class (for example, .male) or an HTML element with a certain CSS class (for example, p.male). See "Selectors in OL Connect" on page 1083 for further explanation on selectors.
3. Apply the same CSS class to all elements that should be shown or hidden under the condition that you have set in the conditional script. Click each element and type the class (without the preceding dot) in the **Class** field.

Showing or hiding a text selection

When you right-click on an element and make it conditional, the element as a whole will be made conditional. This happens even when you select a few words in a paragraph and right-click those words; the paragraph as a whole will be made conditional.
It is, however, possible to show or hide a part of a paragraph.

If the text that you want to show or hide is short, you could use the **Selector and Text** option in the Conditional Content wizard and copy the text into the **Text** field. This option uses the specified selector as a trigger for the script. The script applies to all instances of the text found within the specified selector.

Searching for text can be a lengthy operation, compared to searching for an element with an ID. If performance is an issue, of if the search text is long, a better alternative is to wrap the text in a span element first.

1. Select the part of the text that you want to make conditional.
2. Right-click the selected text and click **Wrap in span**.
3. Type an **ID** and/or a **class**. An ID is fine if this is the only thing that should be shown or hidden on a given condition. Use a class if there is more that should be shown or hidden on the same condition.
4. Start creating a conditional content script from the **Scripts** pane. Use the ID or class as the selector of the script. See "Showing or hiding elements using the Conditional Content Script wizard" on page 959.

Conditional Print sections

You can include or exclude entire Print sections from the output, depending on a field's value. This can be done using the Conditional Print Section Script Wizard, described below. Alternatively you could write a Control Script (see "Control Scripts" on page 1098).

Including or excluding Print sections using the Conditional Print Section Script wizard

To open the Conditional Print Section Script wizard:

- Right-click the section and click **Make Conditional**.
- Alternatively click the black triangle on the **New** button on the **Scripts** pane at the bottom left of the window, and click **Conditional Print Section Script**. Double-click the new script to open the Conditional Print Section Script wizard.

In the wizard, first make the following **settings**.

1. **Rename** the script so that it reflects what the script does.

   Scripts can only have the same name when they are not in the same folder. (See "Managing scripts" on page 1071.)
2. Select the section you want to put a condition on.
3. Set the **Action**: **Print** or **Skip** that is performed when the condition below is **true**. The opposite action is applied when the condition returns false.

Once these settings have been made, the **condition** can be build.

The ✚ **Add** button adds a rule that evaluates a **data field** to a group.
To add either a **group**, or a rule that evaluates a **runtime parameter**, click the downward pointing arrow next to this button and select **Group** or **Parameter Rule**.

> **Note:** Creating a parameter rule is only possible if the template contains runtime parameters; see "Runtime parameters" on page 608.

The rule(s) apply at a Group level. There may be one rule or many rules at the same level, and there may be groups within groups, providing the ability to create quite complex nested logical structures.

Group

A group consists of one or more rules with a logic operator. Four logic choices are available at the Group level. These are:

- **All of the following**.
  This equates to the logical operator (... AND ...).

If *all* of the associated criteria are met, then this group resolves to TRUE.

- **Any of the following**.
  This equates to the logical operator (... OR ...).
  If *any* of the associated criteria are met, then this group resolves to TRUE.
- **Not all of the following**.
  This equates to the logical operator (NOT(... AND ...)).
  If *any (but not all)* of the associated criteria are met, then this group resolves to TRUE.
- **Not any of the following**.
  This equates to the logical operator (NOT (... OR ...)).
  If *none* of the associated criteria are met, then this group resolves to TRUE.

The top level is always a group. A group can contain one or more Rules and/or Groups.

Rule

A single rule evaluates one data field or runtime parameter using a single operator or function and a value (the "operand").

To construct a rule:

1. Click the downward pointing arrow next to **Field**, then select the data field or runtime parameter that should be evaluated.
2. Select the **operator**. The downward pointing arrow expands the list of operators with which the data can be evaluated.

   The options available will depend upon the Data Field or Parameter type (they are "type aware"), but most have at least "*is equal to*" and "*is not equal to*" as an option. For the complete list of options per data type see "Conditional Content script dialog" on page 1197.

   > **Tip:** Data fields in data loaded from a file - i.e. not via a data mapping configuration - are of type String (text) by default. To change the data type of such a field, right-click it in the Data Model and select **Set Type**.

3. Enter the **Value** that should be used for the conditional check.
   - Values in Strings are **case sensitive** by default; you can click the button next to the value to make them case insensitive.
   - **Dates** should be entered in ISO standard notation (yyyy-mm-dd) (see ISO 8601). It is best to select the date using the date selection option.

It isn't possible to enter more than one value in one rule. Comparisons to multiple values can be made by combining rules in a group.
For example, an "*after*" and "*before*" rule combined in one "all of the following" group can be used to check whether a date falls *between* two dates.

To save the script, click **Apply** or **OK**.

The selected action will be performed if the joint groups and rules evaluate to **true**.
If, conversely, the condition evaluates to false, the opposite action will be performed.

To see the result, toggle to the **Preview** tab at the bottom of the workspace (or select **View > Preview View** on the menu).

Note: If a condition cannot be made with the wizard - for example, one that compares a data field to a runtime parameter value - you can click **Expand** and edit the code of the script (see "Writing your own scripts" on page 1064). But be careful: once the script has been altered, there is no going back to the wizard.

Tip: Take a look at the Resources pane: on each Print section that is affected by a Conditional Print Section script a small decorator appears if it is skipped with the current data record.

Cloning sections

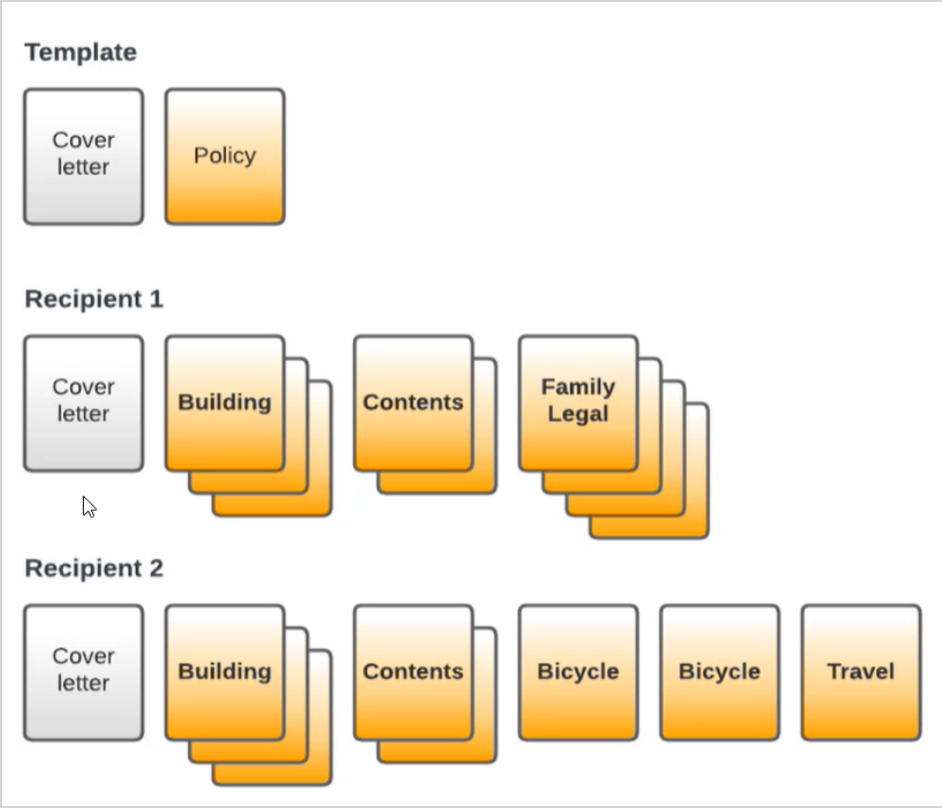
Print sections can be cloned (copied) on the fly, so that documents in the output can have a **variable number of sections**, based on data. Each clone can have its own background image and variable data.

Types of documents for which cloning sections can be useful are, for example:

- **Insurance policies, loan statements**. The number of policies and statements can vary per customer.
- **Under-copies** of delivery notes or picking lists. Under-copies may need to contain additional information.
- **Textbooks and course material**. These usually consist of a varying combination of chapters. Chapters that are available as multipage PDFs can be set as section backgrounds. The number of pages in each section will match the number of (selected) pages in the corresponding PDF.
- **Exam information** (answering sheets). Student-specific exam information can be set as a section background and supplemented with personalized content.

- **Pick lists; box and shipping labels**. Each of the section clones can display details about the contents of a single box in a shipment or a single item on a pick list.
- **Postal payment slips**. These are typically attached to a reminder letter.

In the case of insurance policies, for example, a template would only need to contain a cover letter section and a policy section. For each recipient, the policy section would be cloned as many times as necessary, i.e. once for each insurance the recipient has.



Cloning a section using the wizard

A Control Script that clones a section can be created with a wizard. This topic explains how to use the wizard.

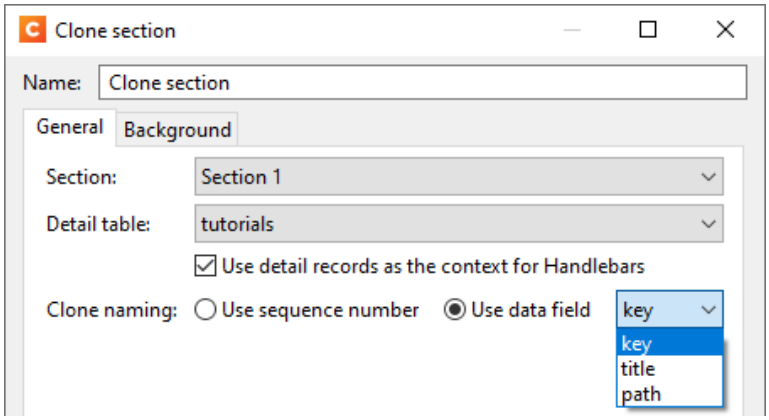
The code in a Clone Section script, and how to enhance it, is covered in more detail in another topic: "Clone Section scripts" on page 1108.

1. Right-click the section on the **Resources** pane and select **Clone Section...**

   Or, on the **Scripts** pane at the bottom left:

- Click the black triangle on the **New** button and select **Clone Section Script**.
- Or, right-click the **Control Scripts** folder and select **New** > **Clone Section script**.

The wizard appears.



2. On the **General** tab, change the name of the script so that it reflects what the script does.

   Note: Scripts can only have the same name when they are not in the same folder.

3. Select the **section** that must be cloned.
4. Select a **detail table**. As many clones are created for each recipient as there are records in that detail table.
5. Normally, Handlebars expressions in sections and clones retrieve data from the current main record. To make Handlebars expressions work on the detail record that was used to create the clone, rather than on the main record, select **Use detail records as the context for Handlebars**. The benefit of this is explained in "Handlebars expressions in section clones" on page 969.

   Note that with this option selected, Handlebars expressions can still access the main record and any detail tables indirectly (via @root, see "Handlebars expressions in section clones" on page 969).
6. Chose the Clone name in the **Clone naming** section. Selection is between **Use Sequence number**, or **Use data field**, which allows selection of a field from the database.

   Note: When a Template is saved to a version pre 2025.1, the Clone name will default to **Sequence Number**.

7. (Optional:) Switch to the **Background** tab if you want to apply PDF background images to the clones. See "Setting a background image for section clones" on the facing page.
8. Click **OK**.

> **Tip:** If the functionality required goes beyond what the wizard provides, you can click the **Expand** button to access the script code. The code for a Clone Section script, and how to enhance it, is covered in more detail in another topic: "Clone Section scripts" on page 1108.
> Note that once the expanded script dialog is closed, this action cannot be undone.

When browsing through the records in **Preview** mode, you will now see the section clones on the Resources pane. Double-click a section clone to open it.

> **Note:** Section clones are only visible on the Preview tab. The maximum number of section clones visible is 50 by default, to prevent the Designer from becoming less responsive. This number can be changed; see: "Editing preferences" on page 1044.

By default, clones have the same name as their source section with an index in round brackets.

> **Example:** Source: "Section 1"
> Clone names: "Section 1 (1)", "Section 1 (2)", "Section 1 (3)", ...

Clones can be renamed in code, after expanding the script. See "Renaming a clone" on page 1111.

Setting a background image for section clones

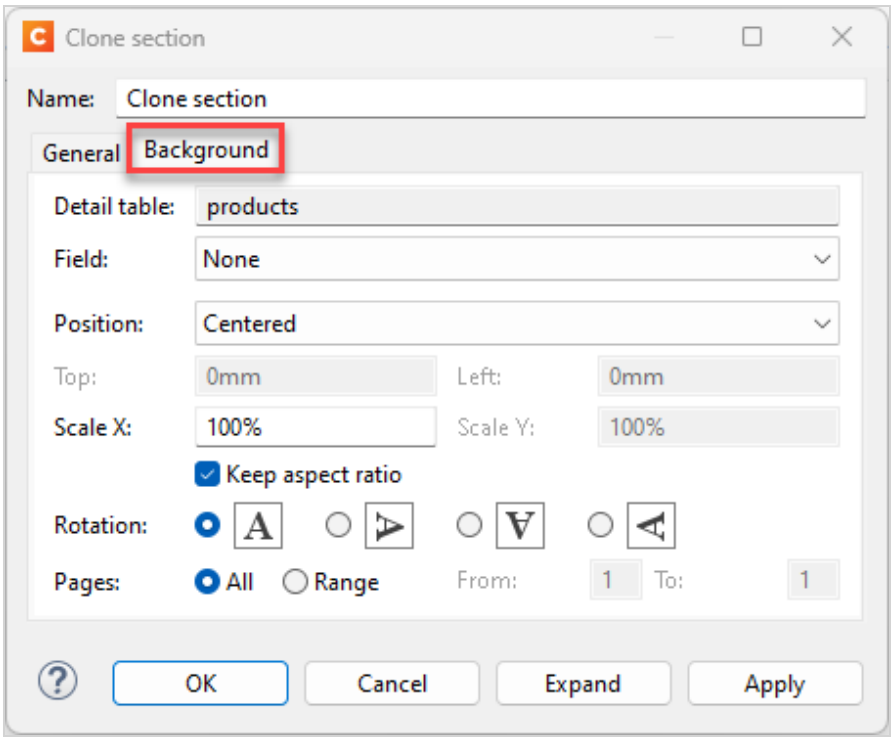
Section clones automatically get the same background image as the original section. If each section clone needs a different background image, this is easily set in the section clone wizard.

> **Note:** The wizard expects the path to a background PDF to be specified in the detail table record on which the clone is based. If the path to a PDF is not specified in the data, follow the steps below, but select a random field in step 3. Then expand the script to construct the path in code. An example is given in the topic "Clone Section scripts" on page 1108.

1. Open the Section Clone wizard; see "Cloning a section using the wizard" on page 966.

   If you've already added the Section Clone script with the wizard, reopen the wizard by double-clicking the script. Or, right-click the section and select **Section Cloning...**.

2. Switch to the **Background** tab.



3. Select the field that holds the path to a PDF. What the path should look like depends on the location of the image; see "Resources" on page 606.
4. Select the desired position, scaling, rotation and page range. For an explanation of these options, see "Settings for a background image" on page 636.
5. Click **OK**.

   Or, click **Expand** if you need to change the script, for example if the path to the PDF needs to be constructed in the code (see "Assigning a background PDF to a section clone" on page 1110).

Handlebars expressions in section clones

The easiest way to personalize section clones is by using Handlebars expressions and partials in the original section (see "Handlebars expressions" on page 1005 and "Partials" on page 1027).

In section clones, Handlebars expressions will either work on the detail record that was used to create the clone, or on the main record like they normally do. This depends on whether the **Use detail records as the context for Handlebars** option was checked in the Section

Clone wizard (step 5 in "Cloning a section using the wizard" on page 966).
For example: if the option was checked, the expression {{Company}} will display the value found in the Company field in the detail table record. If the option was unchecked, the same expression {{Company}} accesses the Company field in the main record.

Note that even if they work on the detail record, Handlebars expressions still have access to the entire record:

- **@root** represents the top level of the data.
- With **.** or **/** you can navigate **down** into the record and access fields and detail tables.

(See "Accessing data at different levels" on page 1006.)

Example: {{@root.Company}}
This expression accesses the Company data field at the root level of the record.

Example: {{@root.Products[0].ProdNumber}}
This expression gets the ProdNumber data field from the first record in the Products detail table.

Retrieving data from the meta property

The **@meta** helper for Handlebars gives access to information stored in the section's `meta` property (see "meta entries in section clones" on page 1110).

Just like with @root, you can use **.** to access fields in the `meta` property.

Example: {{@meta.detail-table}}
This expression returns the name of the detail table that contains the record for which the clone was created.

Example: {{ @meta.detail-table-record}}
This expression returns the index of the current detail table record.

Example: {{+myPartial (lookup @root.policies @meta.detail-table-record)}}
This expression loads a partial and passes the current detail table record to it. To find the current detail record, the `lookup` helper is used (see lookup in the Handlebars documentation). It is given a detail table called 'policies' and the record index set in the clone's `meta` property.

Tip: The Section Clone script can easily be changed to add more fields to the meta property. See "meta entries in section clones" on page 1110.

Accessing detail table data for section clones in scripts

> **Note:** Standard scripts are repeated for every section. This includes clones.

In scripts that apply to section clones that are based on a detail table, there are two ways to get access to data in the corresponding detail table record.

1. Use the detail table name and index

The Section Clone script automatically adds information to the `meta` property of each clone. That information can be used in scripts to retrieve data from the corresponding record in the detail table. The two `meta` fields that you need are:

- **detail-table** - Name of the detail table
- **detail-table-record** - The index of the clone's record in the detail table

These two meta fields can be accessed in code via `merge.section.meta["detail-table"]` and `merge.section.meta["detail-table-record"]`.

Here is a simple example. The script's selector is h1. The script adds the text found in the `summary` detail table field to the heading.

```
let detailTable = merge.section.meta["detail-table"]
let detailRecordIdx = merge.section.meta["detail-table-record"]
if (detailTable in record.tables) {
    let detailRecord = record.tables[detailTable][detailRecordIdx]
    let summary = detailRecord.summary
    results.text(summary)
}
```

> **Tip:** The **in** operator returns true if the specified property is in the specified object or its prototype chain.

2. Use the `meta` property

If many scripts need to have access to the same data field, it can be worth the effort to modify the Section Clone script by adding an entry to the clone's `meta` property and writing the value of a specific field in the detail table record into that entry. By default, Standard Scripts will then have access to that value via the `meta` property.

How to write data into the `meta` property and how to access that data in a Standard Script is described in another topic, see: "meta entries in section clones" on page 1110.

Targeting individual section clones

When working with a variable number of section clones, it is less obvious to write scripts and styles that target individual section clones. By loading Handlebars partials dynamically,

you can customize both the content and the layout of individual section clones. Give elements in partials a class and write CSS for them. (See "Partials" on page 1027 and "Styling templates with CSS files" on page 881).

However, if it is necessary to target a specific clone, the selector to use is: [section="*name of the clone*"].

This selector can be used in CSS as well as in Standard Scripts.

Example: This script changes the color of all h1 headings in the section named "Section 1 (1)":
Selector: [section="Section 1 (1)"] h1
Script: `results.css('color','red');`

Note: The selector is case sensitive.

Dynamic images

Dynamic images are called dynamic because they are switched, depending on the value of a data field. This way, a template can be adjusted to different customers.

This page describes how to insert dynamic images using the Dynamic Image Script Wizard.

Otherwise, adding dynamic images requires a self-made script (see "Writing your own scripts" on page 1064, "template" on page 1613 and "ImageResource" on page 1617).

Adding dynamic images

Dynamic images can be added to the template easily if you have:

- One or more data fields that contain values on the basis of which the images can be switched.
- An appropriate image for each case. All files should be of the same type and they need to be stored in the same location (the **Images** folder on the **Resources** pane, a folder on disk, or online). It is important that they are named after the various possible values of the related data field.

To start, add one image to the template. See "Adding images" on page 854.

Next:

1. Select the image.
2. Use a Handlebars expression in the **Source** field on the Attributes pane. (See "Variable data in text: expressions" on page 999.)

Alternatively, you can use the Dynamic Image Script Wizard:

1. Right-click the image and click **Dynamic Image**. Or select the image and click **Source** (not the field, but the label before the field) in the Attributes pane.
   The Dynamic Image Script Wizard opens. The image's ID is used as the script's selector. If the image did not have an ID, it is automatically generated.
2. Select the location of the files:
   - Select **Resources** if the images reside in the **Images** folder on the **Resources** pane.
   - **Folder on disk** refers to a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder.
     As an alternative it is possible to enter the path manually. You can give a local path (e.g. C:\Images\) or use the "file" protocol.

     The complete syntax of a fully qualified URL with the "file" protocol is: `file://<host>/<path>`. Note: if the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: `file:///c:/resources/images/`.
   - **Online** requires you to give a specific web address (for example, http://www.mysite.com/images/).

   The Dynamic Image Script Wizard composes the file names (including the path) based on the selected **location**, the **prefix**, the value of a **data field**, and the **suffix**. The variable part of the file names is the value of the data field(s) in the **Field** column. The prefix and suffix are meant to contain static parts of the file names.

   Click the first field in the column **Field**, and then click the downward pointing arrow. Select the data field to be evaluated.

   If you want the file name to be composed of the value of several data fields, simply click in the next row or click the **Add** button. This adds a row. Note that the rows will be concatenated to compose one file name. Only the last suffix should contain the file extension.
3. The resulting file name, including the path and file extension, is assigned to the **src** (source) attribute of the image. You can click **Options** to verify this.
4. Click **Apply** or **OK**. Now click the **Preview** tab and browse through the records to verify that the script works as expected.

The dynamic images feature can be used to insert dynamic signatures. This feature is put to use in the Designer walk-through: "Creating a letter" on page 197.

Note: An image with an unknown file extension is represented by a red cross in the output, but no error is logged unless the image refers to a local file that does not exist on disk.
Image file extensions that the software recognizes are: AFP, BMP, EPS, GIF, JPG/JPEG, PCL, PDF, PNG, PS, SVG, and TIF/TIFF.

Editing a Dynamic Image

To edit dynamic images added to the template earlier, right-click the image, or the space reserved for the dynamic images. Then click **Dynamic Image** to open the Dynamic Image Script Wizard again.
Alternatively you may double-click the respective script on the Scripts pane.

Dynamic Table

A Dynamic Table (previously also called a **detail table**) is an essential element in invoice templates. A Dynamic Table is different from a standard table in that it has a **variable** number of rows. In a Print context it will automatically overflow into as many pages as necessary to output all rows and it can display subtotals in its header and footer.

Dynamic Tables are only available if the loaded record set or data mapping configuration contains transactional data in one or more **detail tables** (see "Loading data" on page 926). Depending on the data, a Dynamic Table can also contain data from **nested detail tables**.

Note: Dynamic Tables cannot be used on a Master Page (see "Master Pages" on page 648).

Creating a Dynamic Table

To create a Dynamic Table:

1. Open the Insert Dynamic Table dialog. There are several ways to do that:
    - Double-click the name of the detail table in the Data Model.
    - Drag the data field that contains the name of the detail table from the Data Model into the template.
    - On the menu select **Insert > Table > Dynamic**.
    - On the toolbar, click the **Insert dynamic table** button.
2. Enter the table's desired **attributes**:

- **ID**: A unique identifier for the table. IDs are used to access the table from scripts and as CSS selectors for style rules.
- **Class**: A class identifier for the table. Classes can be shared between elements and are used to access the table from scripts and as CSS selectors for style rules.
- **Detail Table**: Use the drop-down to select which detail table will supply the data to display within the Dynamic Table.
- **Width**: Enter the width of the table (e.g.100%).
- Use the **Insertion Point** drop-down to select where to insert the table.
    - **At cursor position** inserts it where the cursor is located in the template.
    - **Before element** inserts it before the HTML element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be before the tag.*
    - **After start tag** inserts it within the current HTML element, at the beginning, just after the start tag.*
    - **Before end tag** inserts it within the current HTML element, at the end, just before the end tag.*
    - **After element** inserts it after the element in which the cursor is currently located. For example if the cursor is within a paragraph, the insertion point will be after the end tag of the paragraph ( ).*
    - **Replace** inserts it in place of the currently selected element. (This option is not available when inserting content in a Master Page.)

* If the current element is located inside another element, use the **Elements** drop-down to select which element is used for the insertion location. The list displays every element in the breadcrumbs, from the current selection point down to the root of the body.

Note: HTML has some restrictions as to which types of elements are allowed as children of other elements. For example, a paragraph element is not allowed to have children that are block level elements - like a Div or a Table. If inserting content at the selected location would produce invalid HTML the final result may be different than expected. For example, when you insert a Div into a paragraph, the paragraph gets split in two. This means you end up with two paragraphs with the Div in between.

3. Click **Next**. The **Dynamic Table Builder** appears.
   At this point the table has one row, which is based on the selected detail table. You can add, delete and reorder **rows** and **fields**.

   Fields - cells in the table - correspond to **data fields** in a (nested) detail table.

   - To add a field, select a data field from the drop-down next to the row's **Add Field** button. Or click the **Add Field** button itself to add an empty field.

     > **Tip:** You can add a barcode to a field after inserting the Dynamic Table. See "Adding barcodes to a Dynamic Table" on page 770.
     > Empty fields are also useful for positioning cells underneath each other to make them look like columns.

   - The **Synchronize fields** button refills a row with all the data fields of the detail table to which the row is associated.
   - Drag a field up or down to change the order of the cells in the table row.
   - Use the **Format** drop-down to select how the value must be formatted. Which options are available depends on the type of the data field (see "Data types" on page 436). For an explanation of the options see "Formatting variable data" on page 952.
   - The **Text alignment** setting at the bottom determines how text alignment settings are added to cells in the table.
     - **HTML defaults**: The text-alignment is not set in this dialog. Text in cells will be aligned in the way that is standard for HTML.
     - **Via inline styles**: The text-alignment is set via the `style` attribute of the cells. This is also called local formatting. (See also: "Styling and formatting" on page 879.)
     - **Use CSS utility classes**: The cells get a class for which the default style sheet for Dynamic Tables (default_table_styles.css) has a style rule. For more information see "Utility classes for text in a Dynamic Table" on page 993.

     The latter two options allow you to click the **Left**, **Center** or **Right** buttons in this dialog to align the cell's content.

A **row** is associated with a **(nested) detail table**. The row will be repeated in the output (or in Preview mode) as many times as necessary to display all the records in the associated detail table.

- To add a row, select a detail table from the drop-down next to the **Add Row** button at the top of the table. Or click the **Add Row** button itself to add an empty row.

  Note: Multiple rows may be associated with the same detail table. Consecutive rows linked to the same detail table are treated as a **group** and repeated as such.

- You can drag rows up or down to change the order in which they appear in the table.
- By default, a row is repeated once for each record in the detail table, but in tables with several levels and long nested tables you may want an *extra* repetition of a row if its sub-table runs over multiple pages.

  In order to **repeat a row before and/or after a page break**:

a. Add an extra row.

b. Associate it with the detail table of which a row must be repeated in case its sub-table gets split over multiple pages.

c. Change the **Initial** setting to **Before a page break**, **After a page break**, or before and after a page break (**All**). This button changes the data-show-rowattribute of the row; see "A Dynamic Table's data- attributes" on page 988. Note that this button is only available if the row is followed by a nested table.

d. Add fields as you like.

Note that this extra row will only appear in the output in case there is a page break in its sub-table.

Tip: A text like "continued on next page" or "continued from previous page" can be added after the table has been created. See "Editing cells in a Dynamic Table" on page 984.

- Activate the **Header Row** button to display a row with the names of the current row's data fields before each row in the output.
  The header row is not visible in the Wizard, but is inserted in the **body** of the table (the <tbody> element) when the Wizard is closed.
  In the output it will be repeated as many times as the current row. It has the same data-repeat attribute (see "A Dynamic Table's data- attributes" on page 988).
  Note that this button is only available if the row is followed by a nested table.

> **Note:** The data field names of the row with the **most fields** will appear in the **header** (the <thead> element) of the Dyamic Table. If more rows have that same number of fields, the **last** one wins.
>
> The **Initial** and **Header Row** buttons are disabled on the row that provides the titles in the header.

4. Click **Next** and check **Calculate Subtotals** to enable the options for a (sub)total at the end, and (in Print sections) in the footer and/or header. The options are:
   - **Field**: Select the data field for which the subtotal must be calculated.

     > **Note:** The wizard only allows to add subtotals for fields in the detail table itself, not in any nested detail tables.
     > However, it is also possible to add subtotals after the table has been inserted into the template, even for fields in nested detail tables. See "Adding subtotals" on page 982.

   - **Show subtotal lines before page break**: Check this option to display the subtotal in the footer of the table on each page (including at the end of the table).
   - **Show subtotal lines after page break**: Check this option to display the subtotal of the previous page in the header of the table on the next page.

   If neither of the latter two options has been enabled, the subtotal for the selected field will be put in the footer of the table and displayed only at the end of the table.
5. Click **Next** and select the desired table style. The top left option actually applies no styling to the table. The style can be easily changed afterwards; see "Styling Dynamic Tables" on page 992.
6. Check the option **Allow Resizing** to make it possible to change the width of columns in Design mode via drag-and-drop.

7. Check the option **Hide when empty** to make the table invisible when there are no data to display in it.
8. Click **Finish** to add the table to the section. Now you can edit the contents, and customize it further by adding rows and subtotals, hiding empty rows, etc. See:
    - "Customizing a Dynamic Table" on the facing page
    - "Editing cells in a Dynamic Table" on page 984
    - "Styling Dynamic Tables" on page 992

Tip: Once a table has been added to a template, it isn't possible to open it in the Wizard again. However, when you add a new Dynamic Table, the Wizard will open with the last used settings.

Caution: Do not remove rows from or add rows to the **body** of the Dynamic Table. This will lead to unexpected behavior and may cause errors.

Expressions or placeholders?

A new Dynamic Table in a template made with OL Connect version 2022.2 or higher is by default filled with expressions. Expressions look like this: {{FieldName}}.

ProductID	Description	UnitPrice	QtyOrdered	ItemTotal
{{ProductID}}	{{Description}}	{{UnitPrice}}	{{QtyOrdered}}	{{currency ItemTotal}}

Expressions are replaced with data by the Handlebars library which is integrated in OL Connect. (See "Handlebars in OL Connect" on page 998.) Unlike expressions in sections, which are replaced with data from the current record, expressions in a Dynamic Table are replaced with data from one or more **detail tables**. (Nested tables are supported, so a single Dynamic Table can use multiple detail tables.)

In templates made with older versions of OL Connect, Dynamic Tables are by default filled with placeholders. Placeholders look like this: @UnitPrice@.

ProductID	Description	UnitPrice	QtyOrdered	ItemTotal
@ProductID@	@Description@	@UnitPrice@	@QtyOrdered@	@ItemTotal@

Whether a cell contains expressions or placeholders has an impact on how it can be edited. See: "Editing cells in a Dynamic Table" on page 984 and "Formatting values" on page 987.

Switching between expressions and placeholders

Whether expressions or placeholders are used depends on a setting in the section. Here's how to change that setting.

1. On the **Resources** pane, right-click the current section and click **Properties**.
2. Check the option **Evaluate Handlebars expressions** to get expressions in new Dynamic Tables and when you insert variable data in the template (see "Variable data in text: expressions" on page 999). Uncheck the option to get placeholders instead.

It is not possible to switch from placeholders to expressions on the fly in an existing Dynamic Table. To change how the cells are filled, you will have to adjust the properties of the section accordingly and then re-insert the Dynamic Table.

Turning a standard table into a Dynamic Table

A standard table can be turned into a Dynamic Table.

1. Select the table (click in the table, then click the table element in the breadcrumbs at the top) and check the **Dynamic Table** box in the **Attributes** pane.
2. Select a row in the table and look at the Attributes pane under **Table**. **Repeat** is set to "None". Change this by choosing a detail table from the drop-down.
3. Now you can drag fields from the relevant detail table to the row. You could also add subtotals as described further on.

HTML: table element and data- attributes

In HTML, a Dynamic Table is a normal <table> element with rows and cells (see "HTML element: table" on page 861). But apart from the native attributes of a table, rows and cells, it has some proprietary `data-` attributes which make it dynamic.

These `data-` attributes are set automatically on a new table that is made with the Dynamic Table Wizard that was introduced in version 2020.1.
They can be changed via the "Attributes pane" on page 1241, depending on which element is selected (see "Selecting an element" on page 766), or directly in the template's HTML via the "Source tab" on page 1263.

> **Tip:** Just like other attributes, `data-` attributes can be used as selector; see "Using scripts in Dynamic Tables" on page 1095.

For a list of all data- attributes that can be used in a Dynamic Table, see "A Dynamic Table's data- attributes" on page 988.

Customizing a Dynamic Table

Once a Dynamic Table been added to a template, it isn't possible to open it in the Wizard again. But you can still add header and footer rows that appear at the start or end of the table or at the top or bottom of every page. You can also add subtotals, and you can hide the table or a table row when it's empty.

Tip: When you add a new Dynamic Table, the Wizard will open with the last used settings.

Adding a header or footer row

A Dynamic Table that is created by the Wizard always has a header (a <thead> element). It will also have a footer (a <tfoot> element) if you chose to add Subtotals when you created the table.

Sometimes you may want to add more rows to the header or footer of a Dynamic Table, for example, to add taxes and/or the total of an invoice, or to add a custom message like 'Continued on next page'.

A header or footer row can be added to a Dynamic Table as follows.

1. In the workspace, open the **Design** tab.
2. If the table already has a header/footer:

a. Right-click the header of the table if you want to add a header row, or the footer of the table if you want to add a footer row.

b. On the shortcut menu select **Row > Insert below** or **Insert above**. The new row will be added to the header or footer.

   If the table doesn't have a header/footer, add it (see "Adding a header or footer" on page 863) and proceed with the next step.

   Tip: To determine if a table has a header and/or footer, expand the table on the "Outline pane" on page 1250.

3. Select the new row (see "Selecting an element" on page 766).
4. On the "Attributes pane" on page 1241, select one of the options from the **Show Row** drop-down that specifies whether the row must be repeated in the output. The options are:
   - **At start of table**: The row will only appear on the page where the table starts.
   - **All**: The row will appear on all pages.
   - **Before page break**: The row will appear on all pages except the last.
   - **After page break**: The row will appear on all pages except the first.
   - **At end of table**: The row will only appear on the last page.

You can fill the new header or footer rows as usual. You could for example drag a root level data field to the row (see "Variable data in the text" on page 924) or type in the cells.

You could add a subtotal (see "Adding subtotals" on the facing page).

You could also add a script, or a script and text (see "Mixing text and data in a cell with a direct link to a data field" on page 986).

Adding subtotals

When creating a Dynamic Table with the Wizard, you can add subtotals based on **one** data field to the table's footer and header. It is also possible to add (extra) subtotals to an existing Dynamic Table.

Subtotals are added by using the **data-sum** and **data-show-row** attributes, normally in the header and/or footer of the table.

Note that this requires the table to have the data-expander="2019" attribute. (This is visible in the HTML. Switch to Source view and look at the <table> element.) Tables created with the Dynamic Table Wizard that was introduced in version 2020.1 will get this attribute automatically. The same goes for a standard table that has been turned into a Dynamic Table.

For information about a Dynamic Table's attributes see "A Dynamic Table's data-attributes" on page 988.

To add subtotals to the header and/or footer:

1. In the workspace, open the **Design** tab.
2. If the table doesn't have a header/footer, add it (see "Adding a header or footer" on page 863). To determine if a table has a header and/or footer, expand the table on the "Outline pane" on page 1250.
3. If you need an extra row in the header or footer:

a. Right-click the header or the footer of the table, depending on where you want to add a row.

b. On the shortcut menu, select **Row > Insert below** or **Insert above**.

4. Specify on which pages the row must appear in the output:

a. Select the row to which you want to add a subtotal (see "Selecting an element" on page 766).

b. On the "Attributes pane" on page 1241, select one of the options from the **Show Row** drop-down:

- **All**: The row will appear on all pages.
- **Before page break** (footer rows only): The row will appear on all pages except the last.
- **After page break** (header rows only): The row will appear on all pages except the first.
- **At end of table**: The row will only appear on the last page.

5. Now, add the subtotal to a cell in that row. This procedure is the same for subtotals in the header and footer.

a. Select the cell (see "Selecting an element" on page 766).

b. Select the data field:

- On the "Attributes pane" on page 1241, from the **Sum** drop-down, select the data field on which the subtotal must be based. The drop-down will offer all eligible fields, including those in nested detail tables.
- Alternatively, e.g. to select a data field in a *nested* detail table, switch to **Source** view and manually add the data-sum attribute to the table cell (the <td> element). Its value should consist of the path to the data field. For example: <td data-sum="Instruments.Items.Amount">. This cell will have a subtotal based on the "Amount" field in the "Items" detail table which is nested in the "Instruments" detail table.

c. You could also add text like "continued on next page" or "continued from previous page" to the cell. See "Editing cells in a Dynamic Table" on the facing page.

Subtotals inside a Dynamic Table

Subtotals normally appear in the header or footer of a table; however, it is also possible to insert subtotals in cells in the body of a table.

1. Start by adding a row to the body of the table: right-click a row in the table, depending on where you want to add a row, and select **Row > Insert below** or **Insert above**.

If the row should occur only once, you can now proceed with step 5 in "Adding a header or footer row" on page 981.

If the row should occur as often as a certain detail table, it must be associated with that detail table first.

1. Select the row (see "Selecting an element" on page 766).
2. Change the row's **Repeat** attribute on the **Attribute** pane from None to the desired detail table. Usually this will be the detail table that is one level higher than the one that contains the field for which the subtotal must be calculated.
3. Proceed with step 5 in "Adding a header or footer row" on page 981. The **Sum** drop-down will now only show the names of fields that occur in the specified detail table and in its nested detail tables.

Hiding an empty Dynamic Table

The number of rows in a Dynamic Table is variable, as it depends on a detail table in the data.
You can hide a **table** that does not contain any data. There are two ways to achieve that.

- When creating a Dynamic Table, you can check the option **Hide when empty**. (See "Creating a Dynamic Table" on page 974.)
- For an existing Dynamic Table you can select the table (see "Selecting an element" on page 766) and check the option **Hide when empty** on the "Attributes pane" on page 1241.

Hiding an empty table row

Empty rows are omitted from a Dynamic Table by default.
A row is considered empty if all elements with a data-field attribute in that row refer to empty data fields. Any additional text is not taken into account.

To keep empty rows visible in the output:

- Switch to **Source** view.
- Add the `data-keep-when-empty` attribute to the respective <tr> element(s).

  Example: `<tr data-repeat="Instruments" data-keep-when-empty="">`.

  Tip: To locate a row you can select it on the "Outline pane" on page 1250.

Editing cells in a Dynamic Table

How cells in a Dynamic Table can be edited depends on whether they contain expressions or placeholders. Not sure what the difference is between expressions and placeholders? The difference is explained in another topic: "Expressions or placeholders?" on page 979

Note that even in a dynamic table where most of the cells contain expressions, cells can have a direct link to a data field. A cell that shows a subtotal always has a direct link to a data field.

Cells with expressions

Table cells with **expressions** can be edited in the following ways.

- Additional text can be typed in a cell directly, before or after the expression. For example: *Price: {{currency ItemPrice}}*.
- You can add expressions to include data that is located somewhere else in the Data Model. Note that the expression will have to indicate where that data is relative to the table row (associated with a detail record) that contains the expression. E.g. use **../** to navigate one level **up**. For example: *{{../year}}*. (See "Accessing data at different levels" on page 1006.) When you drag and drop or double-click a field to add it to the table, the correct expression is added automatically.
- There is a number of "Format Helpers" on page 1015 that you can use in an expression to format the output differently. For example: *{{dateLong DueDate}}*.
- Other types of Helpers can also be used to modify cells in a Dynamic Table. See "Logic Helpers" on page 1011 and "Block Helpers" on page 1012.
- To change the content or style of a cell or span *based on the value of a data field*, you could write a custom Helper (see "Creating custom Helpers" on page 1018) or a script ("Writing your own scripts" on page 1064).

How to target the first or last row

To target the first or last row, you can use the `@first` and `@last` data variables in a Block Helper. A few examples.

Example: {{#if @first}}Policy conditions{{/if}}

In the above example, the {{if}} Block Helper only outputs the text 'Policy conditions' if the cell is in the first row.

Example: {{ItemDesc}}

Here, a Block Helper adds a class to a Span element only if it is in the first or last row. The class can then be used as selector for a CSS style or a script.

Getting the index of a row

In a Block Helper, the `@index` data variable can be used to work with the index number of the row. Note that the index is zero-based.

> **Example:** {{add @index 1}}: {{ItemDesc}}

The above expression outputs the number of the row, followed by the value of the ItemDesc field. E.g. "4: Upgrade Starter to Web".

Since @index is zero-based, the `add` Helper is used to add 1 to it.

> **Tip:** `add` is one of the Logic Helpers that is available out of the box; see "Logic Helpers" on page 1011.

Cells with placeholders

Dynamic Table cells with a **placeholder** by default have a direct link to a data field.

To find out whether a cell or span has a direct link to a data field, select it and:

- Take a look on the **Attributes** pane. If under **Other**, the **Field** drop-down or **Sum** field point to a data field in the detail table, the element has a direct link to a data field .
- Switch to the **Source view**. Elements with a direct link to a data field have a `data-field` or `data-sum` attribute.

Cells with a direct link to a data field are edited as follows.

- Which data field is linked to a cell and how the value is formatted can be changed via the "Attributes pane" on page 1241.
- *Any* text in the cell - including the placeholder - will be replaced with the value of the data field the cell is linked to, when the template is merged with data. To learn how to **mix data and text** in these cells, see "Mixing text and data in a cell with a direct link to a data field" below.

Changing content based on a value

Changing the contents of a cell or row in a Dynamic Table, based on the value of a data field, requires a script. See "Using scripts in Dynamic Tables" on page 1095.

Mixing text and data in a cell with a direct link to a data field

By default, each Dynamic Table cell that has a direct link to a data field shows a **placeholder**. For example: @CompanyName@ refers to a data field named CompanyName.

However, these placeholders are **not used as selector**.

Instead, if a cell has the `data-field` or `data-sum` attribute, the software will replace the **entire contents** of that cell with a value: the current value of the data field, or a running total, respectively.

(For information about data- attributes see: "A Dynamic Table's data- attributes" on page 988).

Any text that was added in Design mode will disappear.

In order to mix data and text in a cell with a data- attribute, you should remove the `data-field` or `data-sum` attribute, respectively, from the cell itself and then put that attribute on an element inside the cell.
Here's how to do that.

1. In the Workspace, open the **Design** tab.
2. Select the cell (see "Selecting an element" on page 766).
3. On the "Attributes pane" on page 1241, set **Field** or **Sum** to **None**.
4. Type text in the cell, before and/or after the placeholder.
5. Select the placeholder, right-click it and select **Wrap in Span...**
6. Select the new Span element.
7. On the "Attributes pane" on page 1241, set **Field** or **Sum** to the desired data field.
8. Reapply the format if needed (see "Formatting values" below).

Formatting values

Values in Dynamic Table cells can be formatted in different ways. A date value, for example, could be displayed as "Apr 2, 2016" (short date) or "4/2/16 12:00 AM" (medium date/time).

Which formats are possible depends on the data type of the field that contains the value (see "Data types" on page 436).

Selecting a format

The format of each value can be chosen when creating a table. For an explanation of the possible formats see "Formatting variable data" on page 952.

How you change the formatting of values in an existing table depends on whether the cell or span contains an expression (e.g. {{fieldname}}) or is directly linked to a data field (see "Expressions or placeholders?" on page 979).

If the cell or span contains an expression:

- There is a number of "Format Helpers" on page 1015 that you can type in an expression to format the output differently.

If the cell or span has a direct link to a data field:

- Select the element that contains the value and then use the **Format** drop-down on the "Attributes pane" on page 1241.
- Open the "Source tab" on page 1263 and manually add the desired format to the cell using the `data-format` attribute (see "A Dynamic Table's data- attributes" on the facing page).

A Dynamic Table's data- attributes

In HTML, a Dynamic Table is a normal <table> element with rows and cells (see "HTML element: table" on page 861). But apart from the native attributes of a table, rows and cells, it has some proprietary `data-` attributes which make it dynamic.

These `data-` attributes are set automatically on a new table that is made with the Dynamic Table Wizard that was introduced in version 2020.1.
They can be changed via the "Attributes pane" on page 1241, depending on which element is selected (see "Selecting an element" on page 766), or directly in the template's HTML via the "Source tab" on page 1263.

Tip: Just like other attributes, `data-` attributes can be used as selector; see "Using scripts in Dynamic Tables" on page 1095.

Note: When a Dynamic Table is filled with *expressions* instead of placeholders, no `data-` attributes are added to *cells*. See "Expressions or placeholders?" on page 979

The table below lists the availabe `data-` attributes.

Tip: A number of these attributes can be set in Design mode on the "Attributes pane" on page 1241 after selecting a table, row or cell (see "Selecting an element" on page 766).

`data-column-resize`	<table>	Indicates that the columns may be resized in Design mode via drag-and-drop. This attribute has no value.
`data-detail`	<table>	This attribute identifies a table as a Dynamic Table. It has no value. In tables made with an OL Connect version older than 2020.1, this attribute's value is the name of a detail table in the data.

`data-expander="2019"`	<table>	Specifies that the table should be expanded using the expander that was introduced in OL Connect version 2020.1. Most of the `data-` attributes listed here require the Dynamic Table to have the `data-expander="2019"` attribute. Tables made with an OL Connect version older than 2020.1 don't have this attribute. Tables without this attribute are expanded using the previous expander, which doesn't support all of the attributes listed here. (See "Differences in Dynamic Tables made with previous Connect versions" on page 991).
`data-field`	Table cells and other HTML elements within repeating rows: <td>, , etc.	This attribute associates a **data field** with the element, e.g. `data-field="InstrumentClass"`. The data field's current value will automatically replace the contents of the element when generating output (or previewing). By default this attribute is placed on cells. In order to mix data and text in a cell, move the `data-field` attribute from the cell itself to an element inside it. See:"Mixing text and data in a cell with a direct link to a data field" on page 986. **Note:** This attribute is not used when a Dynamic Table is filled with expressions. See "Expressions or placeholders?" on page 979

`data-format`	<td> (a table cell)	Defines how a value must be formatted; e.g. `data-format="date-short"` displays a date value using the format for a short date. See: "Formatting values" on page 987.
		
		The available formats depend on the data type of the corresponding field in the Data Model (see "Data types" on page 436).
		They correspond to the options given in the Text Script Wizard. For a detailed description see: "Formatting variable data" on page 952.
		
		• Integer, Float, Currency:
		  ◦ grouped
		  ◦ number-ungrouped
		  ◦ currency
		  ◦ currency-nosymbol
		  ◦ pattern('*some pattern*') (see "Number patterns" on page 1511)
		• String:
		  ◦ uppercase
		  ◦ lowercase
		  ◦ propercase.
		
		Note that it is not possible to set a format on an HTMLString value.
		• Date:
		  ◦ date-short
		  ◦ date-medium
		  ◦ date-long
		  ◦ time-short
		  ◦ time-medium
		  ◦ time-long
		  ◦ datetime-short
		  ◦ datetime-medium
		  ◦ datetime-long
		  ◦ pattern('*some pattern*') (see "Date and time patterns" on page 1491).
		
		Note: This attribute is not used when a Dynamic Table is filled with expressions. See "Expressions or placeholders?" on page 979

`data-hide-when-empty`	<table>	Hide the table if it is empty. A table is considered empty if the detail record is empty. Note that any additional text is not taken into account. See: "Hiding an empty Dynamic Table" on page 984.
`data-keep-when-empty`	<tr> (a table row)	Hide the row if it is empty. A row is considered empty if all elements with a `data-field` attribute in that row refer to empty data fields. Note that any additional text is not taken into account. See: "Hiding an empty table row" on page 984.

`data-repeat="name of detail table"`	<tr> (a table row) in the body of the table (<tbody>) Note that this attribute is not supported for rows in the header or footer of a table (<thead>, <tfoot>).	A row with this attribute is repeated for each record in the detail table that is specified in the attribute's value. Example: `data-repeat="products"`. The name of a nested detail table should include all its parent tables, starting at the root and separated with dots. Example: `data-repeat="InstrumentClass.Sector.Holding"`. **Note:** Empty records/data fields are skipped by default. This attribute cannot be used as a selector for template scripts, since it gets removed when the table is expanded, which happens before template scripts run.
`data-script`	<tr> (a table row), <td> (a table cell),	When creating a script for a row, cell or span in a Dynamic Table this attribute is set on the element and used as selector (see "Quick-start a script with the Create script button" on page 1096). Its value is the name of a data field; e.g. `data-script="ID"`. **Note:** This attribute is not used when a Dynamic Table is filled with expressions. See "Expressions or placeholders?" on page 979
`data-show-row`	<tr> (a table row)	When set on a row in the **body** of the table (<tbody>), this attribute causes a row (a line with data) to appear before and/or after a page break if a sub-table runs over multiple pages. This is particularly useful in tables with several levels and long nested tables. Initially this attribute is omitted. A row without this attribute will be displayed once per record in a detail table. This attribute can be set to the following values: • `all`: Show the row before and after a page break. • `after-break`: Show the row after a page break. • `before-break`: Show the row before a page break. • `start-of-table`: Only show the row at the start of the table. On a row in the **header** (<thead>) or **footer** (<tfoot>) of the table, this attribute can also be set to the following value: • `end-of-table`: Only show this row on the last page.
`data-sum`	<td> (a table cell) in the header or footer of a table	Adds a running total (i.e. a subtotal) to a table cell after pagination. Its value consists of the path to a numeric field in a detail table, for example: `data-sum="Items.Amount"` will calculate a subtotal based on the "Amount" field in the "Items" detail table. See also: "Adding subtotals" on page 982.
`data-breakable`	<tr> (a table row)	This attribute is added by the software to every **copied** row (in Preview mode or in output), in each of them with a unique ID as its value. This is required by the pagination routines of Connect to split the table across pages.

Differences in Dynamic Tables made with previous Connect versions

In Dynamic Tables made with Connect before version 2020.1:

- There is no `data-expander` attribute on the <table> element.
- The `data-detail` attribute on the <table> element specifies the name of the detail table. (In tables made with Connect 2020.1 and up, this attribute has no value.)
- The `data-show-row` attribute can only have the value: `all`.
- The `data-field`, `data-format`, `data-keep-when-empty`, `data-script` and `data-sum` attributes cannot be used.
- The `data-showin` attribute determines the visibility of a header row or footer row if the table gets split over multiple pages. Its possible values are `header`, `footer` and `break`; `break` may be combined with `footer` or `header`, for example: `data-showin="footer, break"`, to make the row show up on every page.

Styling Dynamic Tables

The Insert Dynamic Table wizard lets you select a style for the table.

Tables are styled via CSS: when the wizard adds a Table, the chosen theme's **class** is added to the <table> element, and, if it doesn't exist yet, the **default_table_styles.css** file is added to the resources of the template. This CSS file contains the CSS rules for all table themes. (See: "Styling templates with CSS files" on page 881.)

To change the theme, you could simply select the table and change its class on the "Attributes pane" on page 1241.
The available theme classes are: table--grid, table--colgrid, table--minimalist, table--dark, table--light, table--striped, table--topbar, table--portfolio. (Note the double dash in the class name.)

The default_table_styles.css file is read-only, but of course you could overwrite styles and create your own theme, using your own **style sheets**.
Regardless, you can change the font, font size and color, the borders, the cell padding (the distance between the edge of the cell and its content), and the background color or image of the table and its cells, via the **Formatting** dialog.

Both approaches are explained in the topic: "Styling a table" on page 902.

Dynamic Tables are best styled via CSS (see "Via a style sheet" on page 903), for two reasons:

- With local formatting, all rows that are added on the fly (in Preview mode and in output) will look exactly the same as the first one. Alternating row colors, for example, in dynamically added rows can only be done via CSS.
- When generating output from a template, a Dynamic Table is created slightly faster when it's styled via Cascading Style Sheets than when it's styled with local formatting.

Here are a few tips on how to target rows and cells in a Dynamic Table.

- There are **CSS selectors** with which you can target every so-maniest row (see "Styling the first, last and nth rows" on page 904).
- In the output, Dynamic Table rows are repeated including any **classes** that are set on the row and on its contents.
- Dynamic Tables and their rows and cells have some special **attributes** that can be used as selector. See: "A Dynamic Table's data- attributes" on page 988.

Tip: Styling Dynamic Tables with scripts is demonstrated in this howto: How to script the appearance of data in a Dynamic Table

Utility classes for text in a Dynamic Table

In order to style text in a Dynamic Table, you could use some classes for which there are style rules in the default style sheet for Dynamic Tables (default_table_styles.css). The classes are:

- .u-text-left
- .u-text-center
- .u-text-right
- .u-text-bold
- .u-text-normal
- .u-text-italic

These classes will be applied when you choose **Use CSS utility classes** as Text Alignment mode when creating a table. To change the alignment after the table has been created, select the cell (see "Selecting an element" on page 766) and then type the desired class in the "Attributes pane" on page 1241.

Note that if you selected **Via inline styles** as Text Alignment mode when creating the table, the utility classes won't work, unless you remove the inline style first. Inline styles always get applied since they are more specific. (For an explanation see "Using a more specific CSS rule" on page 890.)

Changing styles based on a data field value

Changing the styles of a row or cell in a Dynamic Table, based on the value of a data field, requires a script. See "Using scripts in Dynamic Tables" on page 1095.

Resizing a Dynamic Table

To change the **width** of a Dynamic Table or of a column in a Dynamic Table, select it (see "Selecting an element" on page 766) and type the desired width as a percentage in the respective field on the "Attributes pane" on page 1241.
It is also possible to resize colums via drag-and-drop (in Design mode only). To allow for this, select the table and check the option **Allow Resizing** on the Attributes pane.

The **height** of the Dynamic Table is adjusted automatically to the amount of data added to it in Preview mode or when generating output.
It is however possible to change the height of the rows: click in the row and type the desired height in the respective field on the **Attributes** pane. All line item rows will have the same height.

Adding leading and trailing space to a Dynamic Table

The best way to add extra space at the top or bottom of a Dynamic Table is to set the Table's top or bottom margin, respectively (see "Styling a table" on page 902).
Do not use empty lines (<br>) or paragraphs (<p>) to add trailing space to a table. In Print output they might end up on a next page without any other visible content, which would still be printed because lines and paragraphs must be treated as content, even when they are empty.

Dynamic Print section backgrounds

Print sections can have a PDF background image (see "Using a PDF file or other image as background" on page 634). This feature is often used to create a Print template from a PDF file while making some minimal changes to it. Of course, the PDF file itself can't be edited in a Designer template, but when it is used as a section's background, text and other elements, such as a barcode, can be laid on top of it.

Dynamic backgrounds are switched automatically depending on the value of a data field.

Adding a dynamic background

Dynamic background PDFs can be added to a Print section using the Dynamic Section Background Script Wizard **if**:

- One or more data fields contain values on the basis of which the PDFs can be switched.
- There is an appropriate PDF for each case. It is important that the PDFs are named after the various possible values of the related data field.

- All PDFs are stored in one folder (the **Images** folder on the **Resources** pane, or an external folder).

Otherwise, adding a dynamic background requires a self-made script (for help on this, see "Control Script: Setting a Print section's background" on page 1106).

To use the Dynamic Section Background Script Wizard:

1. Open the Wizard. There are three ways to do that:
   - Right-click the Print section on the **Resources** pane and click **Dynamic Background**.
   - On the **Scripts** pane, click the arrow next to the **New** button and select **Dynamic Background Script**. Double-click the script to open it.
   - On the **Scripts** pane, right-click **Control** and select **New > Dynamic Background Script**.
2. Select the desired section from the **Section** drop-down. You may also change the name of the script.
3. Select the location of the files:
   - Select **Resources** if the PDFs reside in the **Images** folder on the **Resources** pane.
   - **Folder on disk** refers to a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder.
     As an alternative it is possible to enter the path manually. You can give a local path (e.g. C:\Images\) or use the "file" protocol.

     The complete syntax of a fully qualified URL with the "file" protocol is: `file://<host>/<path>`. Note: if the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: `file:///c:/resources/images/`.
   - **Online** requires you to give a specific web address (for example, http://www.mysite.com/images/).
4. The Dynamic Section Background Script Wizard composes the file names (including the path) based on the selected **location**, the **prefix**, the value of a **data field**, and the **suffix**. The variable part of the file names is the value of the data field(s) in the **Field** column. The prefix and suffix are meant to contain static parts of the file names.
   Click the first field in the column **Field**, and then click the downward pointing arrow. Select the data field to be evaluated.
   If you want the file name to be composed of the value of several data fields, simply click in the next row or click the **Add** button. This adds a row. Note that the rows will

be concatenated to compose one file name. Only the last suffix should contain the file extension (.pdf).

5. Click **Apply** or **OK**.
   The script assigns the resulting file name, including the path and file extension, to the URL of the section's background.
6. Now click the **Preview** tab and browse through the records to verify that the script works as expected.

If you want to see the underlying script, reopen the script and click the **Expand** button. For more information on this type of script, see "Control Script: Setting a Print section's background" on page 1106).

Note: Any non-content features in the original PDF will get stripped out of the original PDF. These include document navigation (outline), local links, hyperlinks, annotations (markup annotations, comments, etc.), digital signatures, multimedia features, interactive forms, and tagged PDF features (marks in the page content will stay, but the logical structure is gone).

Personalized URL

Personalized URLs (pURLs) are links that are tailor-made for a specific purpose, generally for individual clients. They can serve multiple purposes, for instance:

- **Click Tracking**: A unique ID in the link makes it possible to track the source of the click (for example, a link in an email campaign).
- **User Tracking**: A user-specific ID reveals who clicked the link and at what time.
- **Landing Pages**: Information in the link invokes a unique landing page with specific products or services.
- **Personalized User Pages**: Using information from a database, a user is served a completely personalized web page with their name and information tailored to them, enhancing user response.

Typically, a pURL takes the recipient to a personalized landing page, for example to download an invoice or get access to specific products or services.

Prerequisites

Creating a personalized URL in a Connect template requires some planning, because you need to have a few things at hand when adding it to a template.

Tip: If the link must be dynamic but OL Connect does not need to create personalized content, you could simply use a Handlebars expression in the url of the link. See "Hyperlink and mailto link" on page 850.

First of all, you need a **Workflow process** for the pURL to point to; in response, this process will create personalized content.
In order to invoke that process, the host in the pURL must be followed by the HTTP action of the HTTP Server Input task that starts the process. For example, if its action is `MakePDF`, you could trigger the process by adding `/MakePDF` to the address of the Workflow server.
The rest of this Workflow process depends on what the pURL is used for. If it should output a personalized landing page, for example, it would be a web process with some OL Connect tasks (see "Web processes with OL Connect tasks" on page 309).

Secondly, the pURL must contain the **data** that Workflow needs in order to create the personalized response (be it a web page or other file). For instance, creating a personalized page that shows a client's invoice may require the Invoice Number to be present in the pURL, which is then used by Workflow to retrieve the invoice data, generate the invoice in PDF or HTML format using a template, and then return it to the browser.
The data needs to be included in the URL. In this URL, for example: `http://www.example.com/MakePDF?uuid=8c33eda8-72af-11e5-8bcf-feff819cdc9f`, a unique ID is passed as parameter. Parameters end up as key-value pairs in the request XML file. In the Workflow process you may use a Set Job Infos and Variables task to assign a value from that XML to a variable (see Data Selections, in Workflow's Online Help).

Finally, you need another **Workflow process** to provide the pURL to the user, in an email, for example, or in a QR code. This process must add the necessary data to the pURL. That data should be available through the Data Model used in the template.

Tip: In Workflow, you may make use of the system variable **%U** to create unique strings.

Inserting a pURL

Assuming that the necessary data and Workflow processes are available, here's how you insert a personalized URL in a template.

1. Open the template and find or enter the text that you want to turn into a link.
2. Select the text, right-click it and select **Wrap in Span...**.
3. Give the span an ID, for example: `get_pdf_link`. Click OK.
   The text `Download PDF`, wrapped it in a span with the ID `get_pdf_link`, would look

like this on the Source tab: `<span id="get_pdf_link">Download Invoice</span>`.

4. With the span selected, click on 'ID' on the Attributes pane (click on the field name, not the field itself). This will insert a script that has this ID as selector, and open it in the script editor.
   Since it's dynamic, inserting a personalized URL always implies writing a script, however small and simple. For a basic explanation of scripts, see "Writing your own scripts" on page 1064.
5. Type the code that wraps the span in a <a> element (a hyperlink). In the link, the host should be followed by the action with which the corresponding Workflow process is triggered and the data with which Workflow will be able to create a personalized response.

Sample script

Let's assume:

- A Workflow process is triggered by the HTTP action: `MakePDF`.
- The template that outputs the pURL is to be merged with a record that contains a data field `invoice_UUID`. This data field holds the information that the 'MakePDF' Workflow process needs in order to create personalized output.
- The template contains a span with the link text "Download Invoice": `<span id="get_pdf_link">Download Invoice</span>`.

The following script, with the selector `#get_pdf_link`, will create a valid link around the span:

```
results.before('<a href="http://www.example.com/MakePDF?uuid=' +
record.fields.invoice_UUID + '">');
results.after('</a>');
```

Tip: More ways to insert a pURL in a template are described in a how-to: Variables in hyperlinks.

Handlebars in OL Connect

Handlebars is the name of a JavaScript library that implements a templating language (see https://handlebarsjs.com/). It uses a template and an input object (i.e. data) to generate output.

The template looks like regular text with embedded Handlebars expressions. A handlebars expression is some contents enclosed by double curly braces: **{{** ... **}}**.

Example: `<p>Hello {{firstname}}!</p>`

When the template is rendered, these expressions are replaced with values from an input object.

The Handlebars library is **integrated** in OL Connect Designer. This means that:

- Handlebars expressions can be used in all **sections** as of version 2022.2.

  Note: In templates made with OL Connect 2022.1 or older, Handlebars expressions are not evaluated by default. To enable expressions in a section, right-click the section in the Resources pane, select **Properties** and check the option **Evaluate Handlebars expressions**.

- Handlebars expressions can be used in a special type of **snippets** called "Handlebars templates" or "partials". These were introduced in version 2022.1. Handlebars templates can in turn use other Handlebars templates as snippets. See "Handlebars templates" on page 1023.

Expressions are inserted in sections and Handlebars templates the same way; see "Variable data in text: expressions" below.

More about expressions and the functions that you can use in expressions can be found in the topics "Handlebars expressions" on page 1005 and "Using functions in expressions: Helpers" on page 1009.

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Variable data in text: expressions

As of OL Connect 2022.2, variable data can be inserted in a template via **expressions**. Expressions look like this: *{{FieldName}}*. When the template is merged with the data, the expressions get replaced with values.

In previous versions, variable data used to be inserted in a template via **scripts and placeholders** (see "Variable data in text: scripts and placeholders" on page 945). The advantage of expressions is that fewer scripts and sometimes no scripts at all are needed in a template.

Powered by Handlebars

Expressions get replaced with data from the current record by the Handlebars library which

is integrated in OL Connect. For more information about Handlebars, see: "Handlebars in OL Connect" on page 998.

Note: Before you can add variable data fields to the contents of your template you need to load - or create - a Data Model (see "Data Model pane" on page 1244), a data mapping configuration, or data from a data file or database (see "Loading data" on page 926).

Note: Web templates are personalized just like any other template. There are a few extra possibilities, though; see "Using variable data on a Web page" on page 690.

Note: Make sure to include the curly brackets when selecting an expression if you want to style the results. HTML tags inside the curly brackets break the expression.

Writing expressions

To insert an expression that gets replaced with the value of a field in your data, you can simply **write** the field name between double curly braces: *{{FieldName}}*.

- A whitespace before and after the field name is accepted, e.g. *{{ FirstName }}*.
- If the field name contains a space you will also need to enclose it in square brackets: *{{[Field Name]}}*.

This works in Design view and in Source view. You can even use expressions inside HTML tags and attributes.

Tip: An easy way to add a dynamic image or hyperlink is to use an expression in the image's `src` attribute or in the hyperlink's `url` attribute.

In expressions, you can use functions called *Helpers*, such as "Format Helpers" on page 1015 and Block Helpers (see "Using functions in expressions: Helpers" on page 1009).

Tip: If you miss a function, or if an expression becomes very long, consider writing your own Helper; see "Creating custom Helpers" on page 1018.

Inserting expressions via drag-and-drop / double-click

The **drag-and-drop** or **double-click** method is best for data that do not need to be preceded or followed by a space, line break or text. Otherwise it is better to use the wizard (see "Using the Helper Wizard" on page 1003).

1. Open the section or Master Page you want to add the data field to.
2. Either:
    - **Drag and drop** a data field from the **Data Model** pane at the bottom right into the content of your template.
      To select and insert **multiple** data fields at the same time, you could press **Shift** or **Ctrl**, whilst selecting fields in the Data Model pane, but it would be better to use the wizard (see "Using the Helper Wizard" on page 1003).
    - Place the cursor in the template where you want the data to be inserted and then **double-click** the data field in the Data Model pane.
    - Or, keep the **Ctrl** key pressed down while dragging to insert a "Positioned Box" on page 824 with the expression in it.

The first two methods work in Design view and in Source view, the latter only works in Design view.

Tip: Press the **Ctrl** key *while dragging* to wrap the expression(s) in an absolute positioned box (a **div**) at the cursor position.
Absolute positioned boxes are particularly useful when the document mainly consists of a PDF used as the background image of a section (see "Using a PDF file or other image as background" on page 634).

What happens is that:

- The name of the data field appears in the template, enclosed in double curly braces: **{{data field name}}**. This is called an **expression**.

  Example: Hello {{FirstName}}!

  Note: If the data field is of the type **HTML String**, the expression gets three curly brackets to ensure that the output is interpreted as HTML. In expressions with double curly braces, characters that have a special meaning in HTML are *escaped*, e.g. "&" is written out as "&". (See also: "HTML values" on page 1008.)

  Note: Does a **placeholder** appear instead of an expression? Placeholders look like this: *@FieldName@*. This happens by default when a template was made with an OL Connect version prior to 2022.2.

To make the software insert expressions instead of placeholders, right-click the section in the **Resources** pane, select **Properties** and check the option **Evaluate Handlebars expressions**.

See "Variable data in text: scripts and placeholders" on page 945 for an explanation of how placeholders work.

- Depending on the data type of the dragged field, a function that formats the value may be added to the expression automatically. For example, dragging a currency field will produce: *{{currency FieldName}}* which outputs a number formatted as an amount of money.

  Example: The amount due is {{currency Total}}.

  You can remove the function or change it (see "Format Helpers" on page 1015).

To see the expression replaced with the value of the data field in the current record, switch to the **Preview** mode by clicking the Preview tab at the bottom of the workspace. The value will be refreshed when you browse through the records in the Data Model pane.
The same happens when the output is generated. With each record, expressions that hold the name of a data field get replaced with the value of that data field.

Note: Whether expressions in a Master Page are evaluated when output is generated, depends on the Print section that the Master Page is used with. See "Print section properties" on page 1200.

Detail data in expressions

When you drag and drop detail data in a template, the expression includes the path to the data field. For example, suppose there is a detail table called Products that has a Description data field. Dragging and dropping the Description field from the first record of the detail table inserts the following expression:
{{Products[0].Description}}
The **[0]** refers to the first record. Remember that detail tables can have a variable number of records.

Note: Detail data can only be dragged into a row in a Dynamic Table that is based on the same (nested) detail table (see "Dynamic Table" on page 974).

Note: Another way to insert detail data is to use a Block Helper, preferably in a Handlebars template (see "Block Helpers" on page 1012 and "Handlebars templates" on page 1023).

Tip: Double-click the name of a detail table, or drag and drop it from the Data Model into the template, to start the Dynamic Table wizard. See "Dynamic Table" on page 974.

Using the Helper Wizard

Should the value of a data field be preceded or followed by a space, line break or text in the template, like in an **address block**? Then it is best to use the Helper Wizard. Especially if there are any **empty** fields in the data. Otherwise, empty data fields will cause empty lines and superfluous white spaces to show up in the text.

The Helper Wizard is also a handy tool to use if a field name, and hence the expression, is too long to fit in a table cell, for example.

Tip: The following how-to shows how to create an address block using the Helper Wizard: Creating an address block. However, OL Connect also has a special wizard to help you insert an address block: "Address Block" on page 955.

The Helper Wizard creates a "Helper" - a function that can be called in an expression. When called, the Helper inserts one or more data fields into your template, each with an optional prefix and suffix.
The wizard also makes it easier to **format** data differently, for example, display a number as a currency.

1. Open the Handlebars Text Helper Wizard: on the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Handlebars Text Helper**.
2. Change the name of the Helper to make clear what it does. The name must be unique.

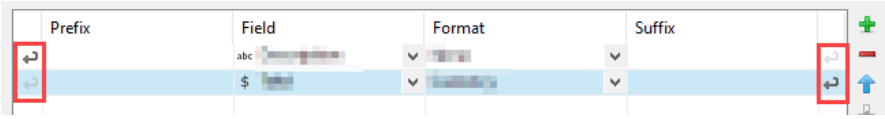
   The name of the Helper is what you use in an expression. For example, if its name is MyHelper, then the expression that calls the Helper is {{MyHelper}}.

   Tip: To make it easy to identify your own Helpers, you can start the name with a special character such as an underscore.

   Tip: Use a short name to make the expression fit in a small space.

3. Click the downward pointing arrow to select the **Scope**: the record or a detail table. The fields you choose in the Helper Wizard must all be on the root of the record, or in the same detail table.
4. Click the downward pointing arrow in the first row in the column **Field**. Select a data field from the list that appears.
5. Add a **Prefix** and/or a **Suffix**. The prefix and suffix can contain text and/or HTML tags, such as:
   - with a Number field, Prefix: *Your invoice* (one space at the end), Suffix: *is now ready to be viewed!*
   - with a field State, Prefix: , (comma then space).

   To add a single **line break**, activate the Line break button in either the first or last column.

   Alternatively, or to add an *extra* line break, you can add *
*, which is the HTML tag for a line break, to the Suffix. For example:
   - with a field LastName, Suffix: *
*

   Note: If a field is empty, the prefix and suffix will be ignored.

   Tip: For a comma between fields, use the Prefix of the second field if you don't want a comma when the second field has no value.
6. The Wizard allows you to format the data (for example, display a text in uppercase, apply thousands separators to numbers, etc.). Click the column **Format**, then click the downward pointing arrow and select one of the formats. For an explanation of the formats see "Formatting variable data" on page 952.
7. Add as many data fields as you need, following the same procedure.
8. Optionally, you can click **Options** at the bottom to specify how the result is inserted:
   - As **HTML**. HTML elements in the result are processed and displayed as HTML elements. For instance, `<b>this is bold</b>` will be displayed as **this is bold**. This is the default setting.

- As **text**. This inserts the result as-is, meaning HTML tags and elements are displayed as text in the output. In this scenario, "<br>" shows up in the text and does not insert a line break. This is the preferred setting if the Helper produces plain text. It will be slightly faster than HTML since it avoids processing with an HTML parser.

9. Close the Wizard. The new Helper appears on the Scripts pane under Control Scripts.
10. Now all you have to do is **drag the Helper** from the Scripts pane into the template. Alternatively you can write the name of the Helper enclosed in double curly brackets: e.g. *{{MyHelper}}*. Do this anywhere where you want the result to be inserted.

    Tip: When one of the included data fields is empty, the respective line, including the prefix and suffix, is skipped. The result will be shorter, causing the rest of the content to move up. If, in a Print context, you don't want the result of a Helper to be part of the text flow (for example, when a letter is going to be sent in an envelope with a window), put the expression that calls the Helper in a positioned box (see "Boxes" on page 823 and "How to position elements" on page 897).

Handlebars expressions

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Data in expressions

A Handlebars expression is some contents enclosed by double curly braces: **{{...}}**.
For example: *Hello {{firstname}}!*

In its most simple form the contents of the expression refer to a data field. They will be replaced with the value of that field if it exists in the data that is passed to the template.

- Expressions in **sections** are resolved using the **current record** with which a template is merged. In other words, the input object is the current record. (See "Variable data in text: expressions" on page 999.)
- Expressions in a **Dynamic Table** are resolved using the **detail record** associated with the closest table row. (Nested tables are supported, so a single dynamic table can use multiple detail tables.) See "Dynamic Table" on page 974.

- In the case of **Handlebars templates** (snippets), you may determine in a **script** with which data the snippet will be merged. See "Handlebars templates" on page 1023 for instructions.

Accessing data at different levels

A field or key name without a **.** or **/** is assumed to refer to a field or table at the root of the data with which the template is rendered. There are a number of ways to access data that is not located at the root.

- With **.** or **/** you can navigate **down** into the record or object. For example: {{Cars.Brand}} and {{Cars/Brand}} refer to the same field.
  Note that to access an item in a detail table or array directly, you also have to navigate down. For example: Cars.Brand.Models[0] refers to the first item in Models. Alternatively you can write: Cars.Brand.Models.0
- **@root** represents the top level of the data.
  If a template is rendered with the current record or part of it, @root refers to the root of the current record.
  With a JavaScript object, @root refers to the root level of the data that was passed to the Handlebars template.
- The built-in helpers #each and #with dive into an object-property, giving you direct access to its properties. (See "Using functions in expressions: Helpers" on page 1009 and the documentation of Handlebars.)
- Use **../** to navigate one level **up**. This is particularly useful in #each and #with blocks.

  Note: If a Handlebars template is rendered with part of the current record, the template still has access to the *entire* record and can navigate up and outside of that part, to a higher level in the current record.
  If a JavaScript object is passed, the template only has access to the passed data, and it does not have access to the current record.

Note: Navigating to a parent scope with something like **@../index** is **not** supported.

Template properties

A template's file properties are accessible in an expression via `@properties`.

Example: @properties.description

For a list of file properties see "File Properties dialog" on page 1150.

Custom file properties are accessed via `@properties.custom`.

Example: @properties.custom.Name1

Runtime parameters

A template's "Runtime parameters" on page 608 can be accessed in an expression via `@parameters`.

For example, if a template has a runtime parameter called `campaign`, you can use the following expression to insert the value of that runtime parameter in a section:

Example: {{@parameters.campaign}}

A section's `meta` properties

The **@meta** helper for Handlebars gives access to information stored in the section's `meta` property (see meta).

Use **.** to access fields in the `meta` property.

Example: {{@meta.title}}

Note: If a `meta` entry is added by a Standard Script, it cannot be accessed in a Handlebars expression, since Handlebars expressions are evaluated *before* Standard Scripts run.

Note: A section's `meta` property is not accessible to Handlebars expressions in a Handlebars template that is rendered using a Control Script (see "Inserting a Handlebars template using a script" on page 1033). Use a Standard Script instead.

Formatting values

Functions that can be used in a Handlebars expression are called "Helpers". OL Connect provides a number of Helpers to **format the value** that is the result of an expression. For example: {{upperCase FirstName }} returns the value of the FirstName data field in uppercase.

The available functions are listed in the topic: "Format Helpers" on page 1015.

Using logic in expressions

With Block Helpers it is possible to use **conditions** and create **loops** in an expression, for example to display content only if a field has a certain value. See "Block Helpers" on

page 1012.

OL Connect has a number of additional Helpers that allow for the use of logic in expressions. See "Logic Helpers" on page 1011.

HTML values

The value returned by an expression is HTML-escaped. This means that characters that have a special function in HTML: **& < > \" ' ` =** are replaced with HTML character references. For example: **&** will be changed into **&**.

Take this expression: {{foo}}.
If the field *foo* contains a '>' and you don't HTML-escape it, this would produce invalid HTML: *...>...*.
After HTML-escaping, it becomes valid HTML: *>*.

If the result of an expression should **not** be HTML-escaped, the expression must be enclosed in three curly brackets: **{{{** ... **}}}**. Characters with a special function in HTML will then be interpreted as HTML. For example, the value "Name" will be displayed as **Name**.

When an HTML field is dragged into a Handlebars template it is automatically enclosed in three curly brackets.

> **Note:** Content provided by 'block helpers' - expressions in the form {{#helperName arguments}}...{{/helperName}} - is **not** automatically HTML-escaped. See "Using functions in expressions: Helpers" on the next page.

Testing expressions

To test whether expressions give the expected result, simply switch to Preview mode in the Workspace. You could also do a Preflight to test with multiple records (see "Doing a Preflight" on page 1075).

The first thing to do if an expression doesn't seem to be evaluated in a section, is to check whether evaluating Handlebars expressions is enabled for that section.

> **Note:** In templates made with OL Connect 2022.1 or older, Handlebars expressions are not evaluated by default. To enable expressions in a section, right-click the section in the Resources pane, select **Properties** and check the option **Evaluate Handlebars expressions**.

Logging

Values can be logged with an expression that uses the log Helper. For examples see Handlebars' documentation.

Log messages will appear in the **Messages** pane (see "Preflight Results and Messages" on page 1248).

> **Note:** In OL Connect it isn't possible to set a log level for the Helper. The log level is always 'info'.

Escaping Handlebars expressions

Handlebars content may be escaped - i.e. displayed as is, without evaluating the expression - by prefixing it with **** (a backslash). For example: *\{{FieldName}}* will show up in the output as *{{FieldName}}*, instead of being replaced with the value of the data field FieldName.

Raw blocks - Handlebars expressions with four curly braces at the start and end - are not supported.

Using functions in expressions: Helpers

> **Note:** The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Functions that can be used in a Handlebars expression are called *Helpers*.

> **Example:** {{dateLong DueDate}}

In this example, *dateLong* is the name of one of the Format Helpers in OLConnect, and *DueDate* is the name of a data field. When the template is merged with data, this expression returns the value of the DueDate field, formatted as a long date, e.g. "February 23, 2022".

Anything that follows a Helper in an expression is passed to the Helper as a parameter. The parameter doesn't have to be a data field. It can also be another Helper that returns a value. This is called a *subexpression*.

> **Example:** {{dateLong today}}

In this example, *today* is the name of the Helper. It returns the current date. (This Helper is provided by OL Connect.) The current date is then passed to *dateLong*, so that this expression returns the current date formatted as a long date, e.g. "February 23, 2022".

You could also create **custom Helpers**, for example, a Helper that returns tomorrow's date. To learn how to create this and other custom Helpers, see "Creating custom Helpers" on page 1018.

Note: Expressions are evaluated from right to left.

Subexpressions can, but don't have to be, delimited by parentheses. For example, *{{#if eq a b}}* and *{{#if (eq a b)}}* are both valid.

counter

Some types of documents must contain a sequential number. Think of labels and raffle tickets. The `counter` helper generates sequential numbers that update for each record in the main data set.

The helper can take the following parameters. Note that all parameters are optional. If a parameter is omitted, the default value is used.

- start: The starting number. Default value: 1.
- step: The increment for the counting sequence. Default value: 1.
- pad: The number of digits the counter will have. If this number is larger than the current counter value, the padding character will be added to the left of the counter value, until its width is equal to the set value. Default value: 0 (no padding).
- padChar: The character to use for padding. Default value: '0'.

Example:

{{counter start=10}}	10, 11, 12, ...
{{counter step=5}}	1, 6, 11, ...
{{counter start=10 step=5}}	10, 15, 20, ...
{{counter pad=4}}	0001, 0002, 0003, ...
{{counter start=10 step=500 pad=5 padChar="x"}}	xxx10, xx510, x1010, ...

Note: When records are sorted and/or filtered using print presets, the counter will change accordingly.

Tip: If you need to write a custom counter, this How-to will get you started: Handlebars helper: counter.

Date: *today*

OL Connect provides a Helper that returns the current date: `today`.

As parameter it accepts a locale: a language code followed by a 2-letter country code (`de-DE`, `zh-CN`, `fr-CA`, `fr-FR`, etc), as defined by the international standards ISO-639-1 and ISO 3166.

In addition, you can use a format Helper to format the date; see "Date and time Helpers" on page 1017.

Example: {{dateShort today "en-US"}}

Formatting

OL Connect provides a number of Helpers to **format the value** that is the result of an expression. For a list, see "Format Helpers" on page 1015.

Logic Helpers

OL Connect has a number of additional Helpers that allow for the use of logic in expressions.

Helpers that work with **numbers**:

- `eq` (equal), `neq` (not equal) - tests two values for equality
- `gt` (greater than), `gte` (greater than or equal), `lt` (lower than), `lte` (lower than or equal) - compares two number values
- `not`, `or`, `and` - logical operators
- `add` - adds two number values
- `sub` - subtracts two number values
- `mul` - multiplies two number values
- `div` - divides two number values

Helpers that work with **text** strings:

- `concat` - merges two or more string values into a single string and returns the result.
- `startsWith` - tests if a string starts with a certain prefix
- `endsWith` - tests if a string ends with a certain suffix
- `matches` - tests if a string matches a regular expression

> **Note:** The `matches` helper tries to match the entire input string. This means that expressions like {{matches 'contains a fox' 'fox'}} will return `false`. To get `matches` to look for substrings and have the expression return `true`, the regular expression must have **.*** before and after the substring: {{matches 'contains a fox' '.*fox.*'}}

How to use these Helpers

These Helpers can be used **as expression** (returns the result).

> **Example:** Recurs every month: {{eq 'Monthly' recurrence}}
>
> This expression returns "true" if the value of the data field *recurrence* is 'Monthly'.

They can also be used **in subexpressions**, for example of a Block Helper.

> **Example:** {{#if (eq 'Monthly' recurrence)}} Recurs every month {{/if}}
>
> This Block Helper outputs "Recurs every month" if the value of the data field *recurrence* is 'Monthly'.

> **Note:** In OL Connect (as opposed to the original Handlebars library), numbers in Handlebars expressions do not need to be surrounded by quotes. For example, to calculate 1+2 you can write (add 1 2) instead of (add "1" "2").

Block Helpers

Block Helpers are a special kind of Helpers. They look like this:

{{#helperName arguments}} ... {{/helperName}}

With Block Helpers it is possible to use **conditions** and create **loops** in an expression, for example to display content only if a field has a certain value.

Handlebars offers a number of built-in Block Helpers. They are documented on Handlebars' website: https://handlebarsjs.com/guide/builtin-helpers.html.

Of those Helpers, the following are supported in OL Connect:

- **#if**: Conditionally renders a block. An optional {{else}} section inside the block will display when the condition is not true.
- **#unless**: Renders a block if the expression returns a falsy value (i.e. a value that is considered false when encountered in a Boolean context). An optional {{else}} section inside the block will display when the condition is true.
- **#each**: Allows to iterate over a list. An optional {{else}} section inside the block will display when the list is empty.

  > **Note:** Instead of creating table rows with #each, it is highly recommended that you use the built-in "Dynamic Table" on page 974 feature.

- **#with**: Can be used to work directly with the passed object or object property. An optional {{else}} section will display only when the passed value is empty.

> **Note:** Good to know when you are using #if or #unless: an empty array is considered falsy (i.e. evaluated as false). This is normally not the case in JavaScript, but the Handlebars library makes an exception.

Unlike in the original Handlebars library, content provided by a Block Helper is **not automatically HTML-escaped** in OL Connect. Blocks are typically used to generate HTML, so it is assumed that the result consists of well-formed HTML.

Variations with {{else}}: else if

Handlebars does not have a helper for `case` or `switch` statements. Instead, you can combine `else` with another helper. For example: {{else if <condition>}}.

```
{{#if eq policy "a"}}
    Policy A
{{else if eq policy "b"}}
    Policy B
{{else if eq policy "c"}}
    Policy C
{{else}}
    Policy D
{{/if}}
```

The syntax is: **{{else <helper> <args>}}**.

The helper that follows `else` will usually be `if`, but you can use any helper. For example, you can use `unless` to invert the condition: {{else unless <condition>}}, or even use a custom helper.

Where to use Block Helpers

It is recommended to only use Block Helpers in a **Handlebars snippet** (see "Handlebars templates" on page 1023).

This is because in the Designer, sections are edited with an HTML editor which will interpret special characters as HTML, like the **/** in a Block Helper such as {{if}} ... {{/if}}.

HTML mixed with Handlebars expressions is not necessarily valid HTML. Processing it with an HTML parser - like the editor in which sections are written - might break both the Handlebars expressions and the HTML. This happens, for example, when `#each` (a Handlebars Helper) is used to create table rows.
Consider creating a Handlebars template whenever you find yourself switching to Source view to insert expressions and HTML. The Handlebars template editor is a plain text editor, not an HTML parser.

Data variables

In Block Helpers, the data variables **@first**, **@last**, **@index** can be used (see https://handlebarsjs.com/api-reference/data-variables.html).

Tip: In Dynamic Tables it is possible to target the first or last row, or to show the index number of a row, using data variables. For examples, see "Cells with expressions" on page 984.

Note: The data variable **@key** is not supported in OL Connect. OL Connect only supports iterating over arrays and tables, not over arbitrary objects.
Navigating to a parent scope with something like **@../index** is also not supported.

Examples of Block Helpers

#if

The following #if block will output a table row with two cells if the data field named O_L10095 is not false, undefined, null, 0, an empty string, or an empty array.

```
{{#if O_L10095}}
<tr>
    <td>Collective insurance</td>
    <td>{{O_L10095}}</td>
</tr>
{{/if}}
```

#unless and #each

The following Handlebars template outputs a row with a header and a variable number of clause descriptions, based on a detail table called 'Clause', *unless* there are no clauses, in which case the row gets a header and one cell that displays the text 'None'.

```
<table class="policy-table">
<tbody>
{{#unless Clause}}
<tr>
    <th class="h3">Clause(s)</th>
    <td>None</td>
</tr>
{{else}}
    {{#each Clause}}
    <tr>
        <th class="h3">{{#if @first}}Clause(s){{/if}}</th>
        <td>{{Descr}}</td>
    </tr>
    {{/each}}
{{/unless}}
</tbody>
</table>
```

#with and #each

This #with block calls a custom Helper (see "Creating custom Helpers" on page 1018) that returns an object with the properties: object, amount, coverage and premium. The object's properties are accessed directly inside the block. The #with block is executed for each of the records in a detail table called *insurance_coverage*.

```
{{#each insurance_coverage}}
    {{#with (getObjectAndCoverage)}}
    <tr>
        <td>{{object}}</td>
        <td class="curr">€</td>
        <td>{{amount}}</td>
        <td>{{coverage}}</td>
        <td class="curr">€</td>
        <td>{{premium}}</td>
    </tr>
    {{/with}}
{{/each}}
```

Format Helpers

OL Connect provides a number of Helpers to **format a value** that is the result of an expression.

A "Helper" is a function that can be called in an expression.

Example: {{dateLong DueDate}}

In this example, *dateLong* is the Helper, and *DueDate* is the name of a data field that contains a date. The expression returns the date, formatted as a long date, e.g. *February 23, 2022*.

> **Note:** The rules of the current locale also influence the way the data is formatted; see "Locale" on page 921.

To change the format of the output of an expression, simply type the name of a Helper in the expression. Which Helpers work depends on the type of the data.

This topic lists the available Helpers that can format a value by data type.

> **Note:** The names of these Helpers are case sensitive. For example, "upperCase" works, but "uppercase" does not.

Text Helpers

These are the Format Helpers that can be used with text.

- **lowerCase** transforms all characters to lowercase.
- **upperCase** transforms all characters to uppercase.
- **properCase** transforms the first character of each word to uppercase and all other characters to lowercase.
- **sentenceCase** transforms the first character of the first word to uppercase. This option is only available in Handlebars expressions and scripts, not in the Text Script and Text Helper wizards.

Number Helpers

The following Format Helpers work with numbers, i.e. data fields of the type Currency, Float and Integer.

- **currency** formats a number as an amount of money, with a currency symbol. Optionally, you can specify a custom pattern. Put the pattern after the name of the data field, and enclose it in double quotes.

  > **Example:** {{currency Total "#.0"}}

  For available patterns, see "Number patterns" on page 1511.
- **currencyNoSymbol** formats a number as a currency whilst omitting the currency symbol.

- **number** allows you to enter a custom format mask. For example, the pattern `000000` means that the number should count six digits; leading zeros are added to numbers shorter than six digits. For an overview of pattern symbols see "Number patterns" on page 1511 and http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html.
  In the data mapping configuration, set the field type to Integer, Float or Currency. When you open a data file or database, all fields are text fields (fields of the type `string`).

  Example: {{number Total "#.0"}}

- **grouped** formats a number using a thousands separator.

Note: Which currency symbol and which thousands separator are used depends on the Locale; see "Locale" on page 921.

Note: In output, spaces in a number or amount of money are transformed into non-breaking spaces to ensure that it is not split onto different lines.

Date and time Helpers

The following Format Helpers work with dates (i.e., Date data fields, and String data fields that follow the ISO (2024-04-02) or UTC data string format). A date can contain a date and time.

Note that this requires the data field to be of the type Date. These Helpers don't work with plain text.

- **date** formats a date object using a custom pattern. Put the pattern after the name of the data field, and enclose it in double quotes.
  The custom pattern may consist of pattern letters, separating symbols and quoted text, for example: "MMMM dd, yyyy"; see "Date and time patterns" on page 1491.
- **dateLong** formats a date as long string representation, for example **April 1, 2022**.
- **dateMedium** formats a date as medium string representation, for example **Apr 1, 2022**.
- **dateShort** formats a date as short string representation, for example **4-1-2022**.
- **dateTime** formats a date and time object using a custom pattern. Put the pattern after the name of the data field, and enclose it in double quotes.
  The custom pattern may consist of pattern letters, separating symbols and quoted text, for example: "yyyy.MM.dd G 'at' HH:mm:ss z"; see "Date and time patterns" on page 1491.

- **dateTimeLong** formats a date and time as long string representation, for example **April 1, 2022 12:00:00 EDT AM**.
- **dateTimeMedium** formats a date and time as medium string representation, for example **Apr 1, 2022 12:00:00 AM**.
- **dateTimeShort** formats a date and time as short string representation, for example **4/1/22 12:00 AM**.
- **time** formats a time using a custom pattern. Put the pattern after the name of the data field, and enclose it in double quotes.
  The custom pattern may consist of pattern letters, separating symbols and quoted text, for example: "'at' HH:mm:ss z"; see "Date and time patterns" on page 1491.
- **timeLong** formats a time as long string representation, for example **12:00:00 EDT AM**.
- **timeMedium** formats a time as medium string representation, for example **12:00:00 AM**.
- **timeShort** formats a time as short string representation, for example **12:00 AM**.

Creating custom Helpers

A Helper is a function that can be used in an expression.

In both Handlebars and OL Connect, a number of Helpers are predefined (see "Using functions in expressions: Helpers" on page 1009), but you can also create your own Helpers. (See the Handlebars guide: https://handlebarsjs.com/guide/#custom-helpers.)

This topic explains how to create a custom Helper in OL Connect.

Let's say you want to create a Helper that returns tomorrow's date. The name of the Helper should be *tomorrow*, so that it can be used in an expression as follows:

Example: {{tomorrow}}

Tip: OL Connect provides a Helper that returns the current date: {{today}}.

Here's how you do that.

1. On the **Scripts** pane, click the black triangle next to the **New** button and click **Control Script**.

   Control Scripts are executed first. See "Control Scripts" on page 1098 and "The script flow: when scripts run" on page 1082.

2. Enter a name for the script in the Name field. This name will appear on the Scripts pane. It doesn't have to be the same as the name of the Helper that can be used in expressions. One Control Script can register multiple Helpers.
3. Write code that registers the Helper, using the `Handlebars.registerHelper(name, helper)` function.
    - `name` is the name of the Helper by which it should be called in expressions. Put the name in quotes.

      Tip: To make it easy to identify your own Helpers, you can start the name with a special character such as an underscore.

    - `helper` is the code that should run when the Helper is called from an expression.

    Here is an example:

```
Handlebars.registerHelper('tomorrow', function() {
    var tomorrow = new Date();
    tomorrow.setDate(tomorrow.getDate() + 1);
    return tomorrow;
});
```

4. Click **OK**. The new script appears on the **Scripts** pane in the **Control Scripts** folder.

Now that the Helper is registered, the expression *{{tomorrow}}* in a template will return tomorrow's date.

In the same expression you could use a Helper that formats the date. For example: *{{dateLong tomorrow}}* will format the date as a long date, e.g. February 23, 2022. (For a list of Format Helpers, see "Format Helpers" on page 1015.)

If you'd like to format the value in the script itself, you can use the functions provided by the `formatter` object in standard Designer scripts; see "formatter" on page 1486.

For example, this code registers a Helper that formats the output as a 'medium' date.

```
Handlebars.registerHelper('tomorrow', function() {
    var tomorrow = new Date();
    tomorrow.setDate(tomorrow.getDate() + 1);
    return formatter.dateMedium( tomorrow );
});
```

Tip: Hash arguments allow named arguments to be passed to a Helper. They make it possible to pass a variable number of arguments, and to pass fields from the parent scope to a Helper. See Hash arguments.

Note: There are a few differences between the official Handlebars library and OL Connect. See "Unsupported features" on page 1032.

Examples of custom Helpers

Format as currency or display "N/A"

You may want to format the value of a particular data field as a currency, or display N/A if it is not a positive value. Here's code that registers a Helper that only works for a data field called *Total*.

```
Handlebars.registerHelper('checkValue', function(){

    let dataValue = parseFloat( record.fields.Total )
    let returnValue = 'N/A'

    if( dataValue > 0) {
        return formatter.currency( dataValue )
    }

    return returnValue
})
```

The expression **{{checkValue}}** formats the value of the field *Total*. It cannot be used to format any other fields.

The same Helper could be written in a way that it can be used with any (numeric) data field.

```
Handlebars.registerHelper('checkValue', function( val ){

    let dataValue = parseFloat( val )
    let returnValue = 'N/A'

    if( dataValue > 0) {
        return formatter.currency( dataValue )
    }

    return returnValue
})
```

The expression that formats the value of the field *Total* would now be: **{{checkValue Total}}**. To use this Helper with other numeric data fields, simply replace *Total* with the name of the other data field.

If you only want to get "N/A" if the value is 0 or 0.0, and format both positive and negative values with a currency symbol, the code could check if the value is falsy:

```
Handlebars.registerHelper('checkValue', function( val ){
    if(!val) {
        return 'n/a'
    } else {
        return formatter.currency( val )
    }
}
```

This code outputs the string 'n/a' if the value is falsy; otherwise it outputs the value, formatted as a currency.

The code can be made even shorter by using the conditional (ternary) operator, which is a short alternative to an if...else statement.

```
return val ? formatter.currency( val ) : 'n/a'
```

Coloring a negative value

Values in your data could be positive or negative. Let's say you want a Helper that colors any negative values red. Here's the code for such a Helper.

```
Handlebars.registerHelper('highlightNegative', function ( value ) {
    let result = value
    if( -1 === Math.sign(value) ) {
        result = '<span style="color:red">' + value + '</span>';
        }
    return new Handlebars.SafeString(result);
})
```

The name of the Helper is *highlightNegative*. The Helper checks whether the value that it gets passed by an expression is negative, using a JavaScript function: Math.sign() (see the documentation of Mozilla). If it is, it wraps the value in a Span with a *style* attribute that has some CSS to color the value red.

Once you've added this Helper, it can be used in any expression that returns a number value.

Example: {{highlightNegative Total}}

This expression calls the Helper with the name of a data field, Total, that has a number. If the value of the data field is -15, the output of this expression will be -15.

Inserting a hyperlink

Let's assume you have hyperlinks in your data as well as texts to use as label for the hyperlinks. For example:

```
[
{
"url": "http://www.nu.nl",
"label": "nu.nl"
}
]
```

The following code registers a Helper called *makeHyperlink* that will turn these values into a hyperlink in the text.

```
Handlebars.registerHelper('makeHyperlink', function () {
    let result = `<a href="${this.url}">${this.label}</a>`;
    return new Handlebars.SafeString( result );
})
```

Note: OL Connect 2025.1 added support for preventing URL-encoding for expressions in `href` and `src` attributes. This can be done through:

- `Handlebars.SafeString(string)`
- Wrapping the text in a triple-stache `{{{...}}}`

Tip: For information on the `SafeString(string)` function see SafeString() explained

In the template you would call the function in an expression.

Example: Go to {{makeHyperlink}}

Targeting the first and/or last item

A Helper can find the first item, last item and the index of the current item in an array via the `options` object. If a Helper is registered with `options` as the last parameter, the `options` object is automatically passed to the Helper when it is called.

This is an example of a Helper that uses `options.data.first` to display the first item in an array.

```
Handlebars.registerHelper("showLabel", function( label, options ) {
    let myLabel = ""
    if (options.data.first) {
        myLabel = "<b>" + this.label + "</b>";
    }
    return new Handlebars.SafeString(myLabel);
});
```

In a Dynamic Table, this Helper could be called from an expression in a cell.

Example: {{showLabel ../year}}

The above example passes the field *year* (from the main record) to the Helper. The current detail table and the `options` object are passed to it automatically.

Note: In a Dynamic Table with expressions you can access these data variables in Block Helpers directly with @first, @last and @index.
Example: {{#if @first}}{{../year}}{{/if}}
See "How to target the first or last row" on page 985.

HTML-escaping in a Helper

If a Helper needs to HTML-escape a string you can call Handlebars.escapeExpression() in that function.
For example, if you would want to HTML-escape the value of `this.field`, but not " " or " ", you could write:

```
function() { return "<p>" + Handlebars.escapeExpression(this.field) + "</p>"; }
```

Note: As a rule of thumb, put as much content in the template as possible and let Helpers generate as little content as possible.

Handlebars templates

OL Connect has a special kind of snippets called "Handlebars templates". Just like other types of snippets (see "Snippets" on page 867), Handlebars templates are stored in the **Snippets** folder on the **Resources** pane, but their file name ends in **.hbs**.

Handlebars templates look like regular text with embedded Handlebars expressions. A Handlebars expression is some contents enclosed by double curly braces: **{{** ... **}}**.

Example: `<p>Hello {{firstname}}!</p>`

More about expressions and the functions that you can use in expressions can be found in the topics "Handlebars expressions" on page 1005 and "Using functions in expressions: Helpers" on page 1009.

In OL Connect, a Handlebars template may be included in a section (which itself functions as a Handlebars template) or in another Handlebars template.
In the Handlebars library, such Handlebars templates are referred to as ***partials***.

Why use a Handlebars template?

You can use Handlebars expressions in all **sections** (see "Handlebars in OL Connect" on page 998 and "Variable data in text: expressions" on page 999). Nevertheless, in a number of cases it is better or even unavoidable to work with Handlebars snippets.

- With Handlebars templates you can easily load content with expressions into your template dynamically, based on data.
- HTML mixed with Handlebars expressions is not necessarily valid HTML. Processing it with an HTML parser - like the editor in which sections are written - might break both the Handlebars expressions and the HTML. This happens, for example, when `#each` (a Handlebars Helper) is used to create table rows.
  Consider creating a Handlebars template whenever you find yourself switching to Source view to insert expressions and HTML. The Handlebars template editor is a plain text editor, not an HTML parser.

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Creating a Handlebars template

To create a new, empty Handlebars template:

1. On the **Resources** pane, right-click the **Snippets** folder and select **New Snippet**.
2. Select the type of snippet that you want to create: **Handlebars template**.
3. Give the snippet a name.
4. Double-click the new file to open it in the Designer and fill it with HTML text and Handlebars expressions.

   Handlebars templates can include other Handlebars templates - see "Using Handlebars templates" on the next page.

   Inline partials are supported as well. See the two examples at https://docs.w3cub.com/handlebars/partials#inline-partials.

Tip: To keep an overview you could group the Handlebars templates in a subfolder of the Snippets folder on the Resources pane.

Note: The editor for Handlebars snippets does not have a Design view. HTML with Handlebars expressions is not necessarily valid HTML. Processing it with an HTML parser might break both the Handlebars expressions and the HTML.

Note: There is no need to register a Handlebars template as a partial, like in the original Handlebars library. OL Connect automatically registers all Handlebars

templates in the Snippets folder, so that they can be referenced in expressions by name, without path and extension.

Using Handlebars templates

To include a Handlebars template - a 'partial' - in either a section or another Handlebars template, write an expression that refers to it by name:

{{+partialname}}

Note: If the name of the partial contains a space, the space needs to be escaped. Either surround the name with quotes or replace the space with %20. For example, a partial called “par tial.hbs” can be referenced as follows: {{+par%20tial}} or {{+('par tial')}}.

The name can be:

- The name of an .hbs snippet in the template's **Snippets** folder.

  Example: `{{+Snippet1}}`

  {{+snippets/Snippet1.hbs}} would also work, but the path and extension can be omitted.

- The name of an .hbs snippet on **disk**.

  The complete syntax is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in **file:///<path>** - note the **three** forward slashes after **file:**.
  In the remainder of the path you can either use escaped backward slashes:

  Example: "file:///C:\\Users\\Administrator\\Desktop\\Partial1.hbs"

  or forward slashes:

  Example: "file:///C:/Users/Administrator/Desktop/Partial1.hbs"

- The name of a **remote** .hbs snippet (starting with http:// or https://). It could also be a URL that retrieves a Handlebars template from an endpoint.

  Example: {{+http://localhost:1880/policies?p=travel}}

Passing data

You could pass data to the partial using one or more **hash arguments**.

Example: `{{+myPartial test=123}}`

See Hash arguments.

Expressions inside a partial are resolved using the record, unless a **custom context** is passed to the partial.

For example, if a field in your data has an `address` object, that object could be passed as follows:

Example: `{{+myPartial address}}`

In **section clones**, the current detail record could be passed to a partial. The following expression uses the `lookup` helper (see lookup in the Handlebars documentation) to find the current detail record in a detail table called 'policies' by the index which is set in a clone's `meta` property.

Example: {{+myPartial (lookup @root.policies @meta.detail-table-record)}}

See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

When a partial is missing

If there is no partial with the specified name, Handlebars can generate some other content, called **failover content**. Failover content can be specified using block syntax. The expression that calls the partial is the start of a block. An expression with a closing tag and the name of the partial signals the end of the block. The content between these expressions will be used if the called partial is missing.

Example: {{#+ myPartial }} Failover content {{/myPartial}}

The expression in this example will render "Failover content" if no "myPartial" partial is found.

Loading Handlebars templates dynamically

There are several ways to load Handlebars templates - aka 'partials' - dynamically, based on data.

1. You could use a **Block Helper** (see "Using functions in expressions: Helpers" on page 1009).

Tip: Partials can include other partials. To keep the template clean and clutter-free it's best to put the Block Helper itself in a partial as well.

2. You could dynamically select a partial by using a **sub expression** between parentheses. The sub expression should evaluate to the name of a partial.

   Example: `{{+ (whichPartial) }}`

   - If `whichPartial` is a data field that returns the name of a partial, the template renders the partial whose name is found in the data field.
   - `whichPartial` could also be the name of a function. Note that to use a function in this way, it must be registered as a *Helper*. See "Creating custom Helpers" on page 1018.

3. Handlebars templates can also be loaded dynamically in a **script**. This requires the Handlebars templates to be rendered before they are added to a section. See "Using Handlebars templates in a script" on page 1032.
4. Instead of loading a different partial you could load the same partial and pass different data to it using one or more **hash arguments**. See Hash arguments.

Partials

Partials are normal Handlebars templates that may be called directly by other templates. This topic explains how to work with Handlebars partials in OL Connect.

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Creating a partial

Since partials are normal Handlebars templates, you can create them just like any other Handlebars template; see "Creating a Handlebars template" on page 1024.

Tip: To keep an overview you could group the partials in a subfolder of the Snippets folder on the Resources pane.

Using partials

To include a partial in either a section or a Handlebars template (i.e. another partial), write an expression that refers to the partial by name:

{{+partialname}}

".hbs" can be omitted from the name of the partial.

> **Note:** If the name of the partial contains a space, the space needs to be escaped. Either surround the name with quotes or replace the space with %20. For example, a partial called “par tial.hbs” can be referenced as follows: {{+par%20tial}} or {{+('par tial')}}.

Partials don't have to be located in the template. They can also be loaded from a disk or from an endpoint (which could facilitate a search in a database or CMS). For instance, you can load a partial from a URL like this:

{{+http://localhost:1880/policies?p=travel}}

To include a partial in a **Handlebars template** (not a section), you can also:

- Drag the partial from the **Snippets** folder on the **Resources** pane to the desired location in the Handlebars template. This copies the contents of the partial into the other.
- Use the expression: {{**>** partialname}}. In sections this does not work, because the greather-than-sign will then be interpreted as HTML.

Custom context

Expressions inside a partial are resolved using the record, unless a **custom context** is passed to the partial. For example, if a field in your data has an `address` object, that object could be passed as follows:

> **Example:** `{{+myPartial address}}`

In **section clones**, the current detail record could be passed to a partial. The following expression uses the `lookup` helper (see lookup in the Handlebars documentation) to find the current detail record in a detail table called 'policies' by the index which is set in the clone's `meta` property.

> **Example:** `{{+myPartial (lookup @root.policies @meta.detail-table-record)}}`

See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Hash arguments

You could pass data to a partial using one or more **hash arguments**.

Example: `{{+myPartial test=123}}`

See Hash arguments.

When a partial is missing

If there is no partial with the specified name, Handlebars can generate some other content, called **failover content**. Failover content can be specified using block syntax. The expression that calls the partial is the start of a block. An expression with a closing tag and the name of the partial signals the end of the block. The content between these expressions will be used if the called partial is missing.

Example: {{#+ myPartial }} Failover content {{/myPartial}}

The expression in this example will render "Failover content" if no "myPartial" partial is found.

Dynamic partials

There are several ways to load partials dynamically, based on data.

1. You could use a **Block Helper** (see "Using functions in expressions: Helpers" on page 1009 and "Using Handlebars templates" on page 1025).

   Tip: To keep the template clean and clutter-free it's best to put the Block Helper itself in a partial.

2. It is also possible to dynamically select a partial by using a **sub expression** between parentheses. The sub expression should evaluate to the name of a partial (with or without the extension: ".hbs").

   Example: `{{+ (whichPartial) }}`

   If `whichPartial` is a data field that returns the name of a partial, the template renders the partial whose name is found in the data field.
   `whichPartial` could also be the name of a function. Note that to use a function in this way, it must be registered as a *Helper*. See "Creating custom Helpers" on page 1018.

3. Finally, instead of loading a different partial you could load the same partial and pass

different data to it. This could be either a **custom context** (see "Custom context" on page 1028) or one or more **hash arguments**. See Hash arguments.

Inline partials

Inline partials are supported as well. See the two examples at https://docs.w3cub.com/handlebars/partials#inline-partials.

Handlebars API

This topic lists the functions of Handlebars that are supported in Designer scripts. It also lists features that are unsupported in OL Connect.

Note: The information in this Online Help focuses on the implementation of Handlebars in OL Connect. For general information about Handlebars and how to use it, see the following web sites: https://handlebarsjs.com/ and https://devdocs.io/handlebars.

Functions

Function	Description

compile('template')	Compiles the specified Handlebars template* into a function that can be called with data**. For example:
	`const template = Handlebars.compile('snippets/Template1.hbs');`
	`const result = template(record);`
	Note: Passing options is not supported in OL Connect.

render('template', data)	Renders a specified Handlebars template* by compiling the specified Handlebars template into a function and immediately calling that function with data**.
	`Handlebars.render( "snippets/Template.hbs" )`
	is identical to:
	`const compiledTemplate = Handlebars.compile( "snippets/Template.hbs" )`
	`compiledTemplate( record )`
	If data is passed, that data is used to render the template.
	If no data is passed, the current **record** is used to render the template.
	The result of this function is **HTML**.
	See also: "Handlebars templates" on page 1023.
	Note: A section's `meta` property is not accessible to Handlebars expressions in a Handlebars template that is rendered using a Control Script (see "Inserting a Handlebars template using a script" on page 1033). Use a Standard Script instead.

Function	Description
registerHelper('name', helper)	Registers a function as Helper with the specified name. Here is an example: `Handlebars.registerHelper('today', function() { return new Date(); });` See: "Creating custom Helpers" on page 1018. **Note:** Registering multiple Helpers with a single call to registerHelper() is not supported.
escapeExpression()	HTML-escapes the supplied string. This means that those characters that have a special function in HTML: **& < > \" ' ` =** are replaced with HTML character references. For example: **&** will be changed into **&**. See also: "HTML values" on page 1008. This function does the same as `escapeHTML()`, which was introduced in OL Connect 2022.1 and is now deprecated.

*** `template`**

In all these functions, 'template' can be:

- the name of an .hbs snippet in the template (for example: `snippets/Snippet 1.hbs`
- the name of an .hbs snippet on disk (starting with file:///)
- the name of a remote .hbs snippet (starting with http:// or https://)
- a string that contains HTML and Handlebars expressions.

With a snippet on disk, the complete syntax is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in **file:///<path>** - note the **three** forward slashes after **file:**.
In the remainder of the path you can either use escaped backward slashes:
"file:///C:\\Users\\Administrator\\Desktop\\Handlebars_LoadFile.hbs"
or forward slashes:
"file:///C:/Users/Administrator/Desktop/Handlebars_LoadFile.hbs"

Note: It is not possible to use `loadhtml()` in Handlebars functions. HTML with Handlebars expressions is not necessarily valid HTML. Processing it with an HTML parser might break both the Handlebars expressions and the HTML.

**** `data`**

The `data` can be the current record or part of it, or another JavaScript object.

it is also possible to pass the record or part of it along with other data, since JavaScript objects may contain multiple objects. Here is an example.
The following script loads JSON and then passes that JSON and the current record to a Handlebars template.

```
const myJson = loadjson('snippets/myJson.json');
Handlebars.render(myHandlebarsTemplate, { json:myJson, record:record })
```

Assuming the *json* in this example has a key called *key1* and the record has a field called *field1*, expressions in the template could look like this: {{json.key1}} {{record.field1}}.

Unsupported features

The following features are not supported in Handlebars **templates** in OL Connect:

- Inline escapes.
- Raw block helpers.
- Passing options when compiling a Handlebars template.

The following Handlebars **helpers** features are not supported in OL Connect.

- In the `log` helper, only the **info** error level is supported.
- The data variable **@key**. OL Connect only supports iterating over arrays and tables, not over arbitrary objects. (@first, @last, @index and @root are supported.)
- Registering multiple helpers with a single call.
- The `blockHelperMissing` and `helperMissing` hooks.
- Passing `options.helpers` and `options.partials` arguments to helpers. (`options.hash`, `options.data`, `options.fn` and `options.inverse` are supported.)

Using Handlebars templates in a script

Handlebars expressions in an HTML snippet do not get replaced with values if the snippet is loaded through a script. Handlebars templates (also called Partials) *can* be loaded through a script. They need to be *rendered* before they are added to a section. When a Handlebars template is rendered, the expressions are replaced with values from input data of your choice. How to do this is explained in this topic.

If you are new to scripting in the Designer, first read: "Writing your own scripts" on page 1064.

Inserting a Handlebars template using a script

It takes only two lines of code to replace the expressions in a Handlebars template with values and add the result to a section.

1. On the **Scripts** pane, click on the black triangle next to the New button and select **Standard Script**.
2. Give the script a name.
3. Enter the selector that identifies the elements in the template for which the script should run. (See: "Selectors in OL Connect" on page 1083.) For example, *#Snippet1* refers to an element that has the ID *Snippet1*.
4. Write a line of code that calls the function `Handlebars.render(template, data)`.
   - The `template` can be the name of a Handlebars template (.hbs) stored in the template, on disk, or remotely (http:// or https://).

     With a snippet on disk, the complete syntax is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in **file:///<path>** - note the **three** forward slashes after **file:**.
     In the remainder of the path you can either use escaped backward slashes:
     "file:///C:\\Users\\Administrator\\Desktop\\Handlebars_LoadFile.hbs"
     or forward slashes:
     "file:///C:/Users/Administrator/Desktop/Handlebars_LoadFile.hbs"
   - The `data` can be the current record or part of it, or a JavaScript object (which may include the record or part of it - see "Handlebars API" on page 1030 for an example).
     If no data is passed, the **current record** will be used.

Example:

```
var html = Handlebars.render( "snippets/policy-info.hbs", record
);
```

Note: If a Handlebars template is rendered with part of the current record, the template still has access to the *entire* record and can navigate up and outside of that part, to a higher level in the current record.
If a JavaScript object is passed, the template only has access to the passed data, and it does not have access to the current record.

> **Note:** A section's `meta` property is not accessible to Handlebars expressions in a Handlebars template that is rendered using a Control Script (see Inserting a Handlebars template using a script). Use a Standard Script instead.

5. Write a line of code that adds the rendered HTML to the content of the section. The following script replaces the elements to which the script applies with the contents of the Handlebars template.

   Example: `results.replaceWith( html )`

6. Save the script. Make sure that there is at least one element in the section that matches the selector of the script.

Alternative: compile and call the template

The `render()` function actually does two things:

1. **Compile the Handlebars template into a function**

   When a Handlebars template is compiled, it is actually compiled into a JavaScript function.

2. **Call the function to replace expressions with data**

   The second step is to call the resulting function, passing in some data. The function will replace the expressions with values from the data and will then return HTML.

Instead of calling `Handlebars.render()` you could call `Handlebars.compile()` and then call the resulting function.

Example:
```
var myFunction = Handlebars.compile( "snippets/policy-info.hbs", record);
let html = myFunction( record );
results.replaceWith( html )
```

Handlebars sample code from an online source will use the two separate steps since `Handlebars.render()` is only available in OL Connect.

See also: "Handlebars API" on page 1030.

Examples

Rendering a Handlebars template repeatedly

This script loops over a detail table called *Policies*, each time passing different data to the same Handlebars template.

```
let policies = record.tables.Policies;
let html = '';
for( var i = 0; i < policies.length; i++) {
html += Handlebars.render( 'snippets/policy.hbs', policies[i] );
}
results.replaceWith( html );
```

Selecting a Handlebars template based on a data field

An approach that is also used with HTML snippets is to use the value of a data field in the current record to determine the name of the Handlebars template to use.

With Handlebars templates there are multiple ways to do this; see "Loading Handlebars templates dynamically" on page 1026.

This is an example of a script that dynamically loads Handlebars templates:

```
let html = ''
let policydata = record.tables.Policy
for (var i=0; i < policydata.length; i++) {
    let templateName = 'snippets/' + policydata[i].fields['snippetName'] + '.hbs'
    html += Handlebars.render( templateName, policydata[i] )
}
results.replaceWith( html );
```

Preferences

The Preferences dialog is used to modify the general software preferences. Changes made in this dialog affect the software globally, not individual templates and data mapping configurations.

The Preferences dialog is separated into individual tabs, where each tab controls certain aspects of the software.

To open the Preferences dialog, select **Window > Preferences**.

Clean-up Service preferences

The Clean-up Service defines how the Connect database and the temporary files created during Connect production runs are cleaned up after the production run has finished.

As part of the job production process OL Connect uses a database for intermediate storage and also creates various temporary "managed" files. These files include data extractions, configuration files and any intermediate files created during the production process. Connect keeps track of all these files through references held within the Connect database.

All the files created and the database references to them are stored for a set amount of time in order to allow Connect to reuse them. However, we do not want to store these

indefinitely, because the database would run out of space. The solution is to use the "Clean-up Service" to remove the temporary data and files once they are no longer needed. This clean-up service is usually managed by the Server Engine.

The more items that are present in the database, and the larger they are, the more time and processing power (CPU) that will be required for cleaning them up. Thus a regular Clean-up of the database (as often as possible) is recommended.

This is especially the case if items are not going to be retrieved from the database at a later date. i.e. If the Connect job is not going to be re-run.

The clean-up can always be set to run outside of business hours (see the "Run according to the cron schedule: Enter the interval at which the Clean-up service runs. " below option below), to reduce impact upon Production systems.

The values below define when the specified targets are to be *set* as being ready for deletion, not *when* they are actually deleted. The actual deletion occurs only as per the cron job scheduling; or when OL Connect is started (if **Run at application start up** is selected); or when the **Run Now** button is pressed.

- **Enable clean-up service**: Check to enable the Clean-up services. When checked, either or both of the *Database clean-up* and *File clean-up* services can be set individually. If the box is not checked, then no Clean-up will occur.
- **Run at application start up**: Click to start the clean-up service when the Designer module is opened, or the Managing Service is started.
- **Run according to the cron schedule**: Enter the interval at which the Clean-up service runs.

  To understand how to write a cron job schedule, please refer to the Quartz Scheduler tutorial: http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/tutorial-lesson-06.html.

  > **Note:** If the **Product managing the service** is set to Designer, then the Designer *must be running* at the time that the cron job is scheduled, for the Clean-up to run.

- **Product managing the service**: Select which of the applications will run the service.

  > **Note:** The **Server Engine** is set as the default as it is generally considered the best option.
  > This is particularly the case when using a scheduled cron job, as the Sever

Engine is always running, whilst the Designer might well not be at the scheduled time (in which case the clean-up will fail to run).

- **Database Clean-up Service**:
  - **Allow database clean-up service**: Select this checkbox to enable the database Clean-up settings, and enable the actual clean-up.
    - **Threads to use for database deletions**: The number of Threads to be used in the clean-up. OL Connect is a multi-threaded application, and the clean-up is likewise.

      Tip: The default number of threads is considered the best compromise for running both clean-up and production jobs simultaneously. If experience suggests that the clean-up is not running efficiently, then upping the number of threads here would be recommended. Conversely, if production appears to be suffering courtesy of the clean-up process, then reduce the number of threads here.

      In general, higher end machines (those with multiple cores) will allow a higher numbers of threads, whilst low end machines will perform better with a lower number of threads.

    - **Number of entities in each deletion batch**: The number of entities to be deleted at a time. This is done to break the clean-up into smaller chunks. This improves OL Connect clean-up responsiveness, whilst the clean-up is occurring.
      The number selected here applies to all the following settings.
      i.e. a selection of 1,000 would delete 1,000 data records within a **Data Set**, 1,000 content items within a **Content Set**, and so on.
    - **Minimum time to retain Data Sets**: The minimum time a Data Set (and all the records it contains) is retained within the database before being set for deletion.

      Tip: In order to prevent attempts at deleting database objects which might still be in use, it is recommended that all **Minimum time** retention values *should always be set to at least the length of your*

longest job. Preferably with some extra time added, for good measure.

For example, if your longest job takes 45 to 50 minutes to run, then set the retention time to 55 minutes (or 1 hour) to ensure that all the database objects created during job processing survive for the duration of the job.

- **Minimum time to retain Content Sets**: The minimum time a Content Set (and all the content items it contains) is retained within the database before being set for deletion.
- **Minimum time to retain Job Sets**: The minimum time a Job Set (and all the jobs information it contains) is retained within the database before being set for deletion.
- **Minimum time to retain Managed Files**: The minimum time file references (to files such as data mapping configurations and templates) are retained within the database before being set for deletion.
- **Minimum time to retain other entities**: The minimum time any orphaned data (such as Finishing tables, Media tables, DataModels and Properties tables) are retained within the database before being set for deletion.

- **Database Partition Settings**:
  - **Use Database Partitioning**: Select to use Database Partitioning.
    - **Empty partition count**: The number of empty partitions that are created each clean-up run. This defaults to 24.
    - **Partition Size**: Enter the length of time before partitions are switched. This can be entered in minutes, hours, days, weeks or months.
- **File Clean-up Service**:
  - **Allow file clean-up service**: Check to automatically detect orphan files and set them for deletion. Orphan files could be resources and internal files used by Connect, but which are not needed by any running job.
    - **Minimum time to retain orphaned files**: The minimum time during which orphaned files are kept in the database before being set for deletion.

Note: In Clustered environments it is ***only*** the server which has the backend Database module installed that should run any of these options. With the exception of **Allow**

orphan file clean-up. This option refers to local files and should be run on ***each*** machine in the cluster. See "Server Clustering" on page 124.

The Clean-up Services preferences also provides you with buttons to :

- **Run Now**: This will run the clean-up service immediately.
- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Data Model preferences

ExtraData control

ExtraData fields were originally intended to be used in a Workflow configuration to add data to the Data Model. Nowadays, **runtime parameters** are the preferred way to pass data at runtime to a data mapping configuration (see "Properties and runtime parameters" on page 387) or template (see "Runtime parameters" on page 608).

- **View ExtraData**: Check this option to display the **Show ExtraData Fields** icon at the top of the Data Model. The topic: "Adding fields and data via Workflow" on page 424 explains how ExtraData fields are used.

DataMapper preferences

Common DataMapper preferences

- **Timeout for individual steps (in seconds)**: Specify how long, in seconds, each task is allowed to run at design time before it errors out. The timeout value applies to all steps except the Preprocessor and Postprocessor steps and when the boundaries are being set. Max: 120 seconds, default: 15 seconds.

DataMapper XML

- **Display New Line Character as ¶**: Check to show line returns as ¶ in the Data Viewer, when XML files are shown. If the option is unchecked, you will not see spaces and line returns after element names in the Data Viewer.

Default Format

DataMapper stores user preferences for the Date, Number and Currency formats. By default, the user preferences are set to the system preferences. These user preferences

become the default format values for any newly created data mapping configuration. Format settings can also be defined at the data mapping configuration level ("Data mapping configurations" on page 337) and/or per field in the Data Model. Any format settings specified in an existing field are always used, regardless of the user preferences or data source settings.

- **Negative Sign Before**: Any value in a numeric field that has a "-" sign is interpreted as a negative value.
- **Decimal Separator**: Set the decimal separator for a numerical value.
- **Thousand Separator**: Set the thousand separator for a numerical value.
- **Currency Sign**: Set the currency sign for a currency value.
- **Treat empty as 0**: A numerical empty value is treated as a 0 value.
- **Date/Time Format**: Set the date format for a date value.
  - **Automatic**: Select this option to parse dates automatically, without specifying a format. This is the default setting for new Date fields.
  - **ISO8601**: This setting allows for dates with different timestamp formats, or belonging to different time zones, to be parsed inside a single job. Dates that do not include a specific time are automatically considered to use the current locale's time zone.
    Select the ISO template to be used when parsing the timestamp. Other ISO8601 formats can be handled via the Custom option.
  - **Custom**: Set a custom date format. For the markers available in the DataMapper see "Date" on page 439.
- **Date Language**: Set the language for a date value (ex: If English is selected, the term May will be identified as the month of May).
- **Use offset from UTC**: Select the default time zone, which is to be used to extract any timestamp that does not already include time zone information with the time.

Data Model display option: In the Data Model, **time stamps** can be:

- **Displayed in Local Time**: The time is formatted in accordance with the current operating system's regional settings.
- **Displayed as UTC/ISO**: The time is displayed in UTC (Coordinated Universal Time) following ISO 8601.

Note that this setting only impacts how dates are displayed in the Data Model. It does not change how dates are stored in the Connect database, nor how they are displayed in a template.

The DataMapper preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Database Connection preferences

Dialog used to change the OL Connect back-end Database.
This dialog supports the swapping of the back-end database between various vendor databases. Note, however, that the alternate vendor database(s) *must already be installed and available* in order to swap to them.

This is not a migration tool. It is a simple connection tool, that enables shifting to a different back-end database. Any existing data will **not** be transferred/ migrated between the databases, and any existing File Stores will be cleansed by the Clean-up Service after the swap.

When shifting to a different back-end database, the changes won't be applied until OL Connect is restarted. Including the Connect services. A full machine restart is recommended, as this provides the cleanest restart of all the services.

- **Basic Connection Settings** selections:
  - **Database vendor**: Select the database type from the drop down list.

    Note: Moving from one vendor database to another will reset all screen selections to defaults, regardless of what may have been previously selected.

  - **Database URL**: This is a read-only summation of the current database connection settings.

    Tip: If the **Test Connection** button shows that the database cannot be successfully connected to using the selected settings, then the contents of this field could be used to try to connect to the database outside of OL

Connect. This should help in determining and refining the acceptable connection options.

- **Hostname**: Enter the IP Address or alias of the server where database resides.
- **Use default port** checkbox: Select to force the use of the default port for the selected database type. This will enable the **Database Instance Name** selection, when used with a Microsoft SQL Server.
- **Database Instance Name**: Enter an existing Microsoft SQL Server's instance name.
  This option only applies to existing Microsoft SQL Server instances, and not for MariaDB or MySQL.
  Note that the Microsoft SQL Server instance name must meet Microsoft's requirements.
- **Port**: Enter Port number. The initial selections are those which the vendors use by default.
- **Schema**: The individual database schema, within the vendor database. The default is **olconnect**.

  Note: If a previously non-existent schema were chosen here, then a new schema of that name will be created within the database when the back-end database swap is applied. The tables within that schema, though, will not be created until Connect is restarted.
- **Use Windows Authentication**: Select to use Windows authenticate for database login. This option is only available for Microsoft SQL Server back-end databases. Selecting this option disables the Username/Password fields.
- **Username**: Enter the database login username. This was initially set during the installation, and the default is **olconnect**.

  Tip: It is considered best practice for this user to have root privileges.
- **Password**: Enter the password associated with selected username.

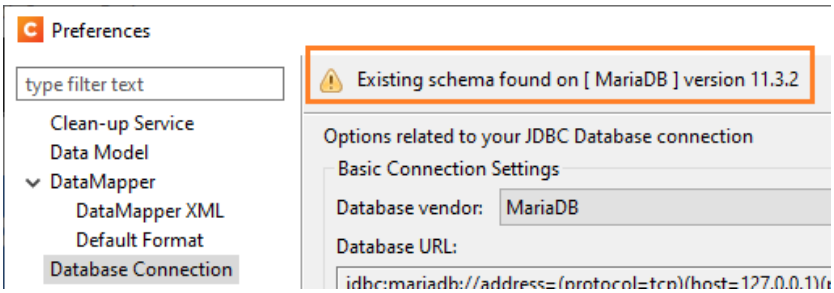
- **Advanced Connection Settings** selections:
  - **Maximum concurrent threads**: This option sets the maximum database threads. The maximum setting is determined by the specific capabilities of the machine Connect is installed upon (CPU speed and the amount of cores being the major determinants).

> **Tip:** Leaving this value set to the default maximum *should* be the best option in most circumstances.
> We recommended this entry be left at the default value.

- **Custom database parameters** table: These are extra parameters which are appended to the database connection URL. The default values are those which have been determined to be useful in connecting to specific vendor databases. For example, in order to use Integrated Authentication with Windows, specify the property `integratedAuthentication` with the value `true`, which adds `integratedAuthentication=true` to the URL.
    - **Property**: These are free field text fields.

      > **Note:** These fields and their associated values get appended to the JDBC connection and therefore must follow all rules regarding acceptable URL addresses for such.

    - **Value**: The value applied to the associated Property.
    - **Add**: Used to add extra Property values to the table.
    - **Delete**: Used to remove existing Property values from the table.
- **Test Connection**: Use to test if current connection settings will connect to the specified database.
  If successful, this will display the version of the database at the top of the Preferences page.



- **Restore Defaults**: Will restore the settings to OL Connect HyperSQL standard defaults.
- **Apply**: When a database connection is confirmed as correct this button becomes active, and is used to actually apply the database swap.

The Database Connection preferences also provides you with buttons to :

- **Test Connection**: This will run a test on the current Database Connection settings.
- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Default Locale

The Default Locale preferences are only available in the Designer Preferences. This setting determines the locale for new templates. By default this is the system's locale.
Select *System Locale* to use the operating system's locale settings.
Select *Explicit Locale* to choose a specific locale from the drop-down **Explicit Locale** list.
The Locale can also be changed on a per template basis. See "Locale" on page 921.

The Default Locale preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Editing preferences

These preferences define different editing options in the Designer module.

- **Object Resizing for elements**: This defines in which contexts to enable the resizing of elements (including Positioned and Inline boxes). Resizing elements may cause layouts to produce undesirable results especially when using Foundation templates.
    - **Enable for Print Context**: Check to enable resizing in the Print context.
    - **Enable for Web Context**: Check to enable resizing in the Web context.
    - **Enable for Email Context**: Check to enable resizing in the Email context.
- **Detail records preview limit**
    - **Maximum number of records to show in preview**: This setting limits the number of records to show in detail tables in both the Designer (Preview tab) and Datamapper. The lower this number, the shorter the time before the preview or the next record will appear.

- **Section clones preview limit**
  - **Maximum number of clones to show per section**: This setting limits the number of section clones to show in the Designer (Preview tab). Note that increasing this number can have a negative effect on the Designer's responsiveness.
- **Dragging data fields to the editor**. When you drag-and-drop a data field into a template, a placeholder appears in the template and a script is added (see "Inserting placeholders via drag-and-drop / double-click" on page 946). This option specifies what to use as the script's selector.
  - **New scripts use a class selector (recommended)**: The placeholder is wrapped in a Span element, and the Span's class is used as selector. This is recommended because it is faster (see: "Optimizing scripts" on page 1078).
  - **New scripts perform a search and replace**: The placeholder text is used as selector.

  Tip: Hold the **ALT** key while dragging to change this behavior on the fly.

  Note: If the option **Evaluate Handlebars expressions** is enabled for a section, dragging data fields to the editor will insert expressions instead of placeholders and scripts. See "Section properties dialogs" on page 1199.

CSS options

A CSS preprocessor is a CSS extension language that allows you to enhance CSS with code (variables, for example) and then compile it into plain CSS. CSS Preprocessor **Sass** is integrated in Connect.
For more information about Sass, see: https://sass-lang.com/.

The CSS options relate to the way Sass (.scss) files are compiled in Connect. See also: "Using a Sass file" on page 891 in the Online Help: https://help.uplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html.

- **Output style**: This setting determines the use of spaces and indentation in the compiled CSS file. For examples of the available output styles, see: https://sass-lang.com/documentation/file.SASS_REFERENCE.html#output_style.
  - **Compact**: Each CSS rule takes up only one line, with every property defined on that line. Nested rules are placed next to each other with no new line, while separate groups of rules have new lines between them.

- **Compressed**: This output style minifies the output. It has no whitespace except that necessary to separate selectors and a new line at the end of the file. It also includes some other minor compressions, such as choosing the smallest representation for colors. It's not meant to be human-readable.
- **Expanded**: This is the default output style. Each property and rule take up one line. Properties are indented within the rules, but the rules aren't indented in any special way.
- **Nested**: Each property has its own line, but the indentation isn't constant. Each rule is indented based on how deeply an element is nested in the HTML and CSS structure.

- **Auto compile on saving .scss files**: When this option is checked, a .scss file is compiled into a .css file whenever you save it, overwriting any previously compiled version of the .css file. Also, when a *partial* .scss file is saved all .scss files are recompiled. (A partial .scss file is meant to be imported in another .scss file. Its name starts with an underscore.)
  By default this option is enabled.

Note:

- Re-compiling a .scss file overwrites any manual changes made to the .css file.
- Partial .scss files cannot be compiled.
- Single line comments (//...) are not added to the compiled .css file, whereas multi-line comments (/* ... */) are maintained. Multi-line comments should be added within the brackets of the definition they apply to, to make sure they appear in the correct place in the compiled css.

Color options

Many of the colors in the user interface of Connect Designer can be adjusted. Click the small colored square next to the field that holds the default color value, to open the Color dialog and pick a color (see "Color Picker" on page 1143 in the Online Help: https://help.uplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html).

- **Edges**: The edges around elements in a section in the Workspace. Click the Show Edges icon to toggle the visibility of these edges.

    - **Box Objects**: This color highlights positioned boxes, inline boxes and Div elements; see "Boxes" on page 823.
    - **Table**: This color highlights tables, and the rows and columns in tables; see "Table" on page 860.
    - **Resizable Table**: This color highlights tables for which the option Allow resizing has been checked when adding the table; see "Table" on page 860.
    - **Forms**: This color highlights forms; see "Forms" on page 842.
    - **Shared Content**: This color highlights shared content, such as shared snippets; see "Snippets" on page 867.
- **Margin and guides**: These settings only apply to Print sections.
    - **Guides**: This is the color for rulers that can help position content correctly; see "Guides" on page 899.
    - **Margins**: This color delineates the content area on a page; see "Pages" on page 641.
    - **Bleed box**: This color delineates the printable area on a page; see "Page settings: size, margins and bleed" on page 642.
- **Master pages**: These edges are only visible on Master pages; see "Master Pages" on page 648.
    - **Header and Footer Margin**: This color highlights the header and footer margin set for the Master page; see "Adding a header and footer" on page 650.
    - **Objects**: This color highlights all elements on the Master page.
- **Script Result Highlighter**:
    - **Results**: Hovering over a script in the Scripts pane highlights content that will be affected by the script; see "Personalizing content" on page 923.

Images preferences

- **Transparent PDF image preview**: Check this option so that PDF resources added to the template (including in the Master Page and Media) display using transparency. Note that this can affect display performance (showing transparent PDFs is slower) but will not affect output speed.

The Editing preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.

- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Email preferences

Email (General) preferences

- **Default From Group**:
  - **Name**: Enter the name that is set by default in the "From name" field in the Send Email and Send Test Email dialogs ("Send (Test) Email" on page 1204).
  - **Email Address**: Enter the email that is set by default in the "From Email" field in the Send Email and Send Test Email dialogs ("Send (Test) Email" on page 1204).
- **Litmus account Group**:
  - **Email Test address**: If you have a Litmus account, enter the test address to use when sending a test email (see "Send (Test) Email" on page 1204 in the Online Help: https://help.uplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html). For more information on Litmus, please see http://litmus.com/.x

The Email (General) preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Email (SMTP) preferences

SMTP server presets can be selected when sending emails using either the Send Email or Send Test Email dialog. (See "Send (Test) Email" on page 1204 and "Email header settings" on page 672 in the Online Help: https://help.uplandsoftware.com/objectiflune/en/olconnect/2025.1/Home.html). For all presets, the password is not saved and must be re-entered when sending emails.

- The **Add**, **Edit** and **Delete** buttons let you create and manage the presets.
- **SMTP Host Settings**: These settings can be made or edited after clicking the Add or Edit button.

- **Name**: The name of the preset. This will show up in the Send Email dialog.
- **Host**: The SMTP server through which the emails are to be sent. Can be a host (mail.domain.com) or an IP address.

  Tip: Gmail only allows Connect to be used as an SMTP client when "Access for less secure apps" is enabled in the Google account settings.

- **Port**: The specified port number will be added to the host name, for example: smtp.mandrillapp.com:465.

  Note: If the mail server supports it, the connection will be encrypted without the need to send the server a STARTTLS instruction when port 465 is used.

- **Use authentication**: Check if a user name and password are needed to send emails through the host.
- **Send STARTTLS**: Enabled if authentication is checked. With STARTTLS the client negotiates with the mail server to use some form of encryption, usually a version of Transport Layer Security (TLS). Since this improves security it is recommend to enable this option if you use port 25 (the default port), 2525, or 587.
  Note that the email will not be sent if the SMTP server does not support TLS or SSL (an older encryption type).
  This option is ignored when port 465 is used.
- **Username**: Enter the user name used to connect to the SMTP server.

The Email (SMTP) preferences also provides you with buttons to :

- **Restore Defaults**: There are three default presets, each for working with a different Email Service Provider (ESP): Mandrilapp.com, Sendgrid and Mailgun (see "Using an ESP with OL Connect" on page 1678).
- **Apply**: Apply the new settings without closing the Preferences dialog.

Emmet preferences

Emmet is a framework that enables the lightning-fast creation of HTML code though the use of a simple and effective shortcut language resembling CSS Selectors (see Emmet Abbreviations: http://docs.emmet.io/abbreviations/syntax/). The Emmet functionality is available in the HTML and CSS source editors of Connect Designer. Emmet transforms abbreviations for HTML elements and CSS properties to the respective source code.

This is, for example, the abbreviation for a element with the class row:

```
div.row
```

On pressing the Tab key, this abbreviation is transformed to:

```
<div class="row"></div>
```

To learn more about Emmet itself, please see their website Emmet.io and their documentation: http://docs.emmet.io/. Emmet is a plugin. All options listed below are Emmet's default options. They are not specifically adjusted for Connect.

Common Emmet preferences

- **Expand abbreviations by Tab key**: Check to enable the Expand Abbreviation function.
- **... in files with extension**: Enter a comma-separated list of all file extensions in which expand abbreviation will work.
- **Upgrade web editors**: This option, inserted by the plugin that is used to integrate Emmet into Designer, does not affect how Emmet works in OL Connect.
- **Extensions Path**: Choose a folder where to put json and js files to extend Emmet. This includes custom snippets, preferences and syntax profiles. For more information see http://docs.emmet.io/customization/.

Emmet Abbreviation preferences

This Preferences tab lets you add and manage custom abbreviations. All standard abbreviations can be found in Emmet's documentation: http://docs.emmet.io/abbreviations/syntax/.

If there is no need to transform the text while expanding it, create an Emmet snippet instead (see below).

- **New**: Add a new abbreviation.
    - **Name**: The name of the abbreviation is also its trigger.
    - **Context**: The context in which the abbreviation is enabled (HTML, CSS, etc.).
    - **Description**: A short description of the abbreviation .
    - **Pattern**: This defines what an abbreviation expands to. Since Emmet is mostly used for writing HTML/XML tags, abbreviation definition uses XML format to describe elements; see http://docs.emmet.io/abbreviations/types/.
      Variables can be inserted via the **Insert Variable ...** button.

    - **Automatically insert**: This standard option doesn't affect how Emmet works in Connect Designer.
- **Edit**: Edit the currently selected abbreviation.
- **Remove**: Remove the currently selected abbreviation.
- **Import**: Click to open a browse dialog to import an XML file containing exported abbreviations. The imported abbreviations are added to the current list.
- **Export**: Click to open a Save as dialog to export all the abbreviations in an XML file that can be shared and re-imported.
- **Preview box**: Shows what the selected abbreviation is expanded to.
- **Restore Defaults**: clear all custom abbreviations.

  To temporarily disable an abbreviation, uncheck the checkbox next to the name of the abbreviation in the list.

Emmet Output preferences

The Output Preferences dialog is used to control how the expanded (output) code behaves when expanding abbreviations and snippets. There are 6 different dialogs to control output and, while they all have identical options, they control different output types: **CSS**, **HAML**, **HTML**, **XML**, **XSL** and the topmost default **Output**, which controlls all the other types.

These options are equivalent to Emmet's syntaxProfiles.json feature (http://docs.emmet.io/customization/syntax-profiles/).

Emmet Snippets preferences

Emmet Snippet are similar to abbreviations in that they are expanded when the Tab key is pressed, but they are just blocks of plain text. Anything in a snippet will be outputted “as is”, without any transformation.

- **New**: Click to create a new snippet.
    - **Name**: The name of the abbreviation is also its trigger.
    - **Context**: The context in which the snippet is enabled (HTML, CSS, etc.).
    - **Description**: A short description of the snippet.
    - **Pattern**: The pattern defines what a snippet expands to.
      Variables can be inserted via the **Insert Variable ...** button.

    - **Automatically insert**: This option doesn't affect how Emmet works in Connect Designer.
- **Edit**: Modify the currently selected snippet.
- **Remove**: Remove the currently selected snippet from the list.
- **Import**: Click to open a browse dialog to import an XML file containing exported snippets. The imported snippets are added to the current list.
- **Export**: Click to open a Save as dialog to export all the snippets in an XML file that can be shared and re-imported.
- **Preview box**: Shows what the selected snippet is expanded to.

  To temporarily disable a snippet, uncheck the checkbox next to the name of the snippet in the list.

Emmet Variables preferences

Variables are placeholders used in Emmet snippets to output predefined data. For example, the html:5 snippet of HTML syntax has the following definition:

```
<!doctype html>\n<html lang="${lang}">...</body>\n</html>
```

In the example above, ${lang} is used to refer lang variable defined in variables below. If your primary language is, for example, Russian, you can simply override lang variable with ru value and keep the original snippets. Also, you can override variable values with inline abbreviation attributes: html:5[lang=ru].

- **Name**: The name of the variable. This should be a single alphanumeric string with no spaces or special characters. For example, the myVar name is referred to as ${myVar}.
    - **Value**: The value of the variable when the snippet is expanded.
    - **New**: Click to create a new variable and define its name and value.
- **Edit**: Click to modify the currently selected Variable.
- **Remove**: Click to delete the currently selected Variable.

The Emmet preferences also provides you with buttons to :

- **Reload Engine**. This allows you to reload the Emmet engine based upon the current settings.
- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.

- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Engines preferences

See Engines Preferences.

General preferences

The General Preferences are as follows:

- **Always run in background**: This option correlates with the "Always run in background" option selectable in the "Document Boundaries Refresh" dialog and "Print via Print Server" dialog. When either of these dialogs is used and the option is checked, it will also be checked here. To prevent the refresh boundaries and print via print server dialogs to automatically run as background, uncheck this option.

The General preferences also provides you with buttons to :

- **Reset all Warning Dialogs**: This re-enables all warning dialogs that might have been previously disabled by selecting the "Don't show again" checkbox within the dialog.
- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Hardware for Digital Signing preferences

Connect can access secure hardware devices (USB Tokens, Smart Cards, and Hardware Security Modules) through PKCS#11, for use in creating PDF digital signatures.

This Preferences page allows for the configuration of these secure hardware devices. It caters for the two conflicting needs of Connect users.

1. Connect user wanting to create digital signatures using a secure device have to configure the DLL for using the secure device, so Connect can find the DLL when a digital signature is necessary.
2. The Connect administrator wanting the configuration of DLLs loaded by Connect Server, Designer, and their engines to happen outside of Presets, Templates, or other

resources. This so the administrator can control what DLLs are loaded instead of allowing user created configurations triggering third party DLLs at runtime.

This preferences page caters for both needs.

The **PKCS#11 Modules and Tokens** table contains a list of Hardware devices that can be used by Connect.
Hardware devices can be added, modified or deleted from the table using the following buttons:

- **Add**: Add a new hardware device to the list. This launches the Add PKCS#11 Module dialog.
- **Edit**: Edit that information relating to an existing hardware device. This launches the Edit PKCS#11 Module dialog.
- **Delete**: Remove an existing hardware device from the list.

The Hardware for Digital Signing preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Language preferences

- **Display language**: Select a language from the drop-down list to be used as the language of the User Interface (after the software is restarted).

The Language Settings preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Logging preferences

OL Connect logs the activities it undertakes whilst running. New Connect logs are created daily and are held for a period before they are automatically deleted.

The settings on this page determine the level of logging and how long the log files should be retained.

These log files can be an essential resource when diagnosing issues with OL Support. The logging settings are global to ***all*** Connect applications and the Logging Preferences can be adjusted from within any of the Designer, Print Manager and Server Configuration Preference dialogs.

The settings are:

- **Overall Logging Level** selection: Select the overall Connect Logging level. This controls how much logging Connect will do.
  By default the logging level is set at the midpoint of *Info*, but it can be set higher to include more logging (*All*, *Trace*, *Debug*), or lower to reduce the amount of logging (*Warning*, *Error*).
  The logging is hierarchical, with each Logging Level selection containing all of the Logging Levels below it. The default logging *Info* Level contains all *Error* and *Warning* entries plus informational log entries.

Caution: Higher logging settings will have an impact upon Connect production speeds, as well as leading to substantially larger log files.

We recommended leaving the logging level to *Info* and only using the higher levels of logging in conjunction with advice from OL support.

- **Rollover policy** selection: Chose whether to retain Connect log files for a certain number of days (**Daily logs**) or based upon some predetermined hard disk usage limitations (**Size-based logs**). The selection are as follows:
  - **Daily Logs**: Use this setting to determine how many days Connect Logs are to be kept, before they are deleted.
    - **Number of days to retain logs**: This value only impacts upon historic (closed) logs.
      Chose between 1 and 99,999 days.

The default value is set to 10 days for a new installation and 99,999 days on existing installations (to preserve backward compatibility).

- **Size-based logs**: Use this setting to restrict log file size, and to keep only a specified number of them.
  By combining the maximize individual log file size with the amount of log files to retain, this effectively allows a hard disk space usage limitation to be placed upon the logging process.
  - **Maximum size for log file**: This sets the maximum size a log file can reach before the logging system creates a new file.
  - **Number of files to keep**:This sets the maximum number of log files kept in the log folder. The default value is set to 50 for a new Connect installation and 99, 999 for an existing installation (to preserve backward compatibility).

- **Logging pattern** edit box: This edit box determines the formatting of the individual log entries.
  By default it is set to a date/time value which allows for simple searching within log files.

  Caution: We recommend leaving the Logging pattern to the default value.

  If you do need to change the Logging pattern, please see the Pattern Formatting guide (https://logback.qos.ch/manual/layouts.html#conversionWord) for help in doing so.

- **Log message preview** display: This displays a real time example of the format and content of individual log file entries, based upon the **Logging pattern** setting.

Note: The **language** of the log files is determined by the Language preferences.

Advanced settings

The Advanced settings over-ride the Overall logging settings, and provide a greater level of logging granularity.

Caution: Higher logging settings will have an impact upon Connect production speeds, as well as leading to substantially larger log files.

The Advanced Log Settings should only be set in conjunction with advice from

OL support, to ensure that only the most relevant settings are set to the higher logging levels.

This Preferences page allows you to add ( ) or remove ( ) individual Connect Packages, or change their logging settings ( ).

The Log Setting and Advanced Log Setting preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Parallel Processing preferences

See "Parallel Processing preferences" on page 116.

Print preferences

In order to print, the Designer connects to a Connect Server. To enable communication between the Designer and the Connect Server, settings have to be made under "Connect Servers preferences" on page 1061.

Available Printers preferences

The Available Printers preferences control which printer definitions are available when generating print output or creating Output Presets. Any printer that is unchecked in this dialog will not be visible in the "Model" drop-down of the Print Options dialog; see "Print options" on page 1380 and "Adding print output Models to the Print Wizard" on page 1652.

Available Printer Preferences:

- **Selected Printers**: Lists the available Printer Definition Files in the system. Note that these are not installed Windows printers or printer queues, but OL Connect Printer Definition Files.
- **Printer checkbox**: This checkbox selects/deselects all printers in the list. Click to check all, click again to uncheck all.

Print Measurements preferences

- **Units**: Use the drop-down to specify the default measurements system (inches, millimeters, or centimeters) used for dimensions of the template and boxes. In addition it defines the co-ordinates/position of box elements.
  The default unit will be added automatically when geometry values are entered without a unit in the Attributes pane or in the Box Properties dialog.
- **Flip insert guide axis**: Check this option to flip the axis on which guides are inserted. Normally, dragging a guide from a horizontal ruler inserts a horizontal guide (see "Guides" on page 899). With this option checked, dragging a guide from a horizontal ruler inserts a vertical guide.

The Print preferences also provides you with buttons to :

- **Test Print Server URL**. This button is only available for the General Print Preferences. It tests the Print Server URL settings made within that Preferences page.
- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Sample Projects preferences

Sample Project deployment settings

- **Workflow Configurations**
  - **Encoding**: By default, "Sample Projects" on page 1185 set the encoding of the Workflow configuration that they create to the encoding of the system on which the Designer runs (**system default**).

    If Workflow runs on another machine, and that system has a different active code page, Workflow may be unable to correctly interpret all the characters that are used in the configuration file (in path or file names, for example). To prevent this, you should select the appropriate encoding from the list.

Save preferences

The saving preferences are a way control if and how often OL Connect saves your work in the background, and if how many backup files it creates when you save the template or data mapping configuration. See also: "Saving a template" on page 590.

Auto Save

After a template or data mapping configuration has been saved for the first time, Connect Designer can auto save it with a regular interval.

- **Enable**: activate the Auto Save function.
- **Interval (minutes)**: enter a number of minutes, e.g. 3 to auto-save the template or data mapping configuration every 3 minutes.

Auto Backup

Connect Designer can automatically create a backup file when you **manually** save a template or data mapping configuration. The Auto Save function does **not** cause backup files to be created.

- **Enable**: activate the Auto Backup function.
- **Revisions to keep**: Enter the maximum number of backup files. When the maximum is reached, Auto Backup will overwrite the oldest file.
- **Destination**: Select the directory in which the backups should be stored.
    - **Original**: the directory in which the original file is stored.
    - **Other directory**: use the **Browse** button to select another directory.

Backup files have the same name as the original file with two underscores and a progressive number (without leading zeros) at the end: **originalname__1.OL-template**, **originalname__2.OL-template**, etc.

The Save preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Scripting preferences

The Scripting preferences define different options related to scripting within OL Connect. See also: "Testing scripts" on page 1074.

- **General**:
  - **Script timeout at design time (sec)**: In Preview mode or when running the Script Profiler (see the Profile Scripts dialog), a long running script is stopped after the amount of time set here. The default is 2 seconds, the minimum is 1 second.
  - **Expanded script quotes style**: When the Expand button in a Script Wizard is clicked, the expanded script will use either **double (")** or **single (')** quotation marks.

    Tip: Using single quotation marks in a script simplifies adding HTML fragments, which typically use double quotes.
- **Designer scripting profiling group**:
  - **Number of iterations**: Enter the number of times to run scripts when running the Profile Scripts dialog. The default is 1000. Accepted values are 1 to 1000000000. Yes, that's 1 billion - which would take a *long* time to run!

The Scripting preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Servers preferences

COTG Servers preferences

This option allows one or more Capture OnTheGo servers to be set up, in order to be used with the Capture OnTheGo WIM product.

All required information is summarized in the table, which has buttons to the right which allow one to add ( ), edit ( ) or delete ( ) individual entries. Double clicking on any entry in the table also launches the Edit dialog.

The options available in the Add/Edit dialogs are as follows:

- **Name**: Enter a unique name.
- **URL**: Enter a valid URL (including the protocol, e.g. http://) for the COTG server.

The COTG Servers preferences also provides you with buttons to :

- **Restore Defaults**: Removes all custom servers from the list and resets to the default Capture OnTheGo server.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Connect Servers preferences

The Connect Servers preferences allow to make settings for the default Connect Server and to set up connections to additional OL Connect Server modules.

The first Connect Server instance is the default instance. This instance is used when generating print output from the Designer and when opening a PS/PCL/AFP file in the DataMapper.
The default instance cannot be removed but it can be edited (except for its name). The Server module can be located on the same computer (hostname: localhost) or on a different machine. Multiple Designer modules can use a single Server module to generate print output, as long as the appropriate hostname, username and password are provided. In essence, this can be used to create a single Print Server.

Any additional Connect Servers may be selected in the Send to Server dialog (see "Sending files to Connect Server or to another server" on page 594).

All required information is summarized in the table, which has buttons to the right which allow one to add ( ), edit ( ) or delete ( ) individual entries. Double clicking on any entry in the table also launches the Edit dialog.

The options available in the Add/Edit dialogs are as follows:

- **Name**: Enter a unique name (case-sensitive).
- **URL**: Enter a valid URL (including the protocol, e.g. http://, and port).
- **Ignore certificate errors on https connections**: Select this option if you want certificate errors on HTTPS connections to be ignored, for example when testing a server that has a self-signed certificate.
- **Authentication**:
  - **User name**: Enter the user name to authenticate to the Connect Server.

    The Default user is "olc-user", but this could have been set to something else in the OL Connect installation.
  - **Password**: Enter the password to authenticate to the Connect Server.

  Authorized users are managed via the **Server Configuration Tool**. See "Security and Users Settings" on page 99.

The Connect Servers preferences also provides you with buttons to :

- **Restore Defaults**: Removes all custom servers from the list and resets to the default Connect Server.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Web Applications preferences

The Web Applications preferences allow to set up connections to additional servers. These servers may be selected in the Send to Server dialog (see "Sending files to Connect Server or to another server" on page 594).

All required information is summarized in the table, which has buttons to the right which allow one to add ( ), edit ( ) or delete ( ) individual entries. Double clicking on any entry in the table also launches the Edit dialog.

The options available in the Add/Edit dialogs are as follows:

- **Name**: Enter a unique name (case-sensitive).
- **URL**: Enter a valid URL (including the protocol, e.g. http://, and port).
- **Ignore certificate errors on https connections**: Select this option if you want certificate errors on HTTPS connections to be ignored, for example when testing a server that has a self-signed certificate.
- **Authentication**: Check to enable or disable the authentication options.
  If enabled, select the **type** of authentication that the server uses:
  - **Basic authentication**: The server authenticates a user by validating a password. Enter the **user name** and **password** to authenticate to the server.
  - **Bearer authentication**: Authentication is done by checking whether the bearer token is valid. Enter a valid **token**.

The Web Applications preferences also provides you with buttons to :

- **Restore Defaults**: Removes all custom servers from the list.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Versioning preferences

These preferences define the user information that will be used to identify changes in a versioned project and to access versioned projects in the cloud. See also "Versioned projects" on page 253 and "Versioned projects in the cloud" on page 258.

- **Author information**: The **author** and **email address** are required by Git. The name of the author appears in the history of a versioned project to show who made what change.
- **Remote repository**: OL Connect needs this information to access versioned projects in the cloud.
    - **Username**: The user name of an account with a Git repository hosting service like GitHub or BitBucket.
    - **Password**: The password (also called a *personal access token* or an *app password*) created for OL Connect by a Git repository hosting service. This password is to be used by OL Connect together with the user name to get access to remote repositories.
    - **Minutes between remote checks**: When a cloud-based versioned project is open, this setting determines how often OL Connect will check the remote repository for changes. If set to 0 (zero), OL Connect does not automatically check for remote changes .

Web preferences

Web Form preferences

These preferences define the default behavior of some form elements.

The preferences are as follows:

- **Insert Form Field Defaults**:
    - **Style**: Defines how labels are added to input form elements:
        - **Wrap input with label**: The label is wrapped around the element, such as <label>First Name <input type="text" name="first_name"></label>
        - **Attach label to input**: The label is placed before the input, and refers to it: <label for="first_name">First Name</label> <input type="text" name="first_name">

        - **Use label as placeholder**: The label is removed and the text is put as a placeholder, such as: <input type="text" name="first_name" placeholder="First Name">
        - **No label**: The label value is ignored.
    - **Insertion Point**: Defines how new elements are inserted, by default:
        - **At cursor position**: The element is inserted where the cursor is located in the template.
        - **Before element**: The element is inserted before the current element where the cursor is located. For example if the cursor is within a paragraph, insertion occurs before the <p> tag.
        - **After start tag**: The element is inserted within the current element, at the beginning, just after the start tag.
        - **Before end tag**: The element is inserted within the current element, at the end, just before the end tag.
        - **After element**: The element is inserted after the current element where the cursor is located. For example if the cursor is within a paragraph, insertion occurs after the <p> tag.
    - **Get Job Data File**: Defines the Workflow URL to be used when the **Get Job Data File on submit** toolbar button is active. This simplifies the process of creating and testing COTG Forms (see "Capture OnTheGo" on page 707).
        - **Workflow URL**: The default URL is: http://127.0.0.1:8080/_getSampleFormData_

The Web preferences also provides you with buttons to :

- **Restore Defaults**. This option restores the preferences to Defaults. This applies to the current Preferences page only, not any other Preferences.
- **Apply**: This option applies the settings made within the current Preferences page, but does not close the Preferences dialog.

Writing your own scripts

Personalization can be taken a lot further than just inserting names and addresses, and hiding or showing text or images. Every bit of information in your communications can be made entirely personal, using scripts.

In OL Connect, a script is a small set of instructions to the program, written in **JavaScript**. When OL Connect generates the actual output, it opens a record set and merges it with the

template. It takes each record, one by one, and runs all scripts for it (in a specific order, see "The script flow: when scripts run" on page 1082).

Any kind of personalization is done via scripts, but you don't have to write every script yourself. In many situations you can use a **Script Wizard**, which will do that for you (see "Personalizing content" on page 923).
For a block of variable data, such as an address, the **Text Script Wizard** is a perfect fit.
Paragraphs can be made conditional with a **Conditional Script Wizard**.
For dynamic images, you can use the **Dynamic Image Script Wizard**.

In an Email context, you are provided with a number of **Email Script Wizards** to set the sender, the recipients and the subject of the email.

However, when you want to do something that goes beyond what you can do with a Wizard, like creating a conditional paragraph with a condition that is based on a combination of data fields, you have to write the script yourself.

This topic explains how scripts work and how you can create and write a script yourself.

Script types

There are three types of scripts in the Designer: **Control Scripts**, **Standard Scripts** and **Post Pagination Scripts**.

Control Scripts

When output is generated from a template, Control Scripts run **before** all other scripts, when a record is merged with a context. They determine how different sections of the context are handled. They can, for example, make the page numbering continue over all Print sections, split Email attachments, or omit Print sections from the output.

Some functionality is provided as a Scripting Wizard, for example Conditional Print Sections.

Control Scripts don't touch the content of the sections themselves, but they can change the way a template is outputted. For more information about Control Scripts and their use, see "Control Scripts" on page 1098.

> **Tip:** Do you find yourself copy-pasting the same function into every new script? You can avoid this by defining your function in a Control Script. Control Scripts are executed first, so the function will then be available in all Standard Scripts and Post Pagination Scripts.

Handlebars Helpers

A Handlebars Helper - a function that can be used in expressions - is also defined in a Control Script.

Helpers can be made with the Handlebars Text Helper wizard (see "Using the Helper Wizard" on page 1003) or you can write them yourself (see "Creating custom Helpers" on page 1018).

Standard Scripts

Standard Scripts can change the contents of sections in a template. This type of script must have a **selector**. The selector can be text, an HTML element and/or a CSS selector (see "Selectors in OL Connect" on page 1083).

Running a Standard Script starts with looking for pieces of content in the template that match the script's selector.
The results of this query can vary from one occurrence of a simple text (for example: @EMAIL@) to a large collection of HTML elements. For example, when the selector is **p**, the HTML tag for a paragraph, all paragraphs will be collected and passed to the script.

> **Tip:** Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

Next, the script can modify the selected pieces of content, using values from the record that is merged with the template at the time the script runs. It can, for example, hide, replace or add text or change the style of those pieces of content. This is how scripts personalize documents.

> **Tip:** Content added by a script isn't visible in Design mode, but is visible and can be inspected in Preview mode.

> **Note:** In a Print context, the scripts in the Scripts pane run once for each section and then once for each Master Page (see Master Pages).

> **Note:** The context for user scripts does not reset across records. This means that if a script uses a variable that is not declared or initialized in that script, the variable retains its value from the previous record. This is by design. It allows you, for example, to maintain a running total across all records of a job.

Post Pagination Scripts

Post Pagination Scripts are run in a **Print** context **after** the content has been paginated. Because they can search through the output of all Print sections, and modify Print sections

(one at a time), they may be used to create a Table Of Contents (TOC), as explained in the topic: "Creating a Table Of Contents" on page 1116.

For more information see "Post Pagination Scripts" on page 1114.

Creating a new Standard Script

Writing a Standard Script starts with this procedure.

1. On the **Scripts** pane at the bottom left, click **New**. A new script appears in the list. Double-click on it to open it.
2. Change the name of the script, so that it reflects what the script does.

   > **Note:** Scripts can only have the same name when they are not in the same folder. (See "Managing scripts" on page 1071.)

3. Choose which kind of **selector** you want to use. Running a Standard Script starts with searching the template for pieces of content that match the script's selector. The collected pieces of content are passed on to the script (all at the same time, or one by one - see "Setting the scope of a script" on page 1071), so that the script can modify them.
   The selector can be:
   - **Text**, for example: @lastname@, or {sender}. The text doesn't have to have any special characters, but special characters do make it easier to recognize the text for yourself. In the Script Wizard, click **Text** and type the text to find.
   - A **selector** (HTML/CSS):
     - HTML elements of a certain **type**, such as a paragraph: <p>. In the Script Wizard, click **Selector** and type the HTML tag in the Selector field without the angle brackets: `p`.
     - HTML elements with a specific CSS **class** (eg. green). In the Script Wizard, click **Selector** and type the class name in the Selector field, preceded by a dot: .green.
     - An HTML element with a specific **ID** (eg. intro). In the Script Wizard, click **Selector** and type the ID in the Selector field, preceded by #: #intro.

       In an HTML file, each ID should be unique. This means that a particular ID can be used only once in each section.

    - Etcetera. See MDN Web Docs - CSS selectors for more selectors and combinations of selectors; also see "Selectors in OL Connect" on page 1083 for selectors that can only be used in OL Connect.
- A **selector and text**. This is text inside an HTML element (or several HTML elements) with a specific HTML tag, CSS class or ID. In the Script Wizard, click **Selector and Text**.

> **Tip:** When output speed matters, choose **selector** or **selector and text**. Searching text is a rather lengthy operation, compared to searching for HTML elements and/or CSS selectors. See also: "Optimizing scripts" on page 1078.

Quick-start a script for a specific ID or class

There is a shorter route to create a script for an element with a specific ID or class:

1. In the **Outline** pane at the left, click the element for which you want to create a script.
2. On the **Attributes** pane at the top right, type an ID or class.
3. On the **Attributes** pane, click the label to the left of the ID or Class input field (ID or Class) to make a new script with that ID or class as selector.

Drag-and-drop a data field to start a script

Here is another way to start a script with a class selector.

1. Right-click a section in the Resources pane and select Properties.
2. Uncheck the option to Evaluate Handlebars expressions and click OK.
3. Open that section and drag-and-drop a data field from the Data Model into the content.

What happens is that:

- A **placeholder** for the value of the data field shows up in the text. It looks as follows: @FIELDNAME@.
- A **script** appears in the **Scripts** pane at the bottom left.

The script replaces the placeholders in the content with the value of a data field in the current record.

As long as the option Evaluate Handlebars expressions is inactive, drag-and-drop will create a script and insert a placeholder that looks like this: @fieldname@. When the option is active, drag-and-drop inserts an expression like {{fieldname}}.

Writing a script

1. Create a new script (see: "Creating a new Standard Script" on page 1067, " Adding a Control Script" on page 1099 or "Adding a Post Pagination Script" on page 1115), or double-click an existing script in the **Scripts** pane on the bottom left.
   If the script was made with a Script Wizard, you have to click the **Expand** button before you can start writing code. This will change the Script Wizard into an **editor window**.

   > **Caution:** When you change an expanded text script and save it, it becomes impossible to edit the script using the Script Wizard again.

2. Write the script. Click **Apply** from time to time to see if the script works as expected. This will be visible on the **Preview** tab in the main workspace.

Syntax rules

Every script in the Designer must follow JavaScript syntax rules. For example, each statement should end with **;** and the keywords that can be used, such as **var** to declare a variable, are JavaScript keywords. There are countless tutorials available on the Internet to familiarize yourself with the JavaScript syntax.
For a simple script all that you need to know can be found on the following web pages: https://www.w3schools.com/js/js_syntax.asp and https://www.w3schools.com/js/js_if_else.asp. A complete JavaScript guide for beginners can be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

> **Tip:** In the editor window, press **Ctrl** + **Space** to see the available features and their descriptions.
> Use the arrow keys to select a function or object and press Enter to insert it in the script.
> Type a **dot** after the name of the function or object and press Ctrl + space again to see which features are subsequently available.
> For more keyboard shortcuts, see "Keyboard shortcuts" on page 1223.

Two basic code examples

Writing a script generally comes down to modifying the piece(s) of content collected from the template with the script's selector, using values, or depending on values of the record that is being merged to the template at the moment the script runs.

Modifying the template

To access and change the results of the query that is carried out with the selector (in other words: to modify the output), use the object **results**.

The following script (with the selector **p**) changes the text color of all paragraphs to red with a single line of code:

```
results.css('color', 'red')
```

It does this for each and every customer, because it does not depend on a value from the record that is being merged to the template.

Using values from the record in a script

To access the record that is being merged to the template when the script runs, use the object **record** (see "record" on page 1515).

Suppose you want to display negative amounts in red and positive amounts in green. Assuming that there is an AMOUNT field in your customer data, you could write the following script (with the selector: **td.amount**, that is: table cells with the class 'amount').

```
var amount = record.fields.AMOUNT;
if (amount >= 0)
    {results.css('color', 'green');}
else if (amount < 0) {
    results.css('color', 'red');
    }
```

When this script executes, it stores the value of the AMOUNT field from the current record in a variable and evaluates it. If the value is zero or higher, the color of text in the **results** - the table cells in this case - will be set to green; if the value is below zero, the text color will be set to red.

If an expanded script contains errors or if there are warnings, icons appear in the overview ruler on the right hand side of the editing area. These icons are shown relative to their position in the script and do not move as you scroll down. You can click on an icon to quickly jump to the error or warning. Script errors are highlighted by a red icon, and warnings in yellow. The topmost icon will display red if any errors exist in the script at all.

Designer API

Features like **results** and **record** do not exist in the native JavaScript library. These are additional JavaScript features, designed for use in Connect scripts only. All features designed for use in the Designer are listed in the Designer's API, with a lot of examples; see "Standard Script API" on page 1481.

Setting the scope of a script

The selector of a script can match multiple elements. By setting the scope of the script you can determine whether you want to run the script **once**, or **once for each element** that matches the selector. The second option is especially useful when a script targets rows or cells in a Dynamic Table (see "Using scripts in Dynamic Tables" on page 1095).

To set the scope of a script, click on **Options** at the bottom of the Script Editor window. The options are:

- **Result set**: The script will run once, regardless of the number of elements that match the selector. It will have access to all elements that match the selector via the `results` object (see "results" on page 1623). The `each()` function could be used to iterate over elements in the results.
- **Each matched element**: The script runs once for each element that matches the selector. The current element is accessible via the `this` object (see "this" on page 1554). If the script targets (something in) a row that is bound to a detail table, the current detail record is accessible via `this.record`.

Note: The scope of Control Scripts can't be set, because they don't have a selector.

Managing scripts

Changing the order of execution

When a record set is merged with a template to generate output, all scripts are executed once for every record in the record set, in the order in which they appear in the **Scripts** pane at the bottom left.

The order in which scripts are executed is particularly important when one script produces content that contains a selector for another script. If the other script has already been executed, it will not run again automatically. So, scripts that produce content that contains one or more selectors for other scripts need to come first.

To change the order in which scripts are executed:

- Click a script or a folder in the **Scripts** pane at the bottom. Drag it up or down and drop it.

Note: Control scripts are always executed first, regardless of where they are in the Scripts pane. They can not be excluded from execution for a specific context or section, using the execution scope of a folder; see "Execution scope" on the facing

page. What you can do is disable the script or the containing folder; see "Enable/disable scripts" on the next page.

Script folders

Scripts can be organized in folders. Why would you do that? For three reasons:

- Folders have an execution scope. You can specify for which contexts and sections the scripts in a folder have to run.
- Folders provide a better overview than a long unorganized list of scripts.
- Folders make it easier to change the order of execution for a bunch of scripts (see: "Changing the order of execution" on the previous page to learn why the order of execution is important). Dragging a folder up or down will cause all the scripts in that folder to be executed earlier or later, respectively.

To make a new folder on the **Scripts** pane:

1. In the **Scripts** pane, click the black triangle on the **New** button.
2. Click **Folder**. The folder will appear in the list of scripts.
3. Change the name of the new folder: right-click the folder and click **Rename**.
4. Drag scripts to the folder.

Tip: It may be helpful to put scripts that have an effect on the same context or section in one folder, because you can set the execution scope of scripts per folder (see: "Execution scope" below).

Note: Control scripts are always executed first, regardless of where they are in the Scripts pane. They can not be excluded from execution for a specific context or section, using the execution scope of a folder; see "Execution scope" below. What you can do is disable the script or the containing folder; see "Enable/disable scripts" on the next page.

Execution scope

A particular script may be used in one context or section, but not in other contexts or sections. Nevertheless, when processing the template, the Designer tries to find the selector of each script in all contexts and sections – unless the script is located in a scripts folder for which the execution scope has been set to the relevant contexts or sections. So, setting the execution scope of a folder saves processing time.

To change the execution scope of a script:

1. Put the script in a folder; see "Script folders" on the previous page.
2. Right-click the folder, and then click **Properties**.
3. Check the contexts and sections for which the scripts in this folder should run.

Note: Control scripts are always executed first, regardless of where they are in the Scripts pane. They can not be excluded from execution for a specific context or section, using the execution scope of a folder; see "Execution scope" on the previous page. What you can do is disable the script or the containing folder; see "Enable/disable scripts" below.

Tip: For more ways to optimize scripts, see "Optimizing scripts" on page 1078.

Enable/disable scripts

A disabled script will not run at all when the template is merged with a record set to generate output. Disabling script execution in certain contexts or sections helps with performance, since scripts normally run, whether or not their placeholder or selector is present in your template. It is highly recommended to disable any script that is not relevant to specific sections or contexts.

When you disable a folder, all scripts in the folder will be disabled.

To enable or disable a script or a folder:

- On the **Scripts** pane, right-click the script or the folder and click **Disable** (if the script or folder was enabled) or **Enable** (if the script or folder was disabled).

If all scripts in a folder are disabled, the folder itself will also be grayed out.

Tip: For more ways to optimize scripts, see "Optimizing scripts" on page 1078.

Import/export scripts

The easiest way to import scripts into another template is to open the template and use the **Import Resources** dialog; see "Import Resources dialog" on page 1155.
Remember also to import or add any files that a script refers to.

Alternatively, scripts can be exported or imported via the Scripts pane.
To export a script:

1. On the **Scripts** pane, click on a script, click the **View Menu** button and then click **Export**.

Or, right-click a script and select **Export**.

2. Give the script a name and click **OK**.

To import a script:

- On the **Scripts** pane, click the **View Menu** button and then click **Import**. Find the script and click **OK**

Files that a script may refer to, such as images, snippets and fonts, are not exported or imported together with a script.

Test the script to make sure that all files are present in the template and that the script's selector matches something in the content of the template; see "Testing scripts" below.

Testing scripts

The quickest way to test that scripts work as expected is to click the **Preview** tab at the bottom of the workspace.

You can even do this while creating a new script, either with a Script Wizard or in the expanded script editor. Click **Apply** at the bottom of the script editor to see the effect of the script on the **Preview** tab of the Designer.

In Preview mode you get warnings, such as when a table or field name is unknown, and errors, such as when unknown properties are referenced.

Note that scripts that use values of data fields can only be effective when a data file or data mapping configuration is open. See "Loading data" on page 926.

> **Note: Post Pagination Scripts** run only when a Print *section* is previewed or outputted. To verify the results of Post Pagination Scripts on a certain *Master Page*, preview the Print section to which that Master Page is applied.

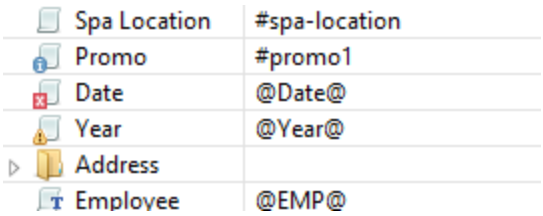
Testing for errors

Information in the Scripts pane

One way to see whether a script is functional is to take a look at the Scripts pane.

> **Tip:** Hover over the name of a script in the **Scripts** pane to highlight parts of the template that are affected by the script.

Icons on the name of scripts in the **Scripts** pane can show that there is a warning, information or error. If one of these icons appears, you can hover over it to see more details.



- The (i) information icon shows that the selector of the script does not produce a result in the current section. Note that standard scripts and post pagination scripts are not executed if their selector does not match any elements. A control script is always executed, assuming the script is enabled.
- The (!) warning icon appears, for example, when a script refers to an unknown field in the record set, or when ; is missing after a statement.
- The (x) error icon displays when the script results in an error, for example, when it uses an undeclared variable.

Information in the script editor

If an expanded script contains errors or if there are warnings, icons appear in the overview ruler on the right hand side of the editing area. These icons are shown relative to their position in the script and do not move as you scroll down. You can click on an icon to quickly jump to the error or warning. Script errors are highlighted by a red icon, and warnings in yellow. The topmost icon will display red if any errors exist in the script at all.

> **Tip:** To do a quick **search** for a certain script or piece of code, type a search text in the filter field located below the buttons in the Scripts pane.

Doing a Preflight

In addition to the icons and messages in the Scripts pane, a **preflight** can show if your scripts function as expected before generating output:

1. On the menu, select **Context > Preflight**.
2. Select **All**, or enter a selection of records. You can specify individual records separated by semi-colons (;) or ranges using dashes. For example: 2,4,6-10 would print pages 2, 4, 6, 7, 8, 9 and 10.
3. Click **OK**.

Preflight executes the template without actually producing output. When a data mapping configuration is used, any pre- and postprocessors are run as well.
The Preflight window displays any issues once it's done. It will tell, for example, which

selectors were not encountered in the template.
Double-click a script warning/error (either in the Preflight Progress dialog or in the Preflight Result view) to open the script in the script editor. The relevant line will be highlighted.

Tip: Be aware that scripts run in a specific order (see "The script flow: when scripts run" on page 1082). When one script unintentionally influences the results of another script, changing the order of the scripts in the Scripts pane may help (see "Changing the order of execution" on page 1071).

Note: An image with an unknown file extension is represented by a red cross in the output, but no error is logged unless the image refers to a local file that does not exist on disk.
Image file extensions that the software recognizes are: AFP, BMP, EPS, GIF, JPG/JPEG, PCL, PDF, PNG, PS, SVG, and TIF/TIFF.

Using the Script Debugger

The Designer's Script Debugger allows you to set breakpoints in scripts and step through them. It simulates an output run with only the current record and shows an overview of the state of all local and global variables, while the Workspace displays the partially merged document.

There are two ways to start the Script Debugger:

- Click the **Debug Scripts** button in the toolbar of the Scripts pane. The Script Debugger will pause and accept input as soon as it processes the first script.
- Right-click an enabled script in the Scripts panel and choose **Debug** from the contextual menu. The Script Debugger will pause and accept input as soon as it processes the selected script.

If the selected script is never processed, a message will pop up. This can happen when:

- The selector has no matches at any time. (This may depend on other scripts, since scripts can add or remove content.)
- The script is excluded through the properties of its parent folder.

For an explanation of the buttons in the Script Debugger, see "Script Debugger" on page 1182.

Testing for speed issues

To measure the time that the execution of scripts will take:

- On the **Context** menu, click **Profile scripts**.

Profiling means running the scripts in the template, with the current record, to see how fast scripts in the **Scripts** pane execute. It helps greatly in troubleshooting performance issues caused by scripts.

After running the Script Profiler you can see in which sections the script has run:

- Hover the mouse over a value in the column **Count** to see the number of times that the script has run, per section.

You can also see the breakdown of the execution time across different execution stages:

- Hover the mouse over a value in the column **Elapsed** to see the time elapsed (in milliseconds) since the start of the session. In the Scripts Profiler, the scripts are by default sorted based on the values in the **Elapsed** column, from high to low.
- Hover the mouse over a value the column **Delta** to see the difference between the time elapsed (in milliseconds) in the previous session and in the current session.

The script execution stages are:

Query: the time it takes to find the selector in the template.

> **Tip:** Looking for text is a rather lengthy operation. Use an ID or class (possibly in combination with a text) instead of a text selector to make the query faster. For more tips, see "Optimizing scripts" on the facing page.

Execution: the time it takes to execute the script. If you are an experienced JavaScript coder you may be able to optimize the code to speed up the execution of the script.

> **Tip:** Functions that actually change the content of the template (for example, **append ()**) are comparatively time consuming. Avoid using such functions in a loop. For more tips, see "Optimizing scripts" on the facing page.

Note that the times vary slightly per run of the Script Profiler. Run the Script Profiler a number of times and calculate an average from the results, before trying to speed up the execution of a script.

Script Profiler settings

Number of runs

By default, the Script Profiler runs on 1000 instances of all the scripts. To test on a higher or lower number of instances:

1. On the menu, select **Window > Preferences**.
2. Click **Scripting**.
3. Set a number of iterations (maximum one billion) and click OK.

Sorting

In the Scripts Profiler, the scripts are by default sorted based on the values in the **Elapsed** column, from high to low. Click any of the columns to sort the scripts according to the values in that column.

Script timeout

When testing scripts, either by toggling to Preview mode or by using the Script Profiler, a script timeout is active in the Designer, so that scripts that need a very long time to run are stopped after a set time. You can adapt this timeout to your needs, as follows:

1. On the menu, select **Window > Preferences.**
2. Click **Scripting**.
3. Set a timeout in seconds (for example: 2s) and click OK. The minimum timeout is 1 second.

Note: The script timeout is not active when generating output.

Optimizing scripts

In the process of output generation, the execution of scripts may take up more time than necessary. To optimize a template, it helps to disable scripts that don't have an effect on the output; see "Managing scripts" on page 1071.

This topic presents a number of other ways to speed up script execution by optimizing the scripts.

Use an ID or class as selector

Scripts (except Control Scripts) start with a query. The **selector** in the second column in the **Scripts** pane is what a script looks for in the template. The selector can be a text, HTML/CSS tags, or a combination of text and HTML/CSS tags.

Looking for text in a template is a less optimized operation and may impact output speeds in longer documents. To speed up the output process, it is recommended to use an ID or class as selector instead. This narrows the scope of the search and results in a very fast query, as elements with an ID or class are indexed by Connect Designer's layout engine.

For information about how to give an element an ID or class, see "ID and class" on page 764. See also: "Quick-start a script for a specific ID or class" on page 1068.

Targeting text

Text in itself cannot have an ID or class, but the element that contains it can. The smallest possible container of text is a Span. To learn how to put text inside a Span, see "Span" on page 826. Give the Span an ID or class and use that as the script's **selector**.

To target text in a bigger container (a paragraph, for example), change the Find method of the script to **Selector and Text**, use the ID or class of the container element as **Selector** and type the text in the **Text** field.

Tip: Dragging a data field directly to the **Scripts** pane creates a script with a *class* selector, without adding a placeholder to the template.

Tip: When you use the **drag-and-drop** method (without pressing the ALT key) to insert variable data placeholders into a template, the script's selector is by default the **class** of the element - a Span or Div - in which the placeholder is wrapped. For more information see: "Variable data in text: scripts and placeholders" on page 945.

Avoid DOM manipulations

The Scripting API of the Designer is a very powerful tool to manipulate and personalize your document. But keep in mind that DOM manipulation commands like append(), prepend(), before() and after() are resource intensive.

Try avoiding DOM modifications, especially within loops. Storing the content in a variable and appending the information after the loop is more efficient: this way, the template will be touched only once.

Example: The following example loads a snippet into a variable and uses the find() and text() commands of the Designer scripting API.

```
var labelElm = loadhtml('snippets/label.html');
for(var i = 0; i < record.tables.products.length; i++) {
    var label = labelElm.clone();
    label.find('@ProductLabel@').text(record.tables.products[i].ProductDescription);
    results.after(label);
}
```

What's wrong with this code is that it inserts the personalized information **within** the loop. The after() command runs as many times as there are records in the detail table 'products'.

Example: The script below is much more efficient: it adds the personalized content to a string called labelStr and only calls after() after the `for` loop.

```
var labelElm = loadhtml('snippets/label.html');
var labelStr = "";
for( var i = 0; i < record.tables.products.length); i++) {
    var label = labelElm.clone();
    label.find('@ProductLabel@').text(record.tables.products[i].ProductDescription);
    labelStr += label;
}
results.after(labelStr);
```

Use replace()

When personalizing HTML fragments retrieved from a snippet or from the template itself, JavaScript's `replace()` method shows the best performance.

`Replace()` can only be used on Strings, while the commands `loadhtml()` and `query()` return or a QueryResult, which is a set of strings, like the `results` object.

A QueryResult allows you to perform DOM manipulations like adding and removing elements, adding and removing CSS classes etc. When the required manipulations are limited to find/replace actions, you could change the QueryResult into a string. This allows you to replace text using the `replace()` method.

For this, you could use `toString()`:

```
var labelSnippet = loadhtml('snippets/label.html').toString();
```

Or you could copy the HTML of the QueryResults to a variable:

```
var block = results.html();var labelSnippet = loadhtml('snippets/label.html').toString();
var labelStr = "";
for( var i = 0; i < record.tables.detail.length; i++) {
    var label = labelSnippet;
    label = label.replace('#', i);
    label = label.replace('@product@', record.tables.detail[i].fields['product']);
    label = label.replace('@notes@', record.tables.detail[i].fields['notes']);
    label = label.replace('@netweight@', record.tables.detail[i].fields['netweight']);
    labelStr += label;
}
results.after(labelStr);
```

Tip: The `replace()` method as used in the above example replaces only the first occurrence of the search string. To **replace every occurrence** of a search string in a given string, use a **regular expression**. In the following line of code, the regular expression `/@product@/g` makes replace() search for all occurrences of the string @product@ in the `label` string:

```
label = label.replace(/@product@/g, record.tables.detail[i].fields
['product']);
```

In this example, @product@ is a pattern (to be used in a search) and g is a modifier (to find all matches rather than stopping after the first match). For more information about possible regular expressions, see MDN Web Docs - Regular expressions.

Replace several placeholders in one script

Suppose there are 20 different placeholders in a postcard (for the address, account and customer details, a promo code, the due date, discounts, a link to a personalized landing page etc.). Typically this would require 20 queries. Even after optimizing these scripts by using an ID as selector for those scripts, there are still 20 scripts, 20 queries to run.

If there was only one query, one single script to do all the work, the output could be generated much faster. Reducing the number of scripts improves the performance of the template. How to do this?

First, wrap the content that contains all of the placeholders in one (inline) Box and give that Box or Span an ID (on the Attributes pane). Next, create a script that uses that ID as selector. Then replace all placeholders in the script and put the content back in the template.

This is similar to working with snippets, but in this case the element is extracted from the actual template.

Example: The following script replaces all of the placeholders on a postcard. It takes advantage of the JavaScript replace() command. Assuming that the ID of the block that requires personalization is `promoblock`, the script has to have its selector set to `#promoblock`.

```
var block = results.html();
var data = record.fields;
block = block.replace('@name@',data.first + ' ' + data.last);
block = block.replace('@address@',data.address);
block = block.replace('@zip@',data.zip);
block = block.replace('@city@',data.city);
block = block.replace('@country@',data.country);
block = block.replace('@saldo@',data.saldo);
block = block.replace('@promo@',data.promo);
block = block.replace('@customercode@', data.customercode);
…
results.html(block);
```

The first line retrieves the HTML of the promo block and stores it in a variable called block. To make the code more readible, the fields from the record are stored in a variable named `data`. After replacing the placeholders by values, the script replaces the HTML of the promoblock with the personalized string.

Other resources

There are also many resources online to help learn about JavaScript performance and coding mistakes. See for example:

- JavaScript performance
- The 10 most common JavaScript mistakes

Note that most resources on the web are about JavaScript in the *browser*, but the greatest majority of the tips do, indeed, apply to scripts in general, wherever they are used.

The script flow: when scripts run

When Connect generates the actual output – letters, web pages or emails -, it opens a record set and merges it with the template. It takes each record, one by one, and runs all scripts for it, in a specific order, as explained below.

First all Control Scripts are executed, in the order in which they appear in the Scripts pane. Control scripts don't touch the content of the sections themselves, but they change the way a template is outputted, for example by selecting or omitting sections from the output (see "Control Scripts" on page 1098).

Secondly, any **Handlebars expressions are evaluated** in sections where the option *Evaluate Handlebars expressions* is enabled. (On the Resources pane, right-click the section and select Properties to find this setting.)

Then the Standard Scripts are executed, once for each section, in the order in which they appear in the Scripts pane.
Standard Scripts can change the contents of the current section in a template.
This type of script must have a **selector**: text, an HTML element and/or a CSS selector (see "Writing your own scripts" on page 1064 and "Selectors in OL Connect" on the next page).
Running a template script starts with looking in the current section for pieces of content that match the script's selector.
Important to note is that **if nothing matches the selector, the script is not executed.**

In a **Print context**, the Standard Scripts in the Scripts pane run once for each section and then for each Master Page (see "Master Pages" on page 648). Next, each processed Master Page is put behind every page to which it should be applied.
Scripts are NOT executed again for every page.

> **Note: Translations** are applied before the Standard Scripts run, and also every time a Standard Script adds content to a section, for example with the `results.html()` function.

Finally, Post Pagination Scripts run, in the order in which they appear in the Scripts pane (see "Post Pagination Scripts" on page 1114).
Post Pagination Scripts are run in a **Print** context **after** the content has been paginated. Because they can search through the output of all Print sections, and modify Print sections (one at a time), they may be used to create a Table Of Contents (TOC), as explained in the topic: "Creating a Table Of Contents" on page 1116.

Tip: Do you find yourself copy-pasting the same function into every new script? You can avoid this by defining your function in a Control Script. Control Scripts are executed first, so the function will then be available in all Standard Scripts and Post Pagination Scripts.

Note: Any JavaScript files included in a section run **after** the scripts in the Scripts pane.

Selectors in OL Connect

Selectors are patterns used to select one or more HTML elements. They were originally developed to be able to define the layout of web pages without touching their content, through Cascading Style Sheets (CSS). In OL Connect, since each section in a template is in fact an HTML file (see "Editing HTML" on page 762), the very same selectors can be used in style sheets (see "Styling templates with CSS files" on page 881) and Standard Scripts (see "Personalizing content" on page 923 and "Writing your own scripts" on page 1064). Using selectors for scripting can increase the speed with which a template and data are merged; see "Use an ID or class as selector" on page 1078.

Standard CSS selectors

Selectors are made up of one or more of the following components:

- An **HTML element**. Type the HTML tag without the angle brackets (e.g. `p`) to select all elements of that type (`p`selects all paragraphs).
- A **class**. Type the class name, preceded by a dot, e.g.: `.green`, to select HTML elements with that class.
- An **ID**. Type the ID, preceded by #, e.g.: `#intro`, to select an HTML element with that ID.

  Note: In an HTML file, each ID should be unique. This means that a particular ID can be used only once in each section.

- An **attribute** of an HTML element. Type the attribute and, optionally, its value, between square brackets, e.g.: `[target]`, to select HTML elements with a matching attribute.
- A **pseudo-class**. For example, `tr:nth-child(even)` selects all even table rows.

These components can be combined in different ways. For example, `p div` selects all paragraphs *inside* elements, while `p, div` selects all paragraphs *and* all elements.

A complete list of selectors and ways to combine them can be found here: MDN Web Docs - CSS Selectors.

A **video** about CSS and Script Selectors, can be found here: Connect with Evie 6 - CSS and Script Selectors.

OL Connect classes and attributes

OL Connect itself sometimes adds a specific class or attribute to elements in a template. Capture OnTheGo widgets, for example, have a role attribute that allows the COTG library to dictate their behaviour. OL Connect-specific classes and attributes can be used in selectors, as will be explained and demonstrated below.

OL Connect-specific classes usually are invisible in the Designer. By opening the currently selected section in your default web browser (click the Preview HTML toolbar button) and using the browser's code or source inspector you can see most of the dynamically added classes.

Caution: Avoid using classes with the **__ol** prefix in your selectors. These dynamically added class names may change in future releases of the software.

Section selector

The Designer writes the name of each section to the `section` attribute of the <html> element. This attribute can be used in selectors.

Note: To make scripts run exclusively on certain sections, it is advised to put them in folders and set the execution scope of the scripts in a folder via the folder properties; see "Execution scope" on page 1072.

Example: The following rule applies formatting to <h1> elements in sections of which the name starts with 'Letter':

```
[section^='Letter'] h1 { color: brown; }
```

Note:

To target sections as well as Master Pages, use the `body` selector without the `masterpage` or `section` selector. For example:

- Selector: `body`
- Script: `results.html('<div style="background:red;width:1in;height:2in">Hello World</div>');,`

In versions prior to 2019.2 the `section` selector would work on sections as well as Master Pages. As of version 2019.2, this is no longer the case. The `section` selector now only selects sections.

Master Page selector

The Designer writes the name of each Master Page to the `masterpage` attribute of the <html> element. This attribute can be used in selectors.

Example: This script adds a box to the `body` of every Master Page.

- Selector: `[masterpage] body`
- Script: `results.html('<div style="background:red;width:1in;height:2in">Hello World</div>');`

The following script adds the box only to the Master Page called "Master Page 1".

1. Selector: `[masterpage="Master page 1"] body`
2. Script: `results.html('<div style="background:red;width:1in;height:2in">Hello World</div>');`

Sheet position selectors

In Print output, pages have a sheet position that depends on the options set in the "Sheet Configuration dialog" on page 1208 (e.g. the Duplex and Allow Content On options). Connect gives each page - or rather, the "MediaBox" div element on that page - a class depending on their sheet position:

- .frontside
- .backside (does not apply to simplex documents)
- .contentpage
- .nocontentpage

The MediaBox contains the Master Page objects and section backgrounds. This means that these classes can only be used to format a Master Page and section background. They do not let you change the formatting of elements residing in the main text flow (e.g. a <h1> element on page 3).

Formatting Master Page objects depending on the sheet position

The following CSS rule sets the color of <h1> elements on a Master Page when that Master Page is present on the front of a sheet.

```
.frontside h1 {
        color: green;
}
```

The next style rule is a bit more specific: it colors <h1> elements on a Master Page when that Master Page is applied to the front of a sheet in Section 1:

```
[section='Section 1'] .frontside h1 {
        color: green;
}
```

The following rule hides <h1> elements on the back of a sheet on which no content (from the main text) is allowed.

```
.backside.nocontentpage .h1 {
        display: none;
}
```

Print section background selector

When you inspect a Print section in a browser, you will see that it has a element as the first child of the <body> element. Inside this there are one or more MediaBoxes: elements with the class page_mediabox. Each MediaBox contains the Media, section background and Master Page that apply to one page (see "Media" on page 652, "Master Pages" on page 648 and "Using a PDF file or other image as background" on page 634).

In the MediaBox, a Print section background is an <img> element with the ol_pdf_datamapper_input class. Its `src` attribute references the PDF file that contains the image and the `page` parameter is used to select a specific page in that PDF (as a PDF can contain more than one page).
For example:
<img src="file:/C:/Users/MyUser/Pictures/mixed.pdf?page=1" class="ol_pdf_datamapper_input">.
You can use the ol_pdf_datamapper_input class as a selector to target the section background in a style rule or script.

Placing the section background in front of the Master Page

The stacking order of elements inside each MediaBox, from bottom to top, is:

1. Media
2. Section background
3. Master Page elements

Using the .page_mediabox selector, you could change this stacking order and place the section background on top of the elements on the Master Page. Set the z-index property to a value larger than 0 (zero) and add !important to make this style rule override the inline style declaration that normally puts the section background behind the Master Page elements:

```
.page_mediabox img.ol_pdf_datamapper_input {
    z-index: 10 !important;
}
```

Scaling the section background

The rule below downscales the section background image and keeps it in the centre of the page:

```
.page_mediabox img.ol_pdf_datamapper_input {
    transform: translate(-50%, -50%) scale(1.5, 1.5) !important;
}
```

View selectors

In the Designer, sections can be viewed on different tabs: Source, Design, Preview and - if it is a Web section - Live. In each view mode (except Source) a specific CSS class is added to the <html> element. The view-specific classes are:

- .DESIGN
- .PREVIEW
- .OUTPUT

.OUTPUT is used when viewing the current section on the Live tab or in an external browser, and when generating output.

View selectors allow you to apply formatting to elements in a certain view, for example to highlight or show elements. The Designer itself does this, for example, to highlight all boxes in the Design view, when the Show Edges icon is clicked.

Adding an outline

The following style rule wraps every element that has the class address-block with a purple dashed outline in Design mode. The outline is not visible in other views or when outputting the document.

```
.DESIGN .address-block {
    outline: 1px dashed purple;
}
```

Adding a background pattern

The Postcard template wizard (in the Basic Print templates group) uses the .DESIGN class to mark areas that are reserved for postal use and should not contain text or images. These areas were added to the Master Page as absolute positioned boxes that have been given the class clearzone. The following style rule assigns a background pattern to elements with that class in the Design view:

```
.DESIGN .clearzone {
    background:url(...SuQmCC) repeat;
}
```

Note: The pattern image was created on www.patternify.com and is added as a data URI (see Data URIs).

Showing hidden Foundation elements

In Capture OnTheGo templates based on the Foundation framework the .DESIGN selector can be used to show elements that would otherwise be hidden in the Design view.
For example, to expand accordion elements and show validation errors in Design view, you could add the following style rules to your template:

```
.DESIGN .accordion .accordion-navigation > .content {
        display: block;
}
.DESIGN small.error {
        display: block;
        margin-top: -20px;
}
```

Writing scripts: tips and tricks

This topic provides some tips and tricks to use when writing template scripts.

For the basics, see "Writing your own scripts" on page 1064.

All features designed for use in the Designer are listed in the Designer's API, with a lot of examples; see "Standard Script API" on page 1481.

Tip: Scripts can be exported from one template and imported into another. See "Import Resources dialog" on page 1155.

Looping over elements

If the selector of a script matches more than one element in a template, you may want the script to loop over them.

One way to do this is by setting the script's scope set to **Each matched element** (see "Setting the scope of a script" on page 1071). The script will be called in a loop over the elements that match the selector of the script. It can access each matched element directly via the `this` object. For an example, see: "this" on page 1554.

Otherwise, there are a number of iterator functions that you can use, such as `each()`, `for(..in...)` and `for...of`.

Function	Description
"each()" on page 1482	A generic iterator function, to iterate over the elements in the result set.
"for(... in ...)" on page 1485	Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.
for...of	The for...of statement executes a loop that operates on a sequence of values sourced from an iterable object. See Mozilla's documentation. **Note:** A for...of loop currently cannot be used in combination with 'const'.

Retaining values between scripts

The context for user scripts does not reset across records. This means that if a script uses a variable that is not declared or initialized in that script, the variable retains its value from the previous record. This is by design. It allows you, for example, to maintain a running total across all records of a job.

> **Example:** x = 123;

Example: numbering optional elements in Connect template

Let's say there are a number of different, optional elements in the template. Whether the optional elements appear in the template depends on standard scripts. The optional elements that become visible should be numbered in sequence (1, 2, ...).

One solution is to assign a class, e.g. "opt", to all the optional elements in the template. In a Control Script, set a variable, e.g. "SequenceNumber", to 0:

```
sequenceNumber = 0;
```

Then create a script that has the `opt` class as selector. Set the scope to **Each matched element**. If an element is visible, the script increments the sequence number and puts it before the element.

```
if (!(this.css('display') == 'none')) {
    sequenceNumber++;
    this.prepend(sequenceNumber + ' ');
}
```

Obviously, this script should run after any optional elements have been made visible.

Note that this solution only works if the optional elements appear in the main flow of the template. It doesn't work for positioned elements.

Defining a function in a Control Script

Do you find yourself copy-pasting the same function into every new script? You can avoid this by defining your own function using a **function expression**:

```
fn = function() { … }
```

The function `fn` can be accessed from all subsequent scripts.

> **Tip:** Define the function in a Control Script. Control Scripts are executed first, so the function will then be available in all Standard Scripts and Post Pagination Scripts.

Tip: Store Control Scripts remotely to make them available for use in multiple templates. See "Using a remote Control Script" on page 1100.

Loading a snippet via a script

It is possible, and often very useful, to load a snippet dynamically.

Create a script (see "Writing your own scripts" on page 1064). In the code use the following function:

- For an HTML snippet: `loadhtml('snippets/nameofthesnippet.html')`.
  To insert the snippet in the content at any position where the script's selector is encountered, write `results.loadhtml('snippets/nameofthesnippet.html')`.
  Make sure that the file name is exactly the same as the file in the **Snippets** folder. If the file name isn't correct, the snippet will not appear in the template.
- For a JSON snippet: `loadjson('snippets/nameofthesnippet.json')`.
- For a Handlebars snippet: see "Handlebars in OL Connect" on page 998.

Tip: To insert the code to load a snippet even quicker, you can:

- **Drag a snippet into the Script window**. The function that loads the script - `loadhtml()` or `loadjson()`, depending on the file type - will automatically be added, including the file name.
- **Right-click a snippet** and select **Copy Resource Location** to copy the relative path of the snippet to the clipboard. It may then be pasted into a script.

Remote snippets are retrieved in the same way, except that the file extension should be `.rhtml` instead of `.html`. If it is a remote JSON snippet, the file extension is `.rjson`. Again, note that the name of the snippet must be exactly the same as in the Snippets folder.

For more examples, see "loadhtml()" on page 1493 and "loadjson()" on page 1497.

Tip: The techniques described in this topic are demonstrated in a how-to: Using a selector to load part of a snippet.

Loading part of an HTML snippet

When a snippet contains a part that can be identified by a selector, that selector can be used to load that part of the snippet into a template.

In script, use the following code:

```
results.loadhtml('snippets/nameofthesnippet.html', 'selector')
```

See "loadhtml()" on page 1493 for more information about this function.

Loading a snippet, depending on the value of a data field

To load a snippet depending on the value of a data field, you have to add a condition to the script.

Example: The following script evaluates if the value of the LANGUAGE field in the record is 'En'. If so, the snippet is added to the content.

```
if (record.fields.LANGUAGE == 'En') {
results.loadhtml('snippets/nameofthesnippet.html');
}
```

Loading part of a snippet, based on the value of a data field

When a snippet contains a part that can be identified by a selector, that selector can be used to load that part of the snippet into a template. It is possible to do this, based on the value of the data field. This is easiest when the selector matches the value of a data field.

Example: The following script reads the value of the LANGUAGE field in the record and uses that value as the selector in the function loadhtml(). If the snippet contains an HTML element with this ID (for example, <p ID="En">), that HTML element will be added to the content:

```
var language = record.fields.LANGUAGE;
results.loadhtml('snippets/nameofthesnippet.html', '#'+ language)
```

See also: "Standard Script API" on page 1481.

Tip: An easy way to group content in an HTML snippet is putting each part in a container and giving that container an ID.
For example: `<div ID="EN"><p>This is text for English customers.</p></div>`

Use the function `.children()` to load the contents of the container, and not the

container itself.
For example: `results.loadhtml('Snippets/myfooter.html', '#EN').children()`

This script loads the paragraph of the example (<p>), but not the container itself ( ).

Load a snippet and insert variable data into it

The following script loads part of an HTML snippet based on the value of a field, and then finds/replaces text with the value of a field before inserting the content into the document.

```
var promoTxt = loadhtml('snippets/promo-en.html', '#' + record.fields['YOGA']);
promoTxt.find('@first@').text(record.fields['FIRSTNAME']);
results.html(promoTxt);
```

Loading content using a server's API

Content in a template is usually static (apart from being personalized) and part of the main text flow. It can also be located in a snippet (see "Snippets" on page 867).
It is also possible to include content that is served by another server. Many servers provide an API to fetch publicly available content from their site. That content may even be dynamic: the **most recent** blog posts on a Wordpress website, for example, or the **current** weather forecast for a certain city.
This topic explains how to retrieve content using a server's API and insert that content in a template.

Step 1: Getting the appropriate link

To request content from another server, you will need a link.
Some websites give the option to **embed** their content in your website by providing a link or the complete HTML. Youtube.com, for example, offers not only a link to share a certain video, but also the full HTML to embed that video in your website.
If that option is not available, you will have to build the link yourself. Find the server's **API** and look through it to get the exact endpoint and parameters that you need.

With many servers it is required to use an **API key** in the link; this key generally comes for free after you sign up to their website. The key will be part of the link that is used to make a request to the server.

Note: Pay attention to the service's Terms of Service. Many servers have limitations on the number of calls that can be made to them for free. Beyond these limits, their content will not show up in your template unless you purchase a business plan.

Step 2: Preparing the template

The next step is to set up a template. If you've got the HTML to embed content in your template you can paste that HTML on the Source tab (and skip Step 3).
Otherwise your template has to contain an element that can be replaced or followed by the remote content: an empty paragraph, for example, or a heading. If the element isn't unique in the template, give the element an ID.

Note that interactive content, such as an interactive map, can only be used in Web templates, and cannot be output on Print or Email contexts (even though they will show up in Preview mode!).

Step 3: Writing a script

The final step is to write a script that retrieves the content and inserts it into the template (see "Writing your own scripts" on page 1064). Use the element or the ID of the element that you added in Step 2 as the script's selector. For information about selectors, see "Selectors in OL Connect" on page 1083.

> **Tip:** Select an element, then click on 'ID' in the Attributes pane, to create a script that has that element's ID as selector.

Retrieving content

Depending on the type of content that the remote server returns - HTML or JSON - you can use `loadhtml(location)` or `loadjson(location)` (see also: "loadhtml()" on page 1493 and "loadjson()" on page 1497) to retrieve the content. The link that you selected in Step 1 should be passed to the function as a string. For example:

```
loadjson('https://blog.mozilla.org/wp-json/wp/v2/posts?per_page=5');
```

If the returned content is JSON data, that data has to be wrapped in HTML before inserting it into the template. This is demonstrated in the example below.

> **Tip:** Install the Postman application to preview JSON returned by an endpoint.

> **Tip:** To load a JavaScript file (.js) or a style sheet (.css) you can use `loadtext()`. See "loadtext()" on page 1499.

Inserting content in the template

To insert the content after the selected element, use `results.after()`. To replace the element with the new content, use `results.html()` or `results.replaceWith()`.

Example: recent posts

The following script loads five posts from Mozilla's blog and inserts their titles as links in a template. Mozilla's blog is a WordPress website. Since the WordPress REST API uses JSON as the response format, the `loadjson()` function has to be used and the received content has to be wrapped in HTML.
If the script's selector was `h1` (a level one heading), the retrieved content would be inserted after each level one heading.

```
var postsObj = loadjson('https://blog.mozilla.org/wp-json/wp/v2/posts?per_page=5');
var html = '';
html = '<ul>';
for (var idx in postsObj) {
    html += '<li><a href="' + postsObj[idx].link + '">' + postsObj[idx].title.rendered + '</a></li>';
}
html += '</ul>';
results.after(html);
```

See WordPress REST API developer endpoint reference.

Using scripts in Dynamic Tables

If the contents or style of cells or rows in a "Dynamic Table" on page 974 need to be personalized based on the value of a data field, you will need a script.

Do you want to style or change the output of an **expression**, based on the value of a data field? Then a custom Helper can come in handy. A Helper is a function that can be used in an expression. How to create a custom Helper is described in the topic: "Creating custom Helpers" on page 1018. After creating a Helper you can type that Helper's name in any expression.

In all other cases you will need a Standard Script.

This topic gives some information that will help you to write a script for (an element in) a Dynamic Table.
It is assumed that you are familiar with the scripting basics; see "Writing your own scripts" on page 1064.

Writing scripts for a Dynamic Table

If you're going to create your own scripts for Dynamic Tables, there are a few things you need to know.

Selectors

Scripts that target (an element in) a Dynamic Table can use the same types of selectors as other scripts. In the output, Dynamic Table rows are repeated including any **classes** that are

set on the row and on its contents.

In addition, Dynamic Tables and their rows and cells have some special **data- attributes** that can be used as selector. See: "A Dynamic Table's data- attributes" on page 988.
Note however, that the `data-repeat` attribute cannot be used as a selector for Standard scripts, since it gets removed when the table is expanded, which happens before Standard scripts run.
The `data-script` attribute is used if you let the Designer create a script for you (see "Quick-start a script with the Create script button" below).

How the scope of a script can simplify code

It is recommended to set the scope of a script that targets (an element in) dynamically added rows to **Each matched element**.

The selector of a script that targets (something in) a row that is linked to a detail table will probably match multiple elements.
By setting the **scope** of a script you can determine whether you want to run the script **once**, or **once for each element** that matches the selector. (See "Setting the scope of a script" on page 1071.)

If a script runs **once**, accessing the detail data and dynamically added rows tends to be a bit complicated, especially when they are nested, since it involves looping over the results and record objects (see "results" on page 1623 and "record" on page 1515).

If a script targets (something in) a row that is bound to a detail table, and runs **once for each matched element**, the current element is accessible via the this object (see "this" on page 1554) and the current detail record is accessible via this.record.

The advantage of the latter approach is demonstrated in an example below.

Quick-start a script with the Create script button

This quick-start procedure only works for elements (a **cell** or **span**) that have a direct link with a data field.

To find out whether a cell or span has a direct link to a data field, select it (see "Selecting an element" on page 766) and:

- Take a look on the **Attributes** pane. If under **Other**, the **Field** drop-down or **Sum** field point to a data field in the detail table, the element has a direct link to a data field.
- Switch to the Source view. Elements with a direct link to a data field have a `data-field` or `data-sum` attribute.

By default, cells that are filled with a placeholder have a direct link with a data field. Cells that are filled with an expression don't. To create a direct link with a data field, select the cell and choose the data field from the **Field** drop-down on the **Attributes** pane.

The easiest way to add a script that targets a specific element (a **cell** or **span**) that has a direct link with a data field is this.

1. In **Design** mode, select the element.
2. Click the **Create Script** button next to the **Field** drop-down on the **Attributes** pane.
3. Click **Yes** to confirm that the direct link with the data field will be broken.

As a result:

- The element will now have a `data-script` attribute.
- The `data-field` or `data-format` attribute is removed.
- A new script will be created and opened in the Script Editor.

The **selector** of the new script is the `data-script` attribute with the name of the data field that the element was linked to as its value. For example: table [data-script='ID'].

The **code** of the script replaces the contents of the element with the current value of the data field:

```
var field = this.record.fields['ID'];
if (field) {
this.text(formatter.upperCase(field));
}
```

Under **Options**, the **scope** of the script is set to **Each matched element** (see "Setting the scope of a script" on page 1071). This means that in the code, this refers to the element that matches the selector, and this.record refers to the **current (nested) detail record** (see "this" on page 1554).

Tip: To get access to the row in which the cell is located, you can use this.parent().

Note: If you bind the element to a data field again by selecting it and then selecting a field from the Field drop-down on the Attributes pane, the data-script attribute will be removed.

Example: Styling a row based on a value in a detail table

Here are two scripts that have the same effect: if in a row, the value of the field Shipped is 1, it colors the text of that row green.

Number	Description	Unit Price	Quantity
53674	Thule Crossroad Railing Foot Pack M450	199.95	3
62516	CamelBak Arete 18 Hydration Pack 2016 Black	65.00	1
117653	Swix Carving Kit 1	44.99	1
148080	Swix Alpine Racing Straps Ski Strap	9.99	4

The first script targets a **cell** in a Dynamic Table row that has a [data-script='Shipped'] attribute. The script has its scope set to "Each matched element", so it can use this (see "this" on page 1554) to access the selected cell and this.record to access the current detail record, without performing any loops.

Selector: table [data-script='Shipped'].
Scope: Each matched element

```
var field = this.record.fields['Shipped'];
if (field == 1)
    this.parent().css('color', 'green');
```

The following script targets the **table**, and has its scope set to "Result set". This script loops over the detail table in the record, evaluating the field Shipped. If the value of that field is 1, it looks up the corresponding row in the results (the selected Dynamic Table).

Selector: #table_1 tbody
Scope: Result set

```
for(var i = 0; i < record.tables.detail.length; i++){
    if(record.tables.detail[i].fields['Shipped'] == 1)
    query("tr:nth-child(" + (i+1) + ")", results).css('color','green');
}
```

Control Scripts

When output is generated from a template, Control Scripts run **before** all other scripts, when a record is merged with a context. They determine how different sections of the context are handled. They can, for example, make the page numbering continue over all Print sections, split Email attachments, or omit Print sections from the output.

Some functionality is provided as a Scripting Wizard, for example Conditional Print Sections.

This topic explains how to add a Control Script and it gives an overview of what Control Scripts can do. It will also tell you where you will find information about each feature, including examples.

The basics of script-writing in the Designer are explained in the following topic: "Writing your own scripts" on page 1064.

What Control Scripts are

Control Scripts are a special kind of Designer script. They can manipulate the way output is generated from a template. They allow you, for example, to change the page numbering in Print output, to split one generated Print document into multiple Email attachments, or to set a Print section's background dynamically. (These are only a few examples; for more uses of Control Scripts see "What to use a Control Script for" on the facing page.)

Control Scripts differ from Standard scripts in two ways:

- Control Scripts run **before** all other scripts. When a template consists of several contexts, and these contexts are combined in the output - for example, when an Email is generated with the Print context as attachment - all scripts run once for each context, but Control Scripts always go first.
- Control Scripts **do not touch the content** - meaning, the text flow - of the sections. They don't have a selector, like the other scripts do. A selector selects parts of the content of a section and stores them in the `results` object, so that they can be modified in the script. As Control Scripts don't have a selector, the `results` object can't be used there.
  Similarly, the `query()` function, which is used to select content from within a script, is unavailable in a Control Script.

Adding a Control Script

To add a Control Script:

1. On the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Control Script**. A new script appears in the list.
2. Double-click the new script to open it. The script editor appears.
3. Change the name of the script so that it reflects what the script does.

   Note: Scripts can only have the same name when they are not in the same folder.
4. Write the script; see the "Control Script API" on page 1591. If you are not familiar with scripting, also see "Writing your own scripts" on page 1064.

   Tip: New Control Scripts added to the template contain code to continue the page numbering over all print sections, and two examples: one to select different

sections of a Print context for email and print output, and one to select a Web section.

Using a remote Control Script

A remote Control Script is not located within your template but is located in a network folder or hosted on an external web server.

To add a remote Control Script:

1. Right-click the **Control** folder on the **Scripts** pane, and click **New > Remote Control Script**.
2. Enter a name for the file as it should appear in the Control Scripts folder. For better management, it's best to use the same filename as the remote resource.
3. Enter the **URL** for the remote script, which must be a JavaScript (.js) file. If the file is located on an external web server, this must be a full URL, including the http:// or https:// prefix, domain name, path and file name.
   If the file is located on your network you can click the Browse button or enter the path and file name manually.

   The complete syntax with the "file" protocol is: `file://<host>/<path>`. If the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: file:///c:/resources/scripts/myControlScript.js.
   If the file is located on another server in your network, the path must contain five slashes after "file:".

   Note: Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file://///myserver/myfolder/myControlScript.js).

Tip: To refresh the remote resources in a Designer view, use the Refresh option in the menu (**View > Refresh**) or the Refresh button at the top of the Workspace.

What to use a Control Script for

Control Scripts let you change the way a template is merged, by giving access to the template with all its contexts and sections in a script. A Control Script may, for example, omit, group and clone sections; add a background to a Print section; or add a header to an email. A number of the things that you can do with them is listed in the table below, with a link to a topic that explains how to do it and that shows what the script should look like.

In a Control Script, `section` usually is the most important object. To get a quick overview and lots of examples, see "section" on page 1627. For help on specific tasks, see the table below.

Task	See topic	Field/function of `section` object
Change the **page numbering** of Print sections	"Control Script: Page numbering" on the facing page	`restartPageNumbering`
Set the **background image** of a Print section	"Control Script: Setting a Print section's background" on page 1106	`background.source`, `background.url`, `background.position`
Split and rename Print email **attachments**	"Parts: splitting and renaming email attachments" on page 1104	`part`
Dynamically set a **password** on PDF attachments	"Control Script: Securing PDF attachments" on page 1112	`password`, `ownerPassword`
Include/exclude sections: • Conditionally omit Print, Web or Email sections • Output one section or another, based on the value of a data field • Select one Print section as PDF attachment if the output is to be emailed, and another Print section if the output is to be printed.	Use a Conditional Print Section script to in-/exclude a Print section based on a simple condition; see "section" on page 1627 In all other cases take a look at the examples in the following topic: "section" on page 1627.	`enabled`
Copy and add sections dynamically (cloning sections)	Create a Clone Section script with the wizard: "Cloning sections" on page 965. Or, create a Clone Section script manually: "Clone Section scripts" on page 1108.	`clone()`
Add a **header** to an email	"section" on page 1627, examples: "Adding custom ESP handling instructions" on page 1679.	`headers`
Set the **margins** of a Print section or Master Page	"margins" on page 1633	`margins`
Set a **Master Page**, **Media**, or **Duplex** printing for a Print section	"sheetConfig" on page 1609	`sheetConfig`

Task	See topic	Field/function of `section` object
Repeat the sheet configuration after a number of pages	"sheetConfig" on page 1609	`sheetConfig.positions.repeat`
Dynamically change the **page size** of print sections and media	"section" on page 1627, "media" on page 1597 **Note:** When a Control Script changes the size of a section, it should also change the size of the linked Media; this is not done automatically. While the output may still look good, a size mismatch can cause an issue if a script or another component assumes that the media size matches the section size. In case of a size mismatch a preflight will show a warning (see "Doing a Preflight" on page 1075).	`height`, `width`

Control Script: Page numbering

This topic explains how to write a Control Script that changes the page numbering in Print sections. Note that when you add a Control Script, it already contains a script to make the page numbering continue over all Print sections.

For information about Control Scripts in general, see "Control Scripts" on page 1098 and "Control Script API" on page 1591. If you don't know how to write scripts, see "Writing your own scripts" on page 1064.

How to change page numbering in a control script

A Control Script can make the page numbering continue over all Print sections or let it restart on a section. This is done by setting the `restartPageNumber` field on a `section` to `true` or `false`.
For example: `merge.template.contexts.PRINT.sections['Section 2'].restartPageNumber = true;`. (Also see "section" on page 1627 and "Control Script API" on page 1591.)

Page numbering starts with page 1 for each section. If for a section `restartPageNumber` is set to `false`, that section will start with the page number following the last page of the previous section.

Note that even if a section is not enabled (so it will not be outputted), its `restartPageNumber` flag is still taken into account for composing the page number sequences.

By default, each section has `restartPageNumber = false` when the first control script runs.

Tip: If you are looking to create a short, simple table of contents in **one section**, you could add a Standard Script that uses the `pageRef()` function. For an example, see "pageref()" on page 1580.
For a **multi-page**, **cross-section** table of contents you must use a Post Pagination Script; see "Creating a Table Of Contents" on page 1116.

Examples

Restarting the page numbers several times

Assume that a template has four sections (of 1 page each) in the Print context and a Control Script sets the page numbering as follows:

1. Section A (1 page) restartPageNumber = true
2. Section B (1 page) restartPageNumber = true
3. Section C (1 page) restartPageNumber = false
4. Section D (1 page) restartPageNumber = true

The code would look like this:

```
if (merge.context.type == ContextType.PRINT) {
    merge.context.sections['Section A'].restartPageNumber = true;
    merge.context.sections['Section B'].restartPageNumber = true;
    merge.context.sections['Section C'].restartPageNumber = false;
    merge.context.sections['Section D'].restartPageNumber = true;
}
```

The page numbering in the output will be:

1. Section A page 1
2. Section B page 1
3. Section C page 2
4. Section D page 1

Disabled section

When a section is disabled, it will not be outputted, but its `restartPageNumber` flag will still be taken into account for composing the page number sequences. So, if the `restartPageNumber` flags are set as follows:

1. Section A (1 page) restartPageNumber = true
2. Section B (2 pages) restartPageNumber = false

3. Section C (3 pages) restartPageNumber = true, enabled = false
4. Section D (4 pages) restartPageNumber = false

In code:

```
if (merge.context.type == ContextType.PRINT) {
    merge.context.sections['Section A'].restartPageNumber = true;
    merge.context.sections['Section B'].restartPageNumber = false;
    merge.context.sections['Section C'].restartPageNumber = true;
    merge.context.sections['Section C'].enabled = false;
    merge.context.sections['Section D'].restartPageNumber = false;
}
```

The page numbering in the output will be:

1. Section A page 1
2. Section B page 2
3. Section D page 1 (page numbering is restarted due to section C's restartPageNumber = true)

Parts: splitting and renaming email attachments

In a Control Script, **parts** can be defined to determine which sections should be output to the same file. This way it is possible to split the Print context or the Web context into multiple email attachments. This topic shows how to do that.

For information about Control Scripts in general, see "Control Scripts" on page 1098 and "Control Script API" on page 1591. If you don't know how to write scripts, see "Writing your own scripts" on page 1064.

Defining parts

Defining parts is done by setting the `part` field on a `section`, for example:

```
merge.template.contexts.PRINT.sections['Section 2'].part = "PDF_Attachment2";.
```

(Also see "section" on page 1627 and "Control Script API" on page 1591.)

- If a part name is given, then that delimits the start of a new part (even if the part name is the same as the previous one). Following sections that don't define a part name, will be added to the previous part.
- A part ends at the last enabled* section or at the last section before the start of a new part.

*When a Control Script has set the `enabled` field of a `section` to `false`, it will not be outputted.

If no part name is set on any section, it is assumed that there is only one part, consisting of the default section (for Web output) or of all sections (for Print output). The attachment(s) will be named after the email subject.

Examples

No parts defined

Assume there are three Print sections: sections A, B and C. When generating Email output with the Print context as attachment, all three Print sections will be put together in one file and attached to the email. If the email's subject is 'Take action', the name of the attached file will be 'Take action.PDF'.

Splitting and renaming a Print attachment

Assume there are three Print sections: sections A, B and C. In a Control Script a part name is defined for section C:

```
var section = merge.template.contexts.PRINT.sections['Section C'];
section.part = 'Part2';
```

When generating Email output with the Print context as attachment, the email will have two attachments:

- attachment 1: Section A, Section B
- attachment 2: "Part2", which is Section C. The file name of this attachment is the part name.

Note: For Web sections, a part always consists of only the given section. Web pages cannot be appended to form a single part. It is however possible to attach multiple Web pages to one email; see the following example.

Controlling multiple Email attachments

The following script attaches the following sections to an email:

- Print section 3 + 4 as attachment with continued page numbers
- Print section 6 as separate attachment
- Web sections A and B as separate attachments

```
if (channel == Channel.EMAIL) { // only when generating Email output
if (merge.context.type == ContextType.PRINT) {
        merge.context.sections['Section 1'].enabled = false;
        merge.context.sections['Section 2'].enabled = false;
        merge.context.sections['Section 3'].enabled = true;
        merge.context.sections['Section 3'].part = "PDFAttach1";
        merge.context.sections['Section 4'].enabled = true;
        merge.context.sections['Section 4'].restartPageNumber = false;
        merge.context.sections['Section 5'].enabled = false;
        merge.context.sections['Section 6'].enabled = true;
        merge.context.sections['Section 6'].part = "PDFAttach2";
} else if (merge.context.type == ContextType.WEB) {
       // disable whatever is the default section
        merge.context.sections['default Section'].enabled = false;
        merge.context.sections['Section A'].enabled = true;
        merge.context.sections['Section A'].part = "WebPartA";
        merge.context.sections['Section B'].enabled = true;
        merge.context.sections['Section B'].part = "WebPartB";
}
}
```

Note: For another example, see this how-to: Output sections conditionally.

Control Script: Setting a Print section's background

In the Print context, an image file can be used as a Print section's background; see "Using a PDF file or other image as background" on page 634.

If you want the section background to be switched automatically, depending on the value of a data field, you need a Control Script. There is a Script Wizard that can generate that script for you, provided that certain conditions are met; see: "Dynamic Print section backgrounds" on page 994.
Otherwise, you will have to write the Control Script yourself. This topic explains how to write a Control Script that sets a Print section's background.
Note that the settings made in a Control Script take precedence over the settings made in the Print Section Properties dialog.

Note: Encrypted PDF files are **not supported** in *PDF pass-through* mode.

Note: Any non-content features in the original PDF will get stripped out of the original PDF. These include document navigation (outline), local links, hyperlinks, annotations (markup annotations, comments, etc.), digital signatures, multimedia features, interactive forms, and tagged PDF features (marks in the page content will stay, but the logical structure is gone).

For information about Control Scripts in general, see "Control Scripts" on page 1098 and "Control Script API" on page 1591. If you don't know how to write scripts, see "Writing your own scripts" on page 1064.

Setting a background in script

The Control Script should first enable a background on the section, in case an initial background wasn't set via the user interface. This is done by setting the source type for the background of the section to either DataMapper PDF or Resource PDF (see "BackgroundResource" on page 1615). For example:

```
merge.template.contexts.PRINT.sections['Policy'].background.source =
BackgroundResource.RESOURCE_PDF;
```

merge.template.contexts.PRINT.sections['Policy'].background.source = BackgroundResource.RESOURCE_PDF;

A DataMapper PDF is, as you would expect, a PDF generated by the DataMapper. A Resource PDF is a PDF from another source.

For a DataMapper PDF, nothing else has to be done to set the background. For a Resource PDF, the Control Script should specify a path, for example:

```
var resourceUrl = 'images/policy-' + record.fields.policy + '.pdf';
merge.template.contexts.PRINT.sections['Policy'].background.url = resourceUrl;
```

Note: An image with an unknown file extension is represented by a red cross in the output, but no error is logged unless the image refers to a local file that does not exist on disk.
Image file extensions that the software recognizes are: AFP, BMP, EPS, GIF, JPG/JPEG, PCL, PDF, PNG, PS, SVG, and TIF/TIFF.

Positioning, scaling and rotating the background

After a background has been selected, it can be positioned, scaled and rotated, using properties of the background object; see "background" on page 1607.

To position the background, for example, set the section's background.position:

```
activeSection.background.position = MediaPosition.FIT_TO_MEDIA;
```

For all possible positions, see "MediaPosition" on page 1618.

Setting a page range in script

When a PDF that serves as a dynamic section background has multiple pages, you can specify a range of pages to be used, in a control script.
Put the number of the first page in the range in the section's background.start field and the last page in background.end.

The following script sets the page range from 2 to 5:

```
merge.template.contexts.PRINT.sections['Policy'].background.start = 2;
merge.template.contexts.PRINT.sections['Policy'].background.end = 5;
```

Setting a page range automatically sets background.allPages to false (see "background" on page 1607).
On the other hand, when you first define a page range and then set background.allPages to true, this disables the page range.

> **Tip:** You could use the `resource()` function to check the number of pages or for example the page height and width before setting it as a background (see "resource()" on page 1599).

This script sets a background on a Print section using absolute positioning.

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.url = "images/somepage.pdf";
activeSection.background.position = MediaPosition.ABSOLUTE;
activeSection.background.left = "10mm";
activeSection.background.top = "10mm";
```

You could replace the last three lines of the previous script by the following line to scale the Print section background to Media size:

```
activeSection.background.position = MediaPosition.FIT_TO_MEDIA;
```

Clone Section scripts

This topic describes the basic Clone Section script and explains how to enhance it.

Rather than starting from scratch, we recommend using the script generated by the Section Clone wizard as a starting point. (See: "Cloning sections" on page 965.) The same script is described in this topic. If the script created by the wizard doesn't do everything you need, it's still a good place to start.

For information about Control Scripts in general, see "Control Scripts" on page 1098 and "Control Script API" on page 1591. If you don't know how to write scripts, see "Writing your own scripts" on page 1064.

The Clone Section script

Here is an example of a Clone Section script created by the wizard.

```
var section = merge.template.contexts.PRINT.sections["Section 1"];
if (!section) {
    fatalError("Section 'Section 1' not found");
}

var table = record.tables["products"];
if (table) {
    for (let detail of table) {
        let clone = section.clone();
        clone.meta["detail-table"] = "products";
        clone.meta["detail-table-record"] = detail.index;
        clone.meta["detail-record-as-scope"] = true;
        section.addAfter(clone);
    }
    section.enabled = false;
}
```

First, the script retrieves a section - in this case: Section 1 - from the "merge" on page 1635 object and stores it in a variable called 'section'.

If the section does not exist, an error is thrown. See: "fatalError(message)" on page 1484.

Then the script retrieves a detail table from the "record" on page 1515, by name. In this example the name of the detail table is **products**.

If the detail table exists, the script loops over the records in the detail table with `for ( .. of .. )`, creating a clone for each of them using the "clone()" on page 1611 function, and filling the `meta` property of the clone with some data (see "meta entries in section clones" on the facing page).
Then it adds the clone to the Print context using the `addAfter()` function.
At the end of the loop it disables the original section.

> **Note:** Cloned sections have the same properties as normal sections (see "section" on page 1627), but they cannot call the section functions `clone()`, `addBefore()` and `addAfter()`, which means you cannot clone a clone, or use a clone as a starting point to insert other clones into a template.

`meta` entries in section clones

The Clone Section wizard inserts the following fields into the `meta` property of each clone:

- **detail-table** - Name of the detail table
- **detail-table-record** - The index of the clone's record in the detail table (zero based)
- **detail-record-as-scope** - The data scope for Handlebars expressions: the detail table (true) or main record (false)

`meta` entries are accessible in Standard Scripts as well as in Handlebars expressions.

Writing and reading entries in the meta property

If many **scripts** need to have access to the same detail table field, it may be worth the effort to add that data field to the `meta` property within the Clone Section script.
Like other `meta` entries, these custom entries can be read in a Standard Script.

For example, assume the detail table that clones are based on has a data field called 'title'. With one additional line of code in the standard Clone Section script, this field can be written to the meta property of each clone, in a new entry 'title'.

Example: clone.meta["title"] = detail.title;

This line should go inside the loop in which clones are created.

The new meta entry 'title' is now available in Standard Scripts. For example, the following script, with a selector of `h1`, replaces the text of h1 headings with the current value of the 'title' detail table field.

Example: let theTitle = merge.section.meta["title"]
results.text(theTitle)

Since `meta` entries can have any type, it would even be possible to write an entire detail table record into the `meta` property.

Tip: In section clones, detail table fields are most easily accessed through a Handlebars expression. See "Handlebars expressions in section clones" on page 969.

Assigning a background PDF to a section clone

Section clones automatically get the same background image as the original section. If each section clone needs a different background image, this is easily set in the section clone wizard.

However, the Clone Section wizard expects the path to a background PDF to be specified in the detail table record the clone is based on. If there is no such field in the data, you may click the **Expand** button in the wizard and modify the script to construct the URL in code.

The following script constructs the path to a PDF for each clone using the contents of a data field.

```
var section = merge.template.contexts.PRINT.sections["Policy"];
if (!section) {
    fatalError("Section 'Invoice' not found");
}

var table = record.tables["policies"];
if (table) {
    for (let detail of table) {
        let resourceUrl = 'images/policy-' + detail.fields["Name"]; + '.pdf';
        let clone = section.clone();
        clone.meta["detail-table"] = "policies";
        clone.meta["detail-table-record"] = detail.index;
        clone.meta["detail-record-as-scope"] = true;
        clone.background.source = BackgroundResource.RESOURCE_IMAGE;
        clone.background.url = resourceUrl;
        clone.background.position = MediaPosition.CENTERED;
        section.addAfter(clone);
    }
    section.enabled = false;
}
```

For information about the properties used to set a section's background, see "Control Script: Setting a Print section's background" on page 1106.

Renaming a clone

By default, clones have the same name as their source section with an index in round brackets.

> **Example:** Source: "Section 1"
> Clone names: "Section 1 (1)", "Section 1 (2)", "Section 1 (3)", ...

Use the `name` property to assign another name to the cloned section, for example:

```
clone.name = "my_section_clone";
```

Just like section names, the clone's name should be unique - within the scope of a single record, that is; across records, the same name can be used.

When two clones get the same name in the same record, one of the clones can no longer be used. This is prevented by giving clones unique names. Unique names also ensure that CSS rules and scripts can target a specific clone.

Control Script: Securing PDF attachments

The Print context can be attached to an email in the form of a PDF file and secured with a password. This can be done without a Control Script, see "Email attachments" on page 679 and "Email PDF password" on page 678.
With a Control Script, you can do the same, and more: the attachment can be split into multiple attachments (see Parts). Each attachment may have a different (or no) set of passwords, so you could mix secured and unsecured attachments in a single email. This topic shows how.

For information about Control Scripts in general, see "Control Scripts" on page 1098 and "Control Script API" on page 1591. If you don't know how to write scripts, see "Writing your own scripts" on page 1064.

Setting passwords in script

To set a password on a Print section in a Control Script, the script should first retrieve the Print section/s using `merge.template.contexts.PRINT.sections` or `merge.context.sections` (also see the example below).
Next, the script can split the attachments, if needed (see "Parts: splitting and renaming email attachments" on page 1104), and it can set a password on each `section`. For example:

- `merge.template.contexts.PRINT.sections['Section 2'].password = 'secret';`
- `merge.template.contexts.PRINT.sections['Section 2'].ownerPassword = 'secret';`

When producing a **single** attachment, the password(s) should be set on the first Print section.
When producing **multiple** attachments, it should be set on the first section of each part.

Password types

PDF allows for two types of passwords to be set on a secured PDF file: a user password and owner password. The user password allows a limited access to the file (e.g. printing or copying text from the PDF is not allowed). The owner password allows normal access to the file. The Email PDF password script sets both the user and owner password to the same value, so that when the recipient provides the password, he can manipulate the file without limitations.

In a Control Script:

- `password` is used to set the user password and owner password for a PDF attachment to the same value.
- `ownerPassword` is used to set the owner password for a PDF attachment. Setting only the owner password creates a secured PDF that can be freely viewed, but cannot be manipulated unless the owner password is provided. Note that the recipient needs Adobe Acrobat to do this, because the Acrobat Reader does not allow users to enter the owner password.

Removing a password

Passwords set in the Control Script override the password set through the Email PDF password script (see "Email PDF password" on page 678). This allows you to change or remove the password from a specific part. Removal is done by setting the `password` field to `null` or an empty string ("").

This script splits the Print output into two PDF attachments and sets a password for the second attachment.

```
var printSections;
if (channel == Channel.EMAIL) { // only when generating Email output
if (merge.context.type == ContextType.PRINT) {
printSections = merge.template.contexts.PRINT.sections;
    printSections['Section 1'].part = 'PDFAttach1';
    printSections['Section 2'].part = 'PDFAttach2'
    printSections['Section 2'].password = 'secret';
    }
}
```

Writing an email attachment script

Instead of editing an Email section's properties, or using the Dynamic Attachment Script Wizard, you could write your own scripts to add attachments to an email. Here's how to do that.

1. On the **Scripts** pane at the bottom left, click **New**. The Edit Script dialog will start. Click OK a new script appears in the list. Double-click on it to open it.
2. Change the name of the script, so that it reflects what the script does.
3. The **selector** of the Attachments script should be the **head** of an Email section, e.g.: `html.HTML_EMAIL[section="myEmailSection"] > head`.
4.
   - For resources **inside** the template, specify the file name, including the respective folder in the Resources folder, for example `'images/file.extension'`, or `'fonts/myfont.otf'`, etc.

- For **external** resources, you need the full path to the file, for example: `'file:///c:/resources/attachments/instructions.pdf'`
  or, for a **remote** file:
  `'http://localhost:8080/pod/v1/deliverynotes/{8FCEC8BC-72E8-486B-A206-516BF10E21F6}'`.

Use the `append()` function to add a <link> element to the head of the Email section and set its `href` attribute to where the file comes from. For example:
`results.append("<link rel='related' href='images/letter-CU00048376.pdf'>");`
The link doesn't have to be static; you could use data from the record set to build the link, for example:

```
var customerID = record.fields.ID;
results.append('<link rel="related" href="images/letter-' + customerID + '.pdf">');
```

5. Add the `title` attribute to specify a custom attachment name. For example:
   `results.append("<link rel='related' href='images/{8FCEC8BC-72E8-486B-A206-516BF10E21F6}.pdf' title='INV1375461.pdf'>");`
6. Click **OK** to save the script.

Post Pagination Scripts

Post Pagination Scripts are run in a **Print** context **after** the content has been paginated. Because they can search through the output of all Print sections, and modify Print sections (one at a time), they may be used to create a Table Of Contents (TOC), as explained in the topic: "Creating a Table Of Contents" on page 1116.

This topic explains what a Post Pagination Script is and how to add it to a template.

The basics of script-writing in the Designer are explained in the following topic: "Writing your own scripts" on page 1064.

What Post Pagination Scripts are

Post Pagination Scripts are a special kind of Designer script: they are applied to the output of all sections in a **Print** context **after** the content has been paginated.

Post Pagination Scripts differ from Standard Scripts and Control Scripts in two ways:

- Post Pagination Scripts run **after** all other scripts, more precisely: after the content has been *paginated*. The pagination process applies page breaks to the content of a

Print section, adds Master Pages and sets the Media. A Post Pagination Script may query the rendered document and collect information about elements (for instance, on which page they reside on) and sections (for instance, whether they are enabled). If needed, a Post Pagination Script can change the sheet configuration and re-paginate a section.

- Post Pagination Scripts only apply to the **Print** context. The output of other contexts is not paginated.

Just like Standard Scripts, Post Pagination Scripts have a **selector** (see: "Selectors in OL Connect" on page 1083). A selector selects parts of the content of a section and stores them in the `results` object, so that they can be modified in the script (see "results" on page 1623).

The second most important object in a Post Pagination Script, just like in a Control Script, is a `section` (see "section" on page 1627).

What to use a Post Pagination Script for

After all Print sections have been paginated, Post Pagination Scripts may search through the rendered document and collect information about elements (for instance, which page they reside on) and sections (for instance, whether they are enabled). With this information, a Post Pagination Script can do two things:

- **Modify the output**. The script may modify the output of a section. It could, for example, use information like page numbers to create a Table Of Contents (TOC), as explained in the topic: "Creating a Table Of Contents" on the facing page. If needed, a Post Pagination Script can change the sheet configuration and re-paginate the section (see "paginate()" on page 1634.
- **Add information to the Connect database**. The script may add production information, such as the page, size or position of elements after a merge, as custom properties to a Print Content Item in the Connect database (see "contentitem" on page 1620). Custom properties can be utilized for further processing in a Workflow configuration with the Retrieve Items task. The Retrieve Items task retrieves custom properties along with the base record information (see Retrieve Items in Workflow's Online Help).

Adding a Post Pagination Script

To add a Post Pagination Script:

1. On the **Scripts** pane at the bottom left, click the black triangle on the **New** button and click **Post Pagination Script**. The new script appears in the Post Pagination folder.
2. Double-click the new script to open it. The script editor appears.

3. Change the name of the script so that it reflects what the script does.

   Note: Scripts can only have the same name when they are not in the same folder.

4. Write the script. If you are not familiar with scripting, see "Writing your own scripts" on page 1064.

Note: Post Pagination Scripts run only when a Print *section* is previewed or outputted. To verify the results of Post Pagination Scripts on a certain *Master Page*, preview the Print section to which that Master Page is applied.

Creating a Table Of Contents

This topic explains how to create a multi-page, cross-section Table Of Contents (TOC) using a Post Pagination Script.

For information about Post Pagination Scripts in general, see "Post Pagination Scripts" on page 1114.
The basics of script-writing in the Designer are explained in the following topic: "Writing your own scripts" on page 1064.

Step 1: Opening a Print template

Create or open a Print template.
Make sure to use HTML headings level 1 and level 2 (**<h1>** and **<h2>**) in your Print sections, if you want to use the script sample that is given in Step 3. This script collects the text and page numbers of these headings and puts them in the TOC.
To quickly change a paragraph into a Heading, place the cursor inside of it, or select the paragraph (see: "Selecting an element" on page 766). Then select the appropriate element, either on the **Format** menu, or from the 'Element type' drop-down on the toolbar.
Of course, you could just as well create a table of contents using other heading levels, or even other elements. In that case you'll have to adjust the script accordingly.

Step 2: Creating a placeholder for the TOC

Create one extra Print section to put the Table Of Contents in.
Inside the extra section, insert an Article element (see "Inserting an element" on page 765). Give the element an ID, for example: toc-content. This element is the placeholder for the TOC.

Step 3: Inserting the Post Pagination script

Insert a Post Pagination script (see "Adding a Post Pagination Script" on the previous page). Double-click on the new script to open it.

Set its **selector** to the ID that you specified in Step 2, preceded by a #, for example: #toc-content.
Paste the following code in the script editor:

```
// Build the TOC
var toc = '';
merge.context.query("h1, h2").each(function() {
    var pageNo = this.info().pageNo;
    var text = this.text();
    var level = this.tagName();

    toc += '<p class="toc-entry ' + level + '">';
    toc += '<span class="text">' + text + '</span>';
    toc += '<span class="dots"></span>';
    toc += '<span class="number">' + pageNo + '</span></p>';
});

results.html( toc );

// Repaginate the result
merge.section.paginate();

// Update the page numbers
var $numbers = query('.number');
merge.context.query("h1, h2").each(function( index ) {
    var pageNo = this.info().pageNo;
    var entry = $numbers.get( index );
    if( entry ) {
        entry.text( pageNo );
    }
});
```

What the script does

First the script creates a variable to hold the table of contents: toc.
Then it collects all <h1> and <h2> elements - in other words, level 1 and 2 headings. The merge.context.query(selector) function searches **across all Print sections** (see "query (selector)" on page 1621).

The query returns a result set. Each of the elements in the result set goes through the callback function defined in each() (see "each()" on page 1482).

The callback function gets the element's page number, text and HTML tag name:

```
    var pageNo = this.info().pageNo;
    var text = this.text();
    var level = this.tagName();
```

Note that the info() function can also be used to get an element's sheet number, the section it is located in, and the total page count and sheet count of that section (see "PaginationInfo" on page 1622 and "info()" on page 1626). In this case only the page number is used.

Then the callback function adds an entry to the variable that holds the table of contents, using the retrieved info.

```
toc += '<p class="toc-entry ' + level + '">';
toc += '<span class="text">' + text + '</span>';
toc += '<span class="dots"></span>';
toc += '<span class="number">' + pageNo + '</span></p>';
```

The HTML tag name is added as a **class**. This can be used in a CSS file to style the entries in the table of contents according to their level. (See "Step 4: Styling the table of contents" on the next page.)
The empty span between the heading's text and page number has the class dots. This is used to put dots between heading and page number.
The number class (for the page number) is not only used in CSS, but also later on in the script.

The table of contents is inserted in the section with: results.html(toc); (see "html()" on page 1576).

The table of contents may get too long for a single page and affect the page numbers in other sections. In that case it is necessary to re-paginate the content; merge.section.paginate(); does the trick.

> **Note:** Whether page numbering restarts in each section, depends on the settings for page numbering; see "Configuring page numbers" on page 645. By default, page numbering starts with page 1 for each section.

If the pagination process has changed the page numbers, the TOC needs to be updated as well.

To do that, the script first has to collect the page numbers from the table of contents. This is where the number class comes in handy: var $numbers = query('.number');.
Note that this query() function, as opposed to merge.context.query(), only searches the **current** section (see "query()" on page 1513).
Then, the level 1 and 2 headings are collected from all Print sections again using merge.context.query("h1, h2").
The callback function in each() retrieves the heading's new page number. It then uses the index number of the heading in the result set to get the corresponding entry in the TOC: var entry = $numbers.get(index); (see "get(index)" on page 1538), and replaces it with the new page number.

Excluding headings

Often there are certain headings that you don't want to appear in the table of contents. The title of the table of contents itself, for example.
To exclude these headings from the table of contents, do the following:

1. Give all the headings that you want to be ignored the same class (see "ID and class" on page 764), for example `ignore-me`.
2. Add the `:not` selector to the queries in the script, specifying that class. Remember to put a dot before the name of the class. For example: `merge.context.query("h1:not(.ignore-me), h2")`.

Adding internal hyperlinks

It is possible to create links in a table of contents that point to the respective position in the document (for example when generating PDF output). An internal hyperlink points to an element with a specified ID. (See: "Hyperlink and mailto link" on page 850.)

Here's how to change the `each()` function in the script in order to add internal hyperlinks to the table of contents:

1. Add the following variable: `var linknumber = 1;`. This variable is used to generate unique IDs for elements that don't have an ID.
2. Before the first line that starts with 'toc ', add the following:

```
var anchorId = '';
if (this.attr("id")) {
    anchorId = this.attr("id");
}
else {
    anchorId = 'generatedID-' + linknumber;
    linknumber++;
    this.attr("id", anchorId);
}
```

   This code takes the element's ID. If an element doesn't have an ID, the script generates a new, unique ID for it.
3. Replace the line:
   `toc += '<span class="text">' + text + '</span>';`
   with this line:

```
toc += '<a class="text" href="#' + anchorId + '">' + text + '</a>';
```

4. The new line adds a hyperlink: <a … href> with the element's ID to the table of contents.

Step 4: Styling the table of contents

Each component of the table of contents inserted in Step 3 is wrapped in a element that has a class, for example: "dots" or even multiple classes: "toc-entry h1". (Class names are separated by a space.)

These classes can be used to style the table of contents with CSS. The principles of styling with CSS are explained in another topic: "Styling templates with CSS files" on page 881.

Add the following **styles** to the style sheet for the Print context (see "Step 1: edit CSS" on page 886) to align the page number to the right and fill the space between the text and the page number with leader dots.

```
p.toc-entry {
display: flex;
margin: 0 0 0.25em 0;
}

p.toc-entry .text {
flex-shrink: 0;
}

p.toc-entry .dots {
flex-shrink: 1;
white-space: nowrap;
overflow: hidden;
text-overflow: clip;
}

p.toc-entry .dots::after {
font-weight: normal;
content: ". . . .
```

- The **.pot file** is exported from the "Translations pane" on page 1260 in the Designer. The file contains a list of texts in the base language. In OL Connect this is the language in which the template was originally written. (A template's language can be set via the "Locale Settings" on page 1167 dialog.)
- A **.po file** holds translated texts for one language. Each .po file has an entry by which OL Connect recognizes the target language.

(See: "Exporting and importing translation files" on page 1126.)

These file types are widely used in translation software, so you can team up with a professional translator or translation agency to translate your templates, or you can do it yourself using a free online tool, such as the Poedit software.

Other ways to translate a template

The built-in translation feature is easy to use for labels and short texts in for instance invoices, web forms, transaction email messages, and pack lists. It is less suitable for long texts like insurance policies or terms and conditions.

- For longer texts it is recommended to use a **content management system** in combination with scripts.
- It is also possible to use **snippets** or separate **sections** for translation purposes in combination with scripts.

Translating a template

To make use of the built-in translation feature, take the following steps.

1. Tag content for translation. See "Tagging elements for translation" on the facing page.
2. Optional: enable the pluralization option on any translation entry that should conform with the number in a certain data field; see "Pluralization" on page 1125.
3. Export the entries on the Translations pane to a **.pot** file. See "Exporting a file for translation (.pot)" on page 1126.
4. Open the .pot file in a translation tool (e.g. Poedit), translate the texts and/or enter the (text with) plurals.
   Or, if you work with a translator or translation agency, send them the .pot file.
   The result is a **.po** file (one per language).
5. Import the .po file or files. See "Importing translations (.po)" on page 1127.

6. Set the template's Locale, or select the field that contains a Locale value, in order to get output in the correct language. See "Changing the locale" on page 921.

Translating snippets

"Snippets" on page 867 get translated when they are inserted in the output, if the text is tagged for translation (see "Tagging text in snippets" on the next page).

Translating content that is inserted by a script

Personalization scripts may add content to the output. OL Connect will apply translations to that content at the moment it is added to the template, for example with the `html()` or `replaceWith()` function. Translations are applied if that content is marked for translation, and if there is a matching translation entry (see "Tagging text that is inserted by a script" on page 1124).

Translating content via a script

If a template has been prepared for translation, personalization scripts that add content to the output may translate that content using the "translate()" on page 1590 function.

Tagging elements for translation

In order to mark text for translation, you have to tag the HTML element that holds the text.

When you add elements, such as text, images or a table, to the content of a template, you are actually constructing an HTML file. It is possible to edit the source of the HTML file directly in the Designer; see "Editing HTML" on page 762.

However, to tag an element for translation, you don't have to go into the Source view.

1. Place the cursor in the text that should be translated and select the HTML element in the breadcrumbs (see "Selecting an element" on page 766) or select the element via the Outline view. Elements that contain text are paragraphs, headings, table cells, buttons, labels etc.
2. Open the Translations pane. If it isn't visible, select: **Window > Show View > Translations** from the menu.
3. Click the **Tag Element** button in the toolbar of the **Translations** pane. This adds the `data-translate` attribute to the element. (Attributes are visible on the "Source tab" on page 1263.)

If there was no translation entry with the same text already, the text of the tagged element is copied to a new translation entry on the **Translations** pane.

One translation entry works for all HTML elements that are tagged for translation and have exactly the same text.

> **Note:** Remember, only text in HTML elements that are tagged for translation get translated in the output.
> Changing the text in a tagged element will break the relationship with the translation entry. This means that the text will not be translated. Edit the entry in the Translations pane or use the **Sync** button on the "Translations pane" on page 1260 to restore the connection.
> Note that placeholders (e.g. @Name@) and expressions (e.g. {{Name}}) in the template will get replaced with data before translation takes place. If a translation entry contains a placeholder or expression, the text will then no longer match the translation entry.
> Also note that, by default, *all* text in a Dynamic Table cell is replaced with data.
> To prevent this, use the technique described here: "Mixing text and data in a cell with a direct link to a data field" on page 986.

Same text, different translation: add a context

Should the same text be **translated differently** in different locations? Then you need to add a 'context' to the translation entry:

1. **Double-click** the translation entry, or select an element and press the **Ctrl** key when you click the **Tag Element** button.

   > **Tip:** Pressing the **Ctrl** key when you click the **Tag Element** button opens the Translation String Options dialog.

2. In the **Context** field, describe where the source text is used, for example: "This text is in the head of a table", or "Table|Head"

The description is added to the <data-translate> attribute of the HTML element in which the source text is located, as its value, for example: <data-translate="Table">.
Note that a translation will only be applied if the value of the <data-translate> attribute of the respective HTML element is *exactly* the same as the given Context (case-sensitive).

Tagging text in snippets

If the text that you want to tag for translation is located in a snippet, do not tag it in the section where the snippet is used. Instead, open the snippet (see "Snippets" on page 867) and tag the elements inside the snippet as described above.

Tagging text that is inserted by a script

OL Connect will also apply translations to content that is inserted by personalization scripts, but only if that content is marked for translation, and if there is a matching translation entry.

1. On the Translations pane, click on the **New Empty String** button to create the translation entry.
2. Make sure that any element in which the text is inserted, are tagged for translation. There are several ways to do that:
    - Let the script insert the tagged HTML element as well as the text, for example: `results.html( "<p data-translate>Bill to @CustNumber@</p>" );`.
    - If the script inserts text into an existing, empty HTML element, switch to the Source view and add the `data-translate` attribute to the element (e.g. <p id="p1" data-translate>).

Data placeholders in translation entries

A translation entry in a snippet may include placeholders for data fields, e.g. "Dear @name@". (For more information about placeholders see "Variable data in text: scripts and placeholders" on page 945.)
Placeholders must not be translated; otherwise the personalization script will no longer replace them with data

Typically, translators are familiar with entries that contain variables, but you may add a comment explaining that the placeholder is used as a variable and that it should not be translated.

To add a comment, simply double-click the entry in the **Translations** pane and enter your comment in the **Comments** field. Comments will be added to the POT file and are visible to translators in their translation tools.

HTML tags in translation entries

A translation entry may contain HTML tags, for example when part of the text is styled or when there is a hyperlink in the text. HTML tags must not be translated or removed. Normally, translators will recognize a simple bold or italic tag (... or ... respectively), but more complex elements like hyperlinks could cause problems. The translator may accidentally modify the tag, and break the hyperlink for example.

One way to avoid this is to split the text around and inside HTML tags into separate chunks, no matter how short. Select the respective text in the template and select **Wrap in Span** from the contextual menu. Subsequently tag the new element for translation.

Alternatively you could also add a comment explaining that the HTML tag should not be changed.

To add a comment, simply double-click the entry in the **Translations** pane and enter your comment in the **Comments** field. Comments will be added to the POT file and are visible to translators in their translation tools.

Pluralization

Ideally, in templates with variable data, the text automatically adjusts to numbers in the data. Take for example the following line in a template:
"This order contains @num_products@ product."
In the output, the text "@num_products@" will be replaced with the value of a data field, a number in this case. (See "Variable data in the text" on page 924.)
The word "product" should only remain the same when the actual number of products is 1:
"This order contains 1 **product**".
In all other cases it should be changed into "products", e.g.:
"This order contains 10 **products**".

This topic explains how to make OL Connect dynamically select a singular or plural word, depending on the value of a data field.

Adding plurals

Entering plurals, or text with plurals, is a natural part of the translation process of a template. This process starts by creating translation entries as described in the following topic: "Translating templates" on page 1120.

Note that it isn't required to translate the template effectively. You may also create a translation file for the original language of a template and use it just for pluralization by entering plurals only.

To enable pluralization on a translation entry:

1. Double-click the respective entry on the **Translations** pane.
2. Check the **Pluralization** option.
3. Select the data field that holds the number.

   Note: Pluralization cannot be based on a data field in a detail table.

4. Now there are two possibilities:

- Click OK. Proceed with the translation process after you've created all translation entries and enabled pluralization as needed. For further instructions see "Translating templates" on page 1120.
- Enter the singular and plural and click OK. The entries are stored in the template itself.
  OL Connect will automatically fall back to these entries when there is no translation file (.po) that matches the "Locale" on page 921. The singular is used when the value of the data field is 1, and the plural is used in all other cases.

Note: In some languages, like Arabic and Polish, words have multiple plurals for use with different numbers. Multiple plurals can only be entered in a translation tool. See "Translating templates" on page 1120.

Exporting and importing translation files

In the process of the translation of a template, two types of translation files are involved: **Portable Object (.po)** files and a **Portable Object Template (.pot)** file. These are standardized file types following the gettext format.

This topic explains how to export the .pot file and how to import a .po file.

Tip: Translation tools are capable of **combining** .pot files. Combining .pot files may also result in .po files that contain translations for entries in multiple templates. This can be an interesting feature, especially when there are multiple templates in a project, as this eliminates the need to maintain individual .po files for each template.

For information about the translation process, see "Translating templates" on page 1120.

Exporting a file for translation (.pot)

A .pot file holds a list of texts in the base language. In OL Connect, this is the language in which the template was originally written. The file contains the entries in the Translation pane, including any comments.

Once all translatable elements in a template are tagged (see "Tagging elements for translation" on page 1122), you can create the .pot file:

1. Click the **Export** button on the toolbar of the **Translations** pane.
2. Specify the name and select a folder for the .pot file.

Importing translations (.po)

A .po file holds translated texts for one language. PO files can be imported into the template file or used remotely.

Importing a .po file

Importing a .po file file is easy: **drag & drop** or **copy & paste** the file from the file system to the **Translations** folder in the **Resources** pane.
Alternatively, right-click the Translations folder in the Resources pane and select **New Translation > PO files...**

You could also **download a remote** .po file that is being used in the template. Expand the Translations folder on the Resources pane. Then right-click the .rpo file and select **Download Resource**.

Editing an imported .po file

When you double-click a .po file in the Translations folder it will be opened in the text editor of the Designer, where you can read and edit it. If you are working with a translator or translating agency, remember to send them a copy of the modified .po file to keep the .po files in sync.

Updating an imported .po file

OL Connect Designer monitors the .po file on disk (in the folder from which it was last imported) for changes. When a .po file on disk has changed, the respective .po file in the template gets a small asterisk behind its name. Right-click the file in the Translations folder and select **Update** from the contextual menu to import the latest changes.

Using a remote .po file

In order to use a remote .po file, you must add the URL to the Translations folder on the Resources pane.

1. Right-click the **Translations** folder on the **Resources** pane, and select **New Translation > New Remote PO file**.
2. Enter a name for the file as it appears in the Translations folder. For better management, it's best to use the same file name as the remote resource.
3. Enter the **URL** for the remote .po file.
   If the file is located on an external web server, this must be a full URL, including the http:// or https:// prefix, domain name, path and file name.
   If the file is located on your network you can click the Browse button or enter the path

and file name manually.

The complete syntax with the "file" protocol is: `file://<host>/<path>`

If the host is "localhost", it can be omitted, resulting in: `file:///<path>`

> **Example:** `file:///c:/resources/translations/de-DE.po`.

If the file is located on another server in your network, the path must contain five slashes after "file:".

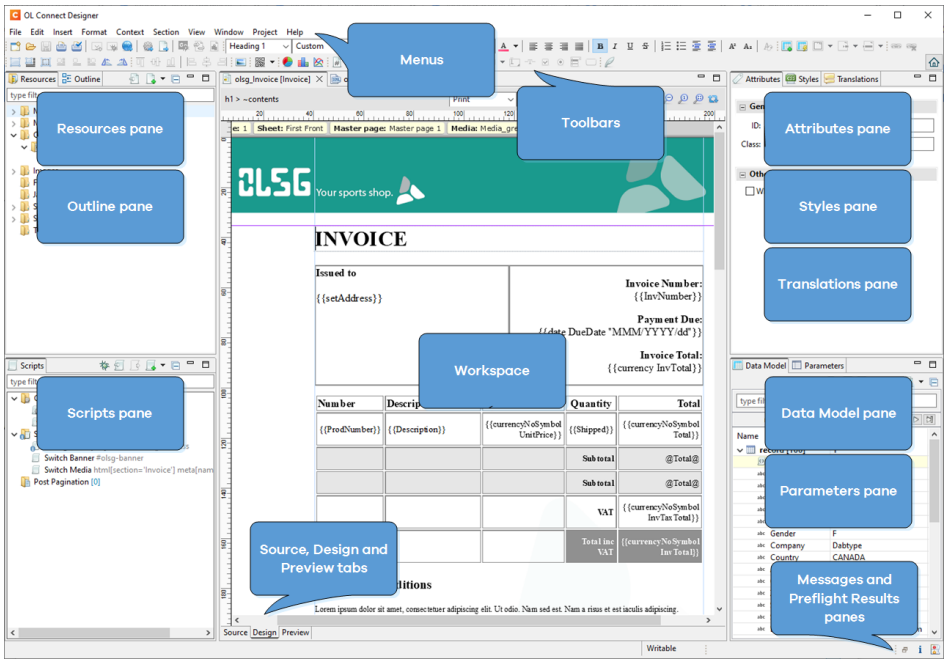
> **Note:** Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file:////myserver/myfolder/file.txt).

To **change the url** to a remote file that is currently being used in the template, expand the Translations folder on the Resources pane, then right-click the .rpo file and select **Properties...**.

Designer User Interface

The main ingredients in the Designer's user interface are the following:

- "Menus" on page 1229
- "Toolbars" on page 1264
- "Resources pane" on page 1250
- "Outline pane" on page 1250
- "Attributes pane" on page 1241
- "Styles pane" on page 1260
- "Translations pane" on page 1260
- Parameters pane (see "Runtime parameters" on page 608)
- "Workspace" on page 1261 (Source, Design and Preview tabs)
- "Data Model pane" on page 1244
- "Scripts pane" on page 1256
- "Preflight Results and Messages" on page 1248



Dialogs

Dialogs can allow you to perform a command or make settings. They can also ask you a question or provide you with information or progress feedback.

Here is a list of all dialogs:

Attachments dialog

The Attachments dialog lets you add and remove static Email attachments. For more information about email attachments, see: "Email attachments" on page 679.

- **Add**: Click this button to open the Select File dialog (see below).
- **Delete**: Deletes the file from the list of attachments. It doesn't remove any files from the template's resources.
- **Attach Print context as PDF**: If a Print Context exists in the template, its output will be generated and a PDF version of it will be attached to the outgoing email. This option can be overridden by the setting in the Send Test Email or Send Email dialog.

Select File dialog

Resources: lists the images that are present in the Images folder on the Resources pane. A preview of the selected image will be shown at the right.

Disk: lets you select a file that resides in a folder on a hard drive that is accessible from your computer. If possible, a preview of the selected file will be shown.

- **Path**. The complete syntax of the path is: file://<host>/<path>. Note: if the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/documents/mydocument.doc.
- **Browse**: opens an explorer window to browse folders and select a file.

Url: lets you select a file from a specific web address. Select the protocol and then enter the URL (for example, http://www.mysite.com/zipfiles/myfile.zip). If the selected file is an image, a preview will be shown below.

- **Protocol**: **http** or **https**.

> **Note:** Certain characters are invalid in a URL (for example, '$', '%', and '&') and must be percent-encoded. The same applies to a file path, since that actually is a URL that starts with the file protocol.
>
> Note that even a space character is invalid in a URL. Spaces in a URL are supported for backward compatibility, but it is recommended to percent-encode a space character as %20.

The option **Save with template** is available when choosing an image from disk or by URL. This inserts the image in the **Images** folder on the **Resources** pane. If not saved with the template, the image will remain external.

> **Note:** External files need to be available when the template is merged with a record set to generate output, and their location should be accessible from the machine on which the template's output is produced. External files are updated (retrieved) at the time the output is generated.

Bar Chart Properties dialog

The Bar Chart dialog appears when a Bar Chart object is right-clicked and the **Chart...** option is clicked. It determines how the Bar Chart is displayed in output and in Preview mode (see "Business graphics" on page 827).

All of the settings in this dialog are equivalent to a property in the amCharts library (see: amCharts' Class Reference). The description of each setting specifies the corresponding property. When you've made a setting, you can also find the property on the Source tab.

General tab

Most settings on the General tab correspond to properties of the AmSerialChart class in the amCharts library; see: AmSerialChart.

- **General Group**:
  - **Display grid above graph**: Check to display the grid on top of the bars so that it is always visible.
    (Equivalent to the `gridAboveGraphs` property; see: gridAboveGraphs.)
  - **Rotate**: Check to rotate the chart 90 degrees so that the bars are horizontal starting from the left.
    (Equivalent to the `rotate` property; see: rotate.)
  - **Stack Series**: Check to stack the values in one bar, instead of having one bar per value.
    (Equivalent to the `stackType` property of the `ValueAxis`; see: stackType.)
- **Text Group**: Determines how text is displayed in labels and legends.
  - **Font**: Enter the font-face to use to display text. The font must be installed on the system and defaults to Verdana if the font is not found.
    (Equivalent to the `fontFamily` property; see: fontFamily.)
  - **Size**: Enter the size of the font. Defaults to 11.
    (Equivalent to the `fontSize` property; see: fontSize.)
  - **Color**: Select the color in which to display text: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a valid CSS color (see MDN Web Docs - named color).
    (Equivalent to the `color` property; see: color.)
- **3D Group**: Creates a 3D effect if both settings in this group are higher than 0.
  - **Depth**: The depth of the 3D part of the plot area.
    (Equivalent to the `depth3D` property; see: depth3D.)
  - **Angle**: The angle of the 3D part of the plot area.
    (Equivalent to the `angle` property; see: angle.)

Value Axis tab

Settings on the Value Axis tab are equivalent to properties of the ValueAxis class in the amCharts library; see: ValueAxis.

- **Title group**:
  - **Label**: Enter a label for the value axis (the Y axis, or the X axis if the graph is rotated).
    (Equivalent to the `title` property; see: title.)
  - **Bold**: Check this option if the title should be bold.
    (Equivalent to the `titleBold` property; see: titleBold.)
  - **Color**: The default label color is black. To select a custom color for the label, click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `titleColor` property; see: titleColor.)
  - **Font Size**: Enter a font size for the label.
    (Equivalent to the `titleFontSize` property; see: titleFontSize.)
- **Grid group**:
  - **Color**: Select a color for the grid lines that divide the value axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `gridColor` property; see: gridColor.)
  - **Opacity**: Enter the opacity percentage of the grid. Default is 15%. 100 is fully opaque, 0 is transparent.
    (Equivalent to the `gridAlpha` property; see: gridAlpha.)
  - **Thickness**: Enter a thickness for the grid lines. Default is 1.
    (Equivalent to the `gridThickness` property; see: gridThickness.)
  - **Tick Length**: Length of the tick marks.
    (Equivalent to the `tickLength` property; see: tickLength.)
- **Axis group**:
  - **Color**: Select a color for the value axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `axisColor` property; see: axisColor.)

    - **Opacity**: Enter the opacity in percentage for the axis. 100 is fully opaque, 0 is transparent. Set opacity to 0 to hide the axis.
      (Equivalent to the `axisAlpha` property of the `ValueAxis`; see: axisAlpha.)
    - **Thickness**: Enter the thickness of the axis.
      (Equivalent to the `axisThickness` property of the `ValueAxis`; see: axisThickness.)
- **Labels**: Check this option to make the labels on the value axis visible. This is often useful in rotated charts.
  (Equivalent to the `labelsEnabled` property of the `ValueAxis`; see: labelsEnabled.)
    - **Frequency**: Defines per how many grid lines a label will be placed.
      (Equivalent to the `labelFrequency` property of the `ValueAxis`; see: labelFrequency.)

Category Axis tab

Settings on the Category Axis tab correspond to properties of the CategoryAxis class in the amCharts library; see: CategoryAxis.

- **Title group**:
    - **Label**: Enter a label for the category axis (the X axis, or the Y axis if the graph is rotated).
      (Equivalent to the `title` property; see: title.)
    - **Bold**: Check if you want the title to be bold.
      (Equivalent to the `titleBold` property; see: titleBold.)
    - **Color**: Select a custom color for the label (default is black): click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
      (Equivalent to the `titleColor` property; see: titleColor.)
    - **Font Size**: Enter a font size for the title.
      (Equivalent to the `titleFontSize`property; see: titleFontSize.)
- **Grid group**:
    - **Color**: Select a color for the grid lines that divide the category axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
      (Equivalent to the `gridColor` property; see: gridColor.)

- **Opacity**: Enter the opacity percentage of the grid. Default is 15%. 100 is fully opaque, 0 is transparent.
  (Equivalent to the `gridAlpha` property; see: gridAlpha.)
- **Thickness**: Enter a thickness for the grid lines. Default is 1.
  (Equivalent to the `gridThickness`property; see: gridThickness.)
- **Position**: Specifies if a grid line is placed on the centre of a cell or on the beginning of a cell.
  (Equivalent to the `gridPosition` property; see: gridPosition.)

  Tip:
  The position of the ticks (which is Middle by default) does not move with the grid.

  To change the tick position, add the following to the `categoryAxis` section on the Source tab: `"tickPosition": "start"`.

- **Tick Length**: Length of the tick marks.
  (Equivalent to the `tickLength` property; see: tickLength.)

  Tip:
  By default, one label per grid line will appear on the Category axis.

  To change that frequency, add the following to the `categoryAxis` section on the Source tab: `"labelFrequency": "2"` (replace 2 by the desired number of grid lines).

- **Axis group**:
  - **Color**: Select a color for the value axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `axisColor` property; see: axisColor.)
  - **Opacity**: Enter the opacity in percentage for the axis. 100 is fully opaque, 0 is transparent. Set opacity to 0 to hide the axis.
    (Equivalent to the `axisAlpha` property; see: axisAlpha.)
  - **Thickness**: Enter the thickness of the axis.
    (Equivalent to the `axisThickness` property; see: axisThickness.)

- **Auto Wrap**: Specifies if axis labels (when horizontal) should be wrapped if they don't fit in the allocated space.
  (Equivalent to the `autoWrap` property; see: autoWrap.)

Graphs tab

Most settings on the Graphs tab correspond to properties of the AmGraph class in the amCharts library; see: AmGraph.

- **Labels**:
  - **Text**: Text of the data labels. You can use tags like [[value]], [[description]], [[percents]], or[[category]]. (Equivalent to the `labelText` property; see: labelText.)
  - **Position**: Position of the data labels. Possible values are: "bottom", "top", "right", "left", "inside", "middle". (Equivalent to the `labelPosition` property; see: labelPosition.)
  - **Rotation**: Rotation of the data labels. Supported notations: `0.0`, `0.0°` or `0.0 deg`. (Equivalent to the `labelRotation` property; see: labelRotation.)
  - **Offset**: Vertical offset of the data labels. A positive offset moves the data labels up. A negative value moves them down. (Equivalent to the `labelOffset` property; see: labelOffset.)
  - **Anchor**: Select whether the start, middle or end of a data label should be anchored to the bar. This setting moves the labels horizontally. (Equivalent to the `labelAnchor`

property; see: labelAnchor.)

- **Columns**:
  - **Width**: Specify a bar width between 0 and 1. (This is a relative width, not pixel width). (Equivalent to the `columnWidth` property; see: columnWidth.)
  - **Spacing**: The gap in pixels between two columns of the same category. (Equivalent to the `columnSpacing` property of the `AmSerialChart` class; see: columnSpacing.)

Legend tab

Settings on the Legend tab correspond to properties of the AmLegend class in the amCharts library; see: AmLegend.

- **Show Legend**: Specifies if legend is enabled or not. (Equivalent to the `enabled` property; see: enabled.)
- **Legend Group**: Defines how the legends are shown.
  - **Equal label widths**: Check so that all labels are of equal width in the Legends box. The Legend's width will accommodate the largest value. (Equivalent to the `equalWidths` property; see: equalWidths.)
  - **Position**: Use the drop-down to select where the legend is shown: at the Right, Left, Top or Bottom. (Equivalent to the `position` property; see: position.)
  - **Align**: Use the drop-down to select how to align the text in the labels: Left, Middle or Right. (Equivalent to the `align` property; see: align.)
  - **Horizontal Space**: Horizontal space between legend items, in pixels. (Equivalent to the `spacing`property; see: spacing.)
  - **Vertical Space**: Enter a numerical value (in pixels) to define the vertical space between legend items, and also between the legend border and the first and last legend item. (Equivalent to the `verticalGap` property; see: verticalGap.)

    - **Max Columns**: Enter a numerical value to define the maximum number of columns in the legend. If the Legend's position is set to "right" or "left", this is automatically set to 1.
      (Equivalent to the `maxColumns` property; see: maxColumns.)
- **Labels Group**: Defines if and how labels are shown in the Legend.
    - **Text**: Enter the text used to display the labels; `[[title]]` is a variable that will be replaced with the title of the graph.
      (Equivalent to the `labelText`property; see: labelText.)
- **Markers Group**: Defines how the Legend's Markers look. Markers are icons with a color matching the Legend with its corresponding line.
    - **Type**: Use the drop-down to select in which shape the Markers are displayed; "none" hides the Markers completely.
      (Equivalent to the `markerType` property; see: markerType.)
    - **Size**: Enter the size (in pixels) for the Markers to be displayed.
      (Equivalent to the `markerSize` property; see: markerSize.)
    - **Label Gap**: Enter the distance (in pixels) between the legend marker and legend text.
      (Equivalent to the `markerLabelGap` property; see: markerLabelGap.)
    - **Border Width**: Use the drop-down to define the thickness of the border added to the Markers. The default value (0) means the line will be a "hairline" (1 px). In case the Marker type is line, this style will be used for the line thickness.
      (Equivalent to the `markerBorderThickness` property; see: markerBorderThickness.)
    - **Border Color**: Color of the Legend's border. Enter a hexademical color code (see Hex Calculator) or a valid color name (see MDN Web Docs - named color), or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
      (Equivalent to the `markerBorderColor` property; see: markerBorderColor.)
    - **Border Opacity**: Enter a numerical value between 0 and 100 to define the opacity (in percentage) of the border.
      (Equivalent to the `markerBorderAlpha` property; see: markerBorderAlpha. When specified on the Source tab, the value should be between 0 and 1, e.g. 0.8.)

Source tab

The JSON on the Source tab reflects the choices made in the other tabs and, more importantly, provides the possibility to add in any amCharts configuration option that is

unavailable via the other tab menus. For more information see: "Adding and editing properties manually" on page 832.

Box Formatting dialog

The Box Formatting dialog is accessible by clicking inside a box in the template and then selecting **Format > Box** in the menu.

All settings in this dialog are in fact CSS properties. Cascading Style Sheets (CSS) were originally designed for use with web pages: HTML files. Since Designer templates are HTML files, they are styled with CSS. To learn how to use CSS in the Designer, see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881. For information about specific properties and their options, see MDN Web Docs - CSS Reference.

Note: When no unit is added to a geometry value, such as the width, height, top or margin, the default unit will be added to the value; see "Print preferences" on page 1057.

Note: Using viewport based units (vw, vh, vmin, vmax) in Print sections is not recommended. This may cause differences between the preview in the Designer and the printed output.

For information about the different types of Boxes, see "Boxes" on page 823.

Box tab

- **General group**:
  - **Width**: Set the width of the box in measure or percentage. When no unit is entered, the default unit will be added to the value (see "Print preferences" on page 1057). Equivalent to the CSS `width` property.
  - **Height**: Set the height of the box in measure or percentage. When no unit is entered, the default unit will be added to the value (see "Print preferences" on page 1057). Equivalent to the CSS `height` property.
  - **Angle**: Set the rotation angle of the box in clockwise degrees. Equivalent to the CSS `transform:rotate` property.
  - **Corner radius**: Set the radius of rounded border corners in measure or percentage. Equivalent to the CSS `border-radius` property.
  - **Display**: Use the drop-down or type in the value for how to display the box. This can also be used to hide an element completely using the `none` option. Equivalent to the CSS `display` property. See MDN Web Docs - CSS Display.

- **Overflow**: Use the drop-down or type in the value for how to handle overflow (text that does not fit in the current size of the box). Equivalent to the `overflow` property.

- **Text wrap group**:
  - **Float**: Use the drop-down or type in the value for how to float the box, if the box is not in an absolute position (see Position, below). Equivalent to the CSS `float` property.
  - **Clear**: Use the drop-down or type in the value for clearing pre-existing alignments. Equivalent to the CSS `clear` property.
- **Positioning**: Note that it depends on both the Context and the type of Box, which settings are available. For information about the different types of Boxes, see "Boxes" on page 823.
  - **Position**: Use the drop-down or type in the value for the type of positioning for the box. Equivalent to the CSS `position` property (see "Using the CSS position property" on page 900).
  - **Top** (Web Context): Set the vertical offset between this box and its parent's top position. Equivalent to the CSS `top` property.
  - **Left** (Web Context): Set the horizontal offset between this box and its parent's left position. Equivalent to the CSS `left` property.
  - **Y-offset** (Print Context): Sets the vertical offset between this box and the current page's top. This only works for Positioned boxes in sections.
  - **X-offset** (Print Context): Sets the vertical offset between this box and the current page's left edge. This only works for Positioned boxes in sections.
  - **Bottom**: Set the vertical offset between this box and its parent's bottom position. Note that you can't set the Height *and* Top (or Y-offset) *and* Bottom of a Box. Remove the Height before setting the Bottom property. Equivalent to the CSS `bottom` property.
  - **Right**: Set the horizontal offset between this box and its parent's left position. Note that you can't set the Width *and* Left (or X-offset) *and* Right of a Box. Remove the Width before setting the Right property. Equivalent to the CSS `right` property.
  - **Z-index**: Set the z-index of the box. The z-index defines in which order elements appear. Equivalent to the CSS `z-index` property.

Background tab

For information about backgrounds see "Background color and/or image" on page 908.

- **General group**:
  - **Color**: Specify the color of the box background: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    Equivalent to the CSS `background-color` property.
- **Background image group**:
  - **Source**: click the **Select Image** button to select an image via the "Select Image dialog" on page 1206. Equivalent to the CSS `background` property.
  - **Size**: select `auto`, `cover` or `contain` (for an explanation see MDN Web Docs - background-size), or type the width and height of the image in a measure (e.g. `80px 60px`) or as a percentage of the box size (e.g. `50% 50%`). Equivalent to the CSS `background-size` property.
  - **Position**: select the position for the background-image. Equivalent to the CSS `background-position` property.

Spacing tab

For information about spacing see "Spacing" on page 922.

- **Padding group**: Defines padding (spacing inside the element) in measure or percentage:
  - **All sides**: Check to set all padding to use the Top value. Equivalent to the CSS `padding` property.
  - **Top, Left, Bottom, Right**: Set padding for each side. Equivalent to the CSS `padding-left`, `padding-top`, `padding-right` and `padding-bottom` properties.
- **Margin group**: Defines margins (spacing outside the element) in measure or percentage:
  - **All sides**: Check to set all margins to use the Top value. Equivalent to the CSS `margin` property.

- **Top, Left, Bottom, Right**: Set the margin for each side. Equivalent to the CSS `margin-left`, `margin-top`, `margin-right` and `margin-bottom` properties.

Border tab

For information about borders see "Border" on page 910.

- **Same for all sides**: Defines the border properties for all sides using the Top properties. Equivalent to the CSS `border` property.
- **Top, Left, Bottom, Right**: Each group defines the following properties:
  - **Width**: Specify the thickness of the border. Equivalent to the CSS `border-width`property.
  - **Style**: Specify the style of the border such as `solid`, `dashed` or `dotted`. Equivalent to the CSS `border-style` property.
  - **Color**: Specify the color of the border: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.Equivalent to the CSS `border-color` property.

Content tab

- **Copy Fit**: Check this option to automatically scale text inside the Box to the available space; see "Copy Fit" on page 896 .
  - **Minimum font size**: Specify the minimum font size using a valid font measurement unit (pt, px, in, cm or mm). Do not put a space between the number and the unit. The smallest possible size is 1pt. The default is 4pt.
    When left **blank**, the font size in Design view becomes the minimum font size. Text can only be made bigger than its initial size.
  - **Maximum font size**: Specify the maximum font size using a valid font measurement unit (pt, px, in, cm or mm). Do not put a space between the number and the unit. The biggest possible size is 1048pt. The default is 48pt.
    When left **blank**, the font size in Design view becomes the maximum font size. Text can only be made smaller than its initial size.
  - **Fit to width only**: When this option is checked, no line breaks will be added to the text.

- **Child** (optional): When specified, the Copy Fit feature will only be applied to the given child element (an element inside the Box or Div). Specify the element by giving its ID, for example: **#product**, or class, for example: **.product** - note the dot.

Type tab

The Type tab has all the settings for text.

- The font, font size, color and background color:
  - **Font**: Select the font used to display text. See also: "Fonts" on page 917. This is equivalent to setting the `font-family` property in CSS.
  - **Font size**. Enter the size in a measure, named size or percentage. This is equivalent to setting the `font-size` property in CSS.
  - **Color**: Specify the color of the text: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on the next page). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    This setting is equivalent to the `color` property in CSS.
  - **Background color**: Specify the background color of the text: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on the next page). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    This setting is equivalent to the `background-color` property in CSS.
- The spacing between letters and words and the way the text is wrapped:
  - **Letter Spacing**: The space between characters in a text in measure or percentage. This is equivalent to the letter-spacing property in CSS.
  - **Word Spacing**: Set the space between each word in a text in measure or percentage. This is equivalent to the `word-spacing` property in CSS.
  - **Whitespace**: Specify how the text wraps. See MDN Web Docs: white-space for details. This is equivalent to the `white-space` property in CSS.

- The style of the text. Check any option to apply the selected style to text within the element. This list shows the CSS property and value for each of the options:
    - **Bold**: Sets the `font-weight` to `700`.
    - **Italic**: Sets the `font-style` to `italic`.
    - **Underline**: Sets the `text-decoration` to `underline`.
    - **Strikethrough**: Sets the `text-decoration` to `line-through`.
    - **Subscript**: Sets the `vertical-align` to `super`.
    - **Superscript**: Sets the `vertical-align` to `sub`.
    - **Capitalize**: Sets the `text-transform` to `capitalize`.
    - **Uppercase**: Sets the`text-transform` to `uppercase`.
    - **Lowercase**: Sets the`text-transform` to `lowercase`.
    - **Small-caps**: Sets the `font-variant` to `small-caps`.

Formats tab

- **General group**:
    - **Line-height**: Specify the height of each line in the box's text, in measure or percentage. Note that this is not spacing between lines, but rather the complete height of the line itself including the text. Equivalent to the `line-height` property in CSS.
    - **Align**: Select how text should be aligned, such as `left`, `center`, `right` or `justify`. Equivalent to the `align` property in CSS.
    - **First Indent**: Specify the indentation of the first line of each paragraph in the box. Equivalent to the `text-indent` property in CSS.
        - **Direction**: Select in which direction text should be displayed (ltr, rtl, auto). Useful for certain languages such as Arabic, Hebrew, etc. Equivalent to the `dir` HTML attribute.
- **Breaks group**: Page break settings made on a box have an unpredictable effect; it is recommended not to change them.

Color Picker

The Color Picker dialog appears when creating a color in the formatting dialogs of certain elements, for example border colors in boxes and paragraphs.
For information about how to define and apply colors and how to keep them consistent in different output channels, see "Colors" on page 913.

The dialog consists of two main parts. On the left is the color wheel that can be used to select a color **hue** by clicking anywhere on that wheel.

To the right of the color wheel there is a vertical bar used to select the color **saturation**.

At the top-right, two colors are shown: the **New** box displays the currently selected color, while the **Original** shows the color currently attributed to the element.

The rest of the dialog has various options for choosing colors:

- **Color Mode**: Use the drop-down to select whether the color is set as **RGB**, **CMYK** or **HEX**. The color mode determines how the color is saved in the formatting properties, and how they are printed or outputted; see "Colors" on page 913.
- **RGB group**: Enter the Red, Green and Blue color values from 0 to 255.
- **HEX**: Enter a hexadecimal color code.
- The **eye dropper** lets you select a color from anywhere on your desktop. To open it, click the eye dropper button next to the HEX color field.
- **CMYK group** (suitable for Print output only): Enter the Cyan, Magenta, Yellow and Black color values from 0 to 100 percent.

Note: Whenever one value within this dialog is changed, all the other values are adjusted to its equivalent.

Colors Properties

The Colors Properties defines and sets named colors used in the template; see "Colors" on page 913. Named colors can be used throughout the templates, in all contexts. They are visible in color selector dialogs and useable with their names in style sheets; see "Styling and formatting" on page 879.

- **Color Type Selector**: Click and use the drop-down to display which color types to show in the list: All, RGB, CMYK or Spot colors.
- **Color List**: Displays the colors, filtered using the Color Type Selector.
- **New**: Create a new color using the **Edit Color** dialog (see below and see "Colors" on page 913).
- **Edit**: Edit the currently selected color using the **Edit Color** dialog (see below and see "Colors" on page 913).
- **Delete**: Delete the currently selected color.
- **Duplicate**: Duplicate the currently selected color using the name [color]CopyX.

Edit color

You can edit the following color properties.

- **Name**: Enter the name of the color. This name should not contain spaces or special characters.
- **Create name based on values**: Check so that the name is automatically based on the color slider values below.
- **Type**: Use the drop-down to specify which type of color this should be: either a Tint or a Color.
- **Option group**: contains the options for the chosen type. Options change depending on the selected type.
    - **Color**:
        - **Model**: This can be either CMYK or RGB.
        - **Spot Color**: Check to set the color as Spot Color. When Spot Colors are used, the Name must match that of the spot color used in the printer.
        - **Cyan/Magenta/Yellow/Black** (CMYK): Each slider sets a percentage for the color. Set the values using the sliders, or type in the percentage directly in the input boxes.
        - **Red/Green/Blue** (RGB): Each slider sets the values of 0-255 for the color. Set the value using the sliders or type in the value directly in the input boxes.
        - **Color Preview**: Box displaying the preview of the color (converted to RGB when relevant).
    - **Tint**:
        - **Source**: Select an existing *Color* in the template. The *tint* or *opacity* will be applied to this color.
        - **Tint/Opacity**: The slider sets the percentage of tint or opacity. Set the value using the slider, or type the percentage directly in the input box.
        - **Use Opacity**: Check to set the Tint slider to use Opacity instead.

Color Settings

Color Management can keep colors consistent across different outputs by using Color Profiles. When producing output to a new device, color adjustments are made to present the color as accurately as possible on this new device.

Note: The color management settings only affect the output, not the preview in Designer. CMYK colors may look different in Designer when compared to the output, but the colors in the output match the colors configured in the template.

- **Enable Color Management**: Check to disable color management and ignore embedded color profiles when importing images (with the exception of imported PDF files as it might contain a multiple tagged sub images).
- **Working Space Group**: Defines the color profiles for the current template.
    - **RGB**: Use the drop-down to select a color profile for RGB colors. The list displays ICC profiles located in "%USERPROFILE%\Connect\color-profiles\rgb".
    - **CMYK**: Use the drop-down to select a color profile for CMYK colors. The list displays ICC profiles located in "%USERPROFILE%\Connect\color-profiles\cmyk"
    - **Gray**: Use the drop-down to select a color profile for Grayscale. The list displays ICC profiles located in "%USERPROFILE%\Connect\color-profiles\gray"
- **Untagged Images Group**: Defines color profiles for any image that does not specifically have color profiles or color settings enabled.
    - **RGB**: Use the drop-down to select a color profile for RGB colors. The list displays ICC profiles located in "%USERPROFILE%\Connect\color-profiles\rgb".
    - **CMYK**: Use the drop-down to select a color profile for CMYK colors. The list displays ICC profiles located in "%USERPROFILE%\Connect\color-profiles\cmyk"
    - **Gray**: Use the drop-down to select a color profile for Grayscale. The list displays ICC profiles located in "%USERPROFILE%\Connect\color-profiles\gray"
- **Options Group**:
    - **Rendering intent**: Use the drop-down to specify how colors are converted that are out of range of a profile. For example, you may use tricks like reducing the saturation of the entire print so that a color that is out of range still appears a bit more vibrant than ones that are in range. Rendering intents use different methods to trick the eye into believing that the print can reproduce irreproducible colors.

Context properties

The respective Context Properties dialogs are opened via the contextual menu:

- Right-click a **context** in the Resources pane and choose one of its properties.

Which properties are available depends on the context. E.g. for a Print context, you can select either Color Output or Finishing.

Email context properties

For information about the Email context, see: "Email context" on page 667.

PDF Attachments tab

For an explanation of the options on the PDF Attachments tab, see "PDF Attachments dialog" on page 1176.

Print context properties

For information about the Print context, see: "Print context" on page 624.

Color Output tab

For an explanation of the options on the Color Output tab see "Overprint and black overprint" on page 628.

Finishing tab

For an explanation of the options on the Finishing tab and when they are applied, see: "Setting the binding style for the Print context" on page 627.

Web context properties

For information about the Web context, see "Web Context" on page 687.

Includes Tab

The Includes tab defines which style sheets and JavaScript files should be included in the Web context when generating output, and in which order; see "Includes dialog" on page 1157. For more information about stylesheets, see "Styling templates with CSS files" on page 881. For more information about using JavaScript, see "Using JavaScript" on page 703.

Edit Label Properties

The Edit Label Properties defines how a Pie Chart Label displays its title and data. It contains two options:

- **Label**: Enter a title for Labels and Legends when they are shown (see "Pie Chart Properties dialog" on page 1176).
- **Value**: Use the drop-down to select which Value to use as data within the Pie Chart as well as for Label and Legend values.

Enter Credentials dialog

The Connect **Enter Credentials** Dialog appears when attempting to run a Production Print (which prints via the Connect Server) from the Designer. The dialog box allows selection of the Server URL, as well as the username to be used for connecting to the Server, if this has not already been set.

This dialog appears the very first time Production Print is selected and any subsequent Production Print run, unless:

1. A valid user is entered in this dialog and the **Save username and password** checkbox is selected.
2. The Default user has already been assigned in the "Connect Servers preferences" on page 1061 sub-section of the Designer Preferences dialog.

Note: There is a standard Connect Server user that is added when installing Connect.

For fresh Connect installations the username is **olc-user** and for upgrades from versions prior to 2020.2 the username is **ol-admin** (password: **secret**).

This dialog box allows the following options:

- **Name**: A non-selectable field that is set to "Default".
- **URL**: The URL of the Server to connect to.
- **Ignore certificate errors on HTTPS connections** checkbox: Select if you wish to ignore such errors on HTTPS connections.
- **Authentication** checkbox: Select to enter the username for connecting to the Connect Sever. See note above.
  If selected, the follow options become available:
  - **Username**: Enter the user for connecting to the Connect Sever. See note above.
  - **Password**: Enter the password associated with this user.
- **Save credentials** checkbox: Select this checkbox to connect to this Sever using the entered credentials hereafter.

In addition to the standard **OK** and **Cancel** buttons, this dialog box provides a **Test Connection** button. This can be used to test the validity of the credentials entered in the dialog.
When the Test is successful, a pop up will be displayed, stating that the dialog successfully connected to the Server.

Note: When using HTTPS in communication with the OL Connect Server, the hostname entered and certificates used must conform with SNI requirements, i.e. the host name must contain at least one dot, and an IP address can only be used if the certificate permits it.

Export Template Report wizard

The Export Template Report wizard is opened through the menu: **File > Export report**.

It contains the following options:

- **File Name**: Enter the desired file name for the template report.
- **Format**: Select PDF or XML.

Click **Next** to go to the next set of options:

- **Generate Thumbnails**: The generated thumbnail images will be included in the PDF file. When exporting an XML report, the thumbnails will be saved in the same folder as the XML, in a subfolder named Thumbnails.
    - **Sections (first page)**: Save the first page of each section as a thumbnail image.
    - **Image resources**: Save all image resources as thumbnail images.
- **PDF Options**:
    - **Standard**: The standard PDF report contains a summary of the template with an overview of all the settings and resources that are used in the template: media, master pages, contexts, sections, images, scripts, runtime parameters, etc. The file properties are included as well (see "File Properties dialog" on the facing page). The report has a standard layout.
    - **Custom**: To create a custom PDF report, you need two files:
        - **Template**: A template design with the desired layout and variable data. This .OL-TEMPLATE file has to be made in the Designer.
        - **DataMapper**: A data mapping configuration that provides the variable data. This .OL-datamapper file has to be made in the DataMapper module, using the standard XML template report as data sample.

Tip: The template and data mapping configuration that are used to generate the standard template report can serve as examples. They are available online: see template-report-resources.zip.

File Properties dialog

The **File Properties** dialog shows the characteristics of the template, such as the file name, author, creation date and keywords. It is also possible to add custom properties; customer name, for example.
To open the File Properties dialog, select **File > Properties**.

When a template report is generated, the file properties are part of it (see "Exporting a template report" on page 594).

File properties can be used in a template in two ways:

- Via a Handlebars expression. For example: {{@properties.Company}} inserts the value of the Company name property in the template. See also: "Variable data in text: expressions" on page 999 and "Handlebars expressions" on page 1005.
- In scripts, via the `template` object; see the Designer API: "template" on page 1613. If you are not familiar with writing scripts, refer to "Writing your own scripts" on page 1064.

Description tab

- **File** (read-only): The name of the file.
- **Author**: The name of the author.
- **Company**: The company the author works for.
- **Description**: Enter a description of the template.
- **Keywords**: Enter one keyword or a list of keywords, separated by semicolons.
- **Created** (read-only): The date on which the template was created.
- **Modified** (read-only): The date on which the template was last modified.
- **Application** (read-only): the name and version of the application in which the template was last modified. The Connect version history of a template will be saved in the template file and exposed in a template report (see "Exporting a template report" on page 594).

Custom tab

On this tab custom properties can be defined. Click the first empty line, or click the Add icon, and enter a name for the property and a value.
Use the arrow icons to reorder the list of custom properties.
Custom property lists can be exported to, and imported from, a .CSV file, using the Import and Export buttons.

Find/Replace Dialog

The Find/Replace dialog can replace text within the current template. The scope of the replacement depends on the currently selected tab in the Workspace. If the Source tab is selected, the replace will affect the HTML source code. If the Design tab is selected, the replace will affect the text on the page. If the Preview tab is selected, the Replace feature is inactive.

> **Note:** When replacing text in the Design tab, formatting in the replaced text will be removed. If formatting is necessary in the new text, select the Source tab before opening the Find/Replace dialog and include the required HTML tags in the replacement text.

Here are the options available in this dialog:

- **Find**: The source string to find.
- **Replace with**: The string to replace the source with.
- **Direction**
    - **Forward**: Look forward from the current position of the pointer in the template or source.
    - **Backward**: Look backward from the current position of the pointer in the template or source.
- **Scope**
    - **All**: Searches in the complete text of the template or source.
    - **Selected lines**: Searches in the currently selected text or source.
- **Options**
    - **Case sensitive**: Use a case sensitive search, which differentiates **TEXT** from **text** or **TexT**.
    - **Wrap search**: Loop back from the end of the template or selection to its beginning, when the Search is at the end of the template or the selection.
    - **Whole word**: Searches for the source string as a whole word.
    - **Incremental**: With this option selected, each letter you type in the Find field causes the editor focus to move to the first complete occurrence of the text you are typing.
    - **Regular expressions**: Enables regular expressions for a search in the **Source** view of the workspace. After checking this option, you can type Ctrl + Space in either

text box to view a list of regular expressions.

> **Tip:** The Find/Replace dialog can fill in regular expressions in the Find field by itself. Open the dialog, check the option Regular expressions and close the dialog again. Select the text you want to search for and reopen the dialog: the Find field will now contain the regular expression for the text to find.

- **Find**: Click to find the next instance of the source string.
- **Replace/Find**: Click to replace the current instance with the replacement text and go to the next instance of the source string.
- **Replace**: Click to replace the current instance with the replacement text.
- **Replace All**: Click to replace all instances of the source string with the replacement text.
- **Close**: Close the dialog.

Font Manager

The Fonts Manager contains the fonts that were added to the template manually. It essentially lists the fonts located in the **Fonts** folder of the Resources pane (see "Fonts" on page 917).

Fonts with the same file name with a different extension are considered variations of the same font. For example, if there are three files, named gotham-book-webfont.eot, gotham-book-webfont.ttf, gotham-book-webfont.woff, only "gotham-book-webfont" appears in the Name column of this dialog.

The following buttons appear to the right of the list of fonts:

- **New**: Click to open the Edit Font dialog to add a new font.
- **Edit**: Click to open the Edit Font dialog to edit the currently selected font.
- **Remove**: Click to delete the currently selected font entry.
- **Duplicate**: Click to create a copy of the currently selected font entry.

Edit Font

The Edit Font dialog appears when clicking New or Edit from the Fonts Dialog.

- **Name**: Enter the name that should be used to refer to the font. This is equivalent to the `font-family` property of the @font-face CSS rule (see MDN Web Docs - font-family).

- **Font Weight**: Use the drop-down to select the default font weight (the thickness, see MDN Web Docs - font-weight):
  - **None**: Does not define the property.
  - **Normal**: Defines font-weight as normal
  - **Bold**: Defines the font-weight as bold (equivalent to a numerical value of 700).
  - **Numerical values**: Defines the line thickness; 400 is normal, 700 is bold.
- **Font Style**: Use the drop-down to select the font style (see MDN Web Docs - font-style):
  - **None**: Does not define the property.
  - **Normal**: Defines font-style as normal.
  - **Italic**: Makes the font italic.
  - **Oblique**: Makes the font oblique (this is generally the same as italic but does not require a special italic version of the font).
- **Name**: Check the fonts in the list to include them in the font definition.

Image Formatting dialog

The Image Formatting dialog is accessible by selecting an image in the template and then selecting **Format > Image** in the menu.

All settings in this dialog are in fact CSS properties. Cascading Style Sheets (CSS) were originally designed for use with web pages: HTML files. Since Designer templates are HTML files, they are styled with CSS. To learn how to use CSS in the Designer, see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881. For information about specific properties and their options, see MDN Web Docs - CSS Reference.

For more information about the use of images, see "Images" on page 852 and "Styling an image" on page 905.

Image Tab

- **General group**:
  - **Width**: Set the width of the image in measure or percentage. Equivalent to the CSS `width` property.
  - **Height**: Set the height of the image in measure or percentage. Equivalent to the CSS `height` property.
  - **Angle**: Set the rotation angle of the image in clockwise degrees. Equivalent to the CSS `transform:rotate` property.

- **Corner radius**: Set the radius of rounded border corners in measure or percentage. Equivalent to the CSS `border-radius` property.
- **Display**: Use the drop-down or type in the value for how to display the image. Equivalent to the CSS `display` property.
- **Overflow**: Use the drop-down or type in the value for how to handle overflow (the part of the image that does not fit in the current size of the box). Equivalent to the CSS `overflow` property.
- **Source**: Enter the web address or local file address of the image. Equivalent to the HTML `src` attribute.
- **Alternate text**: Enter an alternate text for the image. This is displayed in browsers and email clients when the image is loading or if the image cannot be displayed. It is also used for accessibility. Equivalent to the HTML `alt` attribute.

- **Text wrap group**:
  - **Float**: Use the drop-down or type in the value for how to float the image, if the image is not in an absolute position (see Position, below). Equivalent to the CSS `float` property.
  - **Clear**: Use the drop-down or type the value to clear pre-existing alignments. Equivalent to the CSS `clear` property.
- **Positioning**:
  - **Position**: Use the drop-down or type in the value for the type of positioning for the image. Equivalent to the CSS `position` property (see "Using the CSS position property" on page 900).
  - **Top**: Set the vertical offset between this image and its parent's top position. Equivalent to the CSS `top` property.
  - **Left**: Set the horizontal offset between this image and its parent's left position. Equivalent to the CSS `left` property.
  - **Bottom**: Set the vertical offset between this image and its parent's bottom position. Equivalent to the CSS `bottom` property.
  - **Right**: Set the horizontal offset between this image and its parent's right position. Equivalent to the CSS `right` property.
  - **Z-index**: Set the z-index of the image. The z-index defines in which order elements appear. Equivalent to the CSS `z-index` property.

Spacing Tab

- **Padding group**: Defines padding (spacing inside the element) in measure or percentage:
  - **All sides**: Check to set all padding to use the Top value. Equivalent to the CSS `border` property.
  - **Top, Left, Bottom, Right**: Set padding for each side. Equivalent to the CSS `border-left`, `border-top`, `border-right` and `border-bottom` properties.
- **Margin group**: Defines margins (spacing outside the element) in measure or percentage:
  - **All sides**: Check to set all margins to use the Top value. Equivalent to the CSS`margin`property.
  - **Top, Left, Bottom, Right**: Set the margin for each side. Equivalent to the CSS `margin-left`, `margin-top`, `margin-right` and `margin-bottom` properties.

Border Tab

- **Same for all sides**: Defines the border properties for all sides using the Top properties. Equivalent to the CSS `border` property.
- **Top, Left, Bottom, Right**: Each group defines the following properties:
  - **Width**: Specify the thickness of the border. Equivalent to the CSS `border-width` property.
  - **Style**: Specify the style of the border such as `solid`, `dashed` or `dotted`. Equivalent to the CSS `border-style`property.
  - **Color**: Specify the color of the border: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`. Equivalent to the CSS `border-color` property.

Import Resources dialog

The Import Resources dialog lets you import a variety of resources from other templates.

To open this dialog, click **File > Import Resources...** in the menu.

1. Select the template to copy resources from and click **OK**.
2. Now you can select various resources: "Master Pages" on page 648, "Media" on page 652, "Sections" on page 612, "Images" on page 852, style sheets (see "Styling templates with CSS files" on page 881), as well as scripts (see "Scripts pane" on page 1256) and the Data Model (see "Data Model pane" on page 1244).
   If you want to select individual items instead of all resources in a certain folder, expand the folder using the arrow to the left.

   Note: Resources may be interdependent. A Print section may need a certain Master Page or Media, for example; a Dynamic Image script relies on the presence of certain images; etcetera. This isn't visible in the Import Resources dialog. Make sure to select all necessary resources.
3. Choose what happens when a resource with the same name at the same hierarchy level already exists in your template.

   Names that represent files or folders on disk are case-insensitive, while all other names are case-sensitive.

   - **Rename**: Import the resource, giving its name a suffix to make it unique in the target folder.
     Note that when a section is renamed, the properties of its context are not copied over, except for Includes (Web contexts only; see "Includes" on page 688).
     Includes are transferred to the imported section.
   - **Overwrite**: Replace the existing resource.
     If the imported resource is a section, the properties of its context are copied to the target Context. For a Print context this can include settings for Color Output and Finishing.
     When replacing a resource that is opened in a separate editor (not the main editor), that editor will be closed and any unsaved changes will be lost.
   - **Skip**: Don't import the resource.

   Note: When it's selected for import, the Data Model is always overwritten, regardless of the policy for duplicates.
4. Click OK to import the resources.
5. Verify that the template works as expected. Make sure that the template contains everything that the imported resources need, and that all context and section settings are correct. Doing a Preflight is recommended (see "Doing a Preflight" on page 1075).

Includes dialog

The Includes dialog defines which style sheets and JavaScript files should be applied to a section when generating output (see: "Styling templates with CSS files" on page 881 and "Using JavaScript" on page 703).

To open this dialog and make settings for one section, right-click the section on the **Resources** pane and select **Includes**.
To do this for all Web sections at the same time, right-click the Web context on the Resources pane, or select **Context > Includes** on the main menu. (This menu option is only available when a Web section is being edited in the Workspace.)

Email clients do not read CSS files and some even remove a <style> tag when it is present in the email's header. Nevertheless, CSS files can be used with the Email context in the Designer.
When generating output from the Email context, all CSS rules that apply to the content of the email can be processed and added either to the header of the email or to inline style properties as if local formatting was applied, depending on the Email section properties. See "Email section properties" on page 1199.

1. From the **File types** dropdown, select **Stylesheets**, **JavaScripts** or **all**.
2. The list at the left displays the style sheets and/or JavaScript files that are present in the template's resources. The list at the right shows the style sheets and/or JavaScript files that will be included in the output of the current section (or in all Web sections, if you are making settings for the Web context).

   Use the **Include** and **Exclude** buttons to move files from one list to the other.
3. Files are included in the order shown. To change this order, click one of the included files and use the **Up** or **Down** button.

   > **Note:** The styles in each following style sheet add up to the styles found in previously read style sheets. When style sheets have a conflicting rule for the same element, class or ID, the **last** style sheet 'wins' and overrides the rule found in the previous style sheet.

JavaScript Properties

The JavaScript Properties dialog appears when you right-click a JavaScript file on the **Resources** pane and select **Properties**. (See also: "Using JavaScript" on page 703.)

It contains three options:

- **Name**: This option lets you rename the JavaScript resource in the template. The link between the JavaScript file and Web sections that include it will not be broken by renaming the file. (See also: "Including a JavaScript file in a Web context" on page 706.)
- **Async**: When **async** is checked, the script executes asynchronously with the rest of the page (while the page continues the parsing).
- **Defer**: This option postpones the execution of the script until the page has finished parsing. This is required by APIs like Google Maps.

It is recommended to check at least one of the options: async or defer, or both. Page content will become visible more quickly, resulting in a better user experience.
If you enable **both** options, async takes precedence on modern browsers, while older browsers that support defer but not async will fallback to defer.
When **neither** of the options is checked, the script gets fetched and executed immediately, interrupting the page loading and rendering.

JSON sample data dialog

The JSON sample data dialog lets you load JSON data into the Data Model (see "Loading data" on page 926).
By default, the JSON data is mapped to corresponding fields in the existing Data Model. Data that does not correspond to any field is discarded.
You may use the Replace Data Model option to let the JSON data replace the existing Data Model.

The dialog is opened via the menu: **File > Data > Open JSON Sample Data...**, or via the JSON Sample Data toolbar button on the Data Model pane.

The options in this dialog are:

- **File**: The path and name of the JSON file to use. The file is assumed to be UTF-8 encoded.
- **Browse**: Opens an explorer window to browse folders and select a JSON file.
- The box below the file name allows to paste or enter JSON data. After opening a JSON file the JSON data will appear in this box. You can review and edit the JSON.
  For the types of JSON that are accepted, see below.
- **Replace Data Model**: Selecting the Replace Data Model option removes the existing Data Model from the template and creates a new Data Model based on the keys found in the JSON. When this option is not checked, the JSON data is mapped to any corresponding fields in the existing Data Model, and data that does not correspond to

any field is discarded.

> **Note:** If a data mapping configuration is open at the same time, the loaded JSON will always replace the Data Model and no values will be imported.

- **Finish**: If the JSON is valid, you may click Finish to import the data into the Data Model pane. Arrays of objects are converted to records (or detail tables); key-value pairs are converted to data fields.

> **Note:** The JSON Sample Data is not stored with the template. In order to reuse the data you should create a JSON file on disk and load it via the **File** option in this dialog.

Types of JSON data

You can add the following types of JSON data:

- **A JSON object or an array of JSON objects** representing records. The data type is derived from the data:
  - Any value surrounded with quotes is converted to a field of type String.
  - Any numeric value containing a period is converted to a field of type Float.
  - Any numeric value that does not contain a period is converted to a field of type Integer.
  - A value "true" or "false" is converted to a field of type Boolean.
- **Typed JSON**. This JSON follows the structure of a JSON Record Data List (see the REST API Cookbook). Field types are determined by the `schema` object in the JSON.

Any nested JSON objects are displayed on successive levels in the Data Model.

Examples

Example: A single record with two fields

```
{ "first": "Peter", "last": "Parker" }
```

Example: Two records

```
[{ "first": "Peter", "last": "Parker" },
{"first": "Martin", "last": "Moore"
}]
```

Example: A single record with a detail table

```
{
"name":"Peter Parker",
"email":"parkerp@localhostcom",
"detail": [{"id":"123","sports":"running"},
{"id":"456","sports":"swimming"}]
}
```

Example: A JSON Record Data List

A JSON Record Data List describes a list of data fields (as name/value pairs), a data table schema and nested data records (if any) for one or more data records. This example holds a single, simple record, and two runtime parameters.

```
{
"parameters": {
 "campaign":"Campaign Name",
 "folder":"myFolder"
 },
"data": {
 "schema": {
 "columns": {
 "ID": "STRING",
 "Gender": "STRING",
 "FirstName": "STRING",
 "LastName": "STRING"
 }
 },
 "fields": {
 "ID": "CU00048376",
 "Gender": "M.",
 "FirstName": "Benjamin",
 "LastName": "Verret"
 }
 }
}
```

Note that the "parameters" key can be left out if there are no runtime parameters.

For more examples see the REST API Cookbook, and Working with JSON data in the Workflow Online Help.

Line Chart Properties dialog

The Line Chart dialog appears when a Line Chart is right-clicked and the **Chart...** option is clicked. It determines how the chart is displayed when generating output and in Preview mode (see "Business graphics" on page 827).

All of the settings in this dialog are equivalent to a property in the amCharts library (see: amCharts' Class Reference). The description of each setting specifies the corresponding property. When you've made a setting, you can also find the property on the Source tab.

General tab

Most settings on the General tab correspond to properties of the AmSerialChart class in the amCharts library; see: AmSerialChart.

- **General Group**:
  - **Display grid above graph**: Check to display the grid on top of the bars so that it is always visible.
    (Equivalent to the `gridAboveGraphs` property; see: gridAboveGraphs.)
  - **Rotate**: Check to rotate the chart 90 degrees so that the bars are horizontal starting from the left.
    (Equivalent to the `rotate` property; see: rotate.)
  - **Stack Series**: Check to stack the values in one bar, instead of having one bar per value.
    (Equivalent to the `stackType` property of the `ValueAxis`; see: stackType.)
- **Text Group**: Determines how text is displayed in labels and legends.
  - **Font**: Enter the font-face to use to display text. The font must be installed on the system and defaults to Verdana if the font is not found.
    (Equivalent to the `fontFamily` property; see: fontFamily.)
  - **Size**: Enter the size of the font. Defaults to 11.
    (Equivalent to the `fontSize` property; see: fontSize.)
  - **Color**: Select the color in which to display text: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a valid CSS color (see MDN Web Docs - named color).
    (Equivalent to the `color` property; see: color.)

- **3D Group**: Creates a 3D effect if both settings in this group are higher than 0.
  - **Depth**: The depth of the 3D part of the plot area.
    (Equivalent to the `depth3D` property; see: depth3D.)
  - **Angle**: The angle of the 3D part of the plot area.
    (Equivalent to the `angle` property; see: angle.)

Value Axis tab

Settings on the Value Axis tab are equivalent to properties of the ValueAxis class in the amCharts library; see: ValueAxis.

- **Title group**:
  - **Label**: Enter a label for the value axis (the Y axis, or the X axis if the graph is rotated).
    (Equivalent to the `title` property; see: title.)
  - **Bold**: Check this option if the title should be bold.
    (Equivalent to the `titleBold` property; see: titleBold.)
  - **Color**: The default label color is black. To select a custom color for the label, click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `titleColor` property; see: titleColor.)
  - **Font Size**: Enter a font size for the label.
    (Equivalent to the `titleFontSize` property; see: titleFontSize.)
- **Grid group**:
  - **Color**: Select a color for the grid lines that divide the value axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `gridColor` property; see: gridColor.)
  - **Opacity**: Enter the opacity percentage of the grid. Default is 15%. 100 is fully opaque, 0 is transparent.
    (Equivalent to the `gridAlpha` property; see: gridAlpha.)
  - **Thickness**: Enter a thickness for the grid lines. Default is 1.
    (Equivalent to the `gridThickness` property; see: gridThickness.)

    - **Tick Length**: Length of the tick marks.
      (Equivalent to the `tickLength` property; see: tickLength.)

- **Axis group**:
    - **Color**: Select a color for the value axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
      (Equivalent to the `axisColor` property; see: axisColor.)
    - **Opacity**: Enter the opacity in percentage for the axis. 100 is fully opaque, 0 is transparent. Set opacity to 0 to hide the axis.
      (Equivalent to the `axisAlpha` property of the `ValueAxis`; see: axisAlpha.)
    - **Thickness**: Enter the thickness of the axis.
      (Equivalent to the `axisThickness` property of the `ValueAxis`; see: axisThickness.)
- **Labels**: Check this option to make the labels on the value axis visible. This is often useful in rotated charts.
  (Equivalent to the `labelsEnabled` property of the `ValueAxis`; see: labelsEnabled.)
    - **Frequency**: Defines per how many grid lines a label will be placed.
      (Equivalent to the `labelFrequency` property of the `ValueAxis`; see: labelFrequency.)

Category Axis tab

Settings on the Category Axis tab correspond to properties of the CategoryAxis class in the amCharts library; see: CategoryAxis.

- **Title group**:
    - **Label**: Enter a label for the category axis (the X axis, or the Y axis if the graph is rotated).
      (Equivalent to the `title` property; see: title.)
    - **Bold**: Check if you want the title to be bold.
      (Equivalent to the `titleBold` property; see: titleBold.)
    - **Color**: Select a custom color for the label (default is black): click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
      (Equivalent to the `titleColor` property; see: titleColor.)

- **Font Size**: Enter a font size for the title.
  (Equivalent to the `titleFontSize`property; see: titleFontSize.)

- **Grid group**:
  - **Color**: Select a color for the grid lines that divide the category axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
    (Equivalent to the `gridColor` property; see: gridColor.)
  - **Opacity**: Enter the opacity percentage of the grid. Default is 15%. 100 is fully opaque, 0 is transparent.
    (Equivalent to the `gridAlpha` property; see: gridAlpha.)
  - **Thickness**: Enter a thickness for the grid lines. Default is 1.
    (Equivalent to the `gridThickness`property; see: gridThickness.)
  - **Position**: Specifies if a grid line is placed on the centre of a cell or on the beginning of a cell.
    (Equivalent to the `gridPosition` property; see: gridPosition.)

    Tip:
    The position of the ticks (which is Middle by default) does not move with the grid.

    To change the tick position, add the following to the `categoryAxis` section on the Source tab: `"tickPosition": "start"`.
  - **Tick Length**: Length of the tick marks.
    (Equivalent to the `tickLength` property; see: tickLength.)

    Tip:
    By default, one label per grid line will appear on the Category axis.

    To change that frequency, add the following to the `categoryAxis` section on the Source tab: `"labelFrequency": "2"` (replace 2 by the desired number of grid lines).

- **Axis group**:

- **Color**: Select a color for the value axis: click the colored square to open the Color Picker dialog ("Color Picker" on page 1143), or enter a hexademical color code (see Hex Calculator) or a predefined CSS color (see MDN Web Docs - named color).
  (Equivalent to the `axisColor` property; see: axisColor.)
- **Opacity**: Enter the opacity in percentage for the axis. 100 is fully opaque, 0 is transparent. Set opacity to 0 to hide the axis.
  (Equivalent to the `axisAlpha` property; see: axisAlpha.)
- **Thickness**: Enter the thickness of the axis.
  (Equivalent to the `axisThickness` property; see: axisThickness.)
- **Auto Wrap**: Specifies if axis labels (when horizontal) should be wrapped if they don't fit in the allocated space.
  (Equivalent to the `autoWrap` property; see: autoWrap.)

Graphs

Settings on the Graphs tab correspond to properties of the `AmGraph` class in the amCharts library; see: AmGraph.

- **Line**:
  - **Thickness**: Sets the thickness of the lines in a Line chart.
    (Equivalent to the `lineThickness` property; see: lineThickness.)

Tip: Set `"fillAlphas" = 1` on the Source tab to color the space below the lines.

Legend tab

Settings on the Legend tab correspond to properties of the AmLegend class in the amCharts library; see: AmLegend.

- **Show Legend**: Specifies if legend is enabled or not.
  (Equivalent to the `enabled` property; see: enabled.)
- **Legend Group**: Defines how the legends are shown.
  - **Equal label widths**: Check so that all labels are of equal width in the Legends box. The Legend's width will accommodate the largest value.
    (Equivalent to the `equalWidths` property; see: equalWidths.)
  - **Position**: Use the drop-down to select where the legend is shown: at the Right, Left, Top or Bottom.

(Equivalent to the `position` property; see: position.)

- **Align**: Use the drop-down to select how to align the text in the labels: Left, Middle or Right.
  (Equivalent to the `align` property; see: align.)
- **Horizontal Space**: Horizontal space between legend items, in pixels.
  (Equivalent to the `spacing`property; see: spacing.)
- **Vertical Space**: Enter a numerical value (in pixels) to define the vertical space between legend items, and also between the legend border and the first and last legend item.
  (Equivalent to the `verticalGap` property; see: verticalGap.)
- **Max Columns**: Enter a numerical value to define the maximum number of columns in the legend. If the Legend's position is set to "right" or "left", this is automatically set to 1.
  (Equivalent to the `maxColumns` property; see: maxColumns.)

- **Labels Group**: Defines if and how labels are shown in the Legend.
  - **Text**: Enter the text used to display the labels; `[[title]]` is a variable that will be replaced with the title of the graph.
    (Equivalent to the `labelText`property; see: labelText.)
- **Markers Group**: Defines how the Legend's Markers look. Markers are icons with a color matching the Legend with its corresponding line.
  - **Type**: Use the drop-down to select in which shape the Markers are displayed; "none" hides the Markers completely.
    (Equivalent to the `markerType` property; see: markerType.)
  - **Size**: Enter the size (in pixels) for the Markers to be displayed.
    (Equivalent to the `markerSize` property; see: markerSize.)
  - **Label Gap**: Enter the distance (in pixels) between the legend marker and legend text.
    (Equivalent to the `markerLabelGap` property; see: markerLabelGap.)
  - **Border Width**: Use the drop-down to define the thickness of the border added to the Markers. The default value (0) means the line will be a "hairline" (1 px). In case the Marker type is line, this style will be used for the line thickness.
    (Equivalent to the `markerBorderThickness` property; see: markerBorderThickness.)

- **Border Color**: Color of the Legend's border. Enter a hexademical color code (see Hex Calculator) or a valid color name (see MDN Web Docs - named color), or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
  (Equivalent to the `markerBorderColor` property; see: markerBorderColor.)
- **Border Opacity**: Enter a numerical value between 0 and 100 to define the opacity (in percentage) of the border.
  (Equivalent to the `markerBorderAlpha` property; see: markerBorderAlpha. When specified on the Source tab, the value should be between 0 and 1, e.g. 0.8.)

Source tab

The JSON on the Source tab reflects the choices made in the other tabs and, more importantly, provides the possibility to add in any amCharts configuration option that is unavailable via the other tab menus. For more information see: "Adding and editing properties manually" on page 832.

Locale Settings

The Locale dialog box sets the locale used inside the template. The Locale can affect time, currency output, and other formatting that depends on location and language (see "Locale" on page 921). The default Locale for new templates can be set via the Preferences ("Language preferences" on page 1054).

- **Use**: Use the drop-down to select how the Locale is set for the current template.
  - **System Locale**: Select this to use the operating system's locale settings. This is set in the Region settings of the control panel.
  - **Explicit Locale**: Select this option to specify a static locale which will remain static for this template, whichever server the template is used on.
  - **Data Field**: Select this to use a data field from the record. The locale will be record-specific in this case.
  - **Runtime Parameter**: Select this to use the value of a runtime parameter. Runtime parameters are defined in the template, but their value is set at runtime; see "Runtime parameters" on page 608.
- **Locale**: Use the drop-down to select a specific locale. Only enabled when **Explicit Locale** is selected above.
- **Data Field**: Use the drop-down to select a field within the current data model that contains the locale.

- **Runtime Parameter**: Use the drop-down to select a runtime parameter that will contain the locale.

In case of a data field or runtime parameter, the value must be a string and contain the exact locale to be used, such as "en" or "fr-CA". It cannot be an alias such as "english" or "french". The locale supports both ISO-639-1 alone ("en", "fr", etc) or ISO-639-1 followed by a 2-letter country code ("de-DE", "zh-CN", "fr-CA", "fr-FR", etc).

Master Page Properties

Master Pages can only be used in a Print context; see "Master Pages" on page 648.

The following properties are available for Master Page resources:

- **Name**: The name of the master page, displayed in all drop-downs where the Master Page is shown as well as in the "Resources pane" on page 1250.
- **Margins** group:
  - **Header**: The space at the top of the Master Page where no content will print, when this Master Page is used in a Section.
  - **Footer**: The space at the bottom of the Master Page where no content will print, when this Master Page is used in a Section.

Media Properties

Media can only be used in a Print context. For an explanation of what Media are and how to use them, see "Media" on page 652.

Media are not printed, unless you want them to; see "Printing virtual stationery" on page 659.

To open the Media properties dialog, right-click one of the Media in the Media folder on the **Resources** pane and select **Properties**.

The **Page Setup** button closes the Media Properties dialog and opens the Print Section Properties dialog (see "Print section properties" on page 1200).

Properties Tab

- **Name**: The name of the Media, displayed in all drop-downs where the Media is shown as well as in the Resources Pane.
- **Size**: This group is read-only and only used to display the size selected in the linked Print section's properties (see "Print section properties" on page 1200).

- **Page Size**: The named page size.
- **Width**: The width of the page.
- **Height**: The height of the page.

Virtual Stationery Tab

- **Front/Back group**: Defines the pre-printed media used for the front and back of the Virtual Stationery.
  - **PDF**: Click the **Select Image** button to open the "Select Image dialog" on page 1206 and select which PDF (and optionally, which page of the PDF) to display as a background for the page.
    The **Clear** button can be used to clear the path. If the path is not empty it will be used when the Media is linked to a section via a Control script.
  - **Position**: Use the drop-down to select how the PDF is displayed on the page:
    - **Fit to Media**: Select to stretch the PDF to fit the media size.
    - **Centered**: Select to center the PDF on the page, vertically and horizontally.
    - **Absolute**: Select to place the PDF at a specific location on the page. Use the Top and Left options below to specify the positioning of the PDF.
      - **Top**: The distance between the top side of the page and the top side of the PDF.
      - **Left**: The distance between the left side of the page and the left side of the PDF.
- **Front side**: Select the image that is shown as a background for all "front" sides in the template.
- **Back side**: Select the image that is shown as a background for all "back" sides in the template.

Characteristics tab

The characteristics define the type of paper on which the Print context is meant to be printed.

- **Media Type**: The type of paper, such as *Continuous*, *Envelope*, *Labels*, *Stationery*, etc.
- **Weight**: The weight of the media in grammage (g/m2).
- **Front Coating**: The pre-process coating applied to the front surface of the media, such as *Glossy*, *High Gloss*, *Matte*, *Satin*, etc.

- **Back Coating**: The pre-process coating applied to the front surface of the media.
- **Texture**: The texture of the media, such as *Antique*, *Linen*, *Stipple* or *Vellum*.
- **Grade**: The grade of the media, such as *Gloss-coated paper*, *Uncoated white paper*, etc.
- **Hole Name**: Pre-defined hole pattern that specifies the pre-punched holes in the media, such as *R2-generic*, *R2m-MIB*, *R4i-US*, etc.

New dialog

The New dialog lets you create a new OL Connect project, data mapping configuration or template.

To open this dialog, select **File > New** from the menu or click the **New** toolbar button.

Options for this dialog:

- **Data Mapping Configurations**: Lets you create a new data mapping configuration for a database or data file.
    - **Databases**: Select the database type and click **Next**. For the options on the next page, see "Using the wizard for databases" on page 346.
    - **Files**: Select the file type and click **Next**. Then select the file itself and click **Finish**. After opening the file, you have to make settings for the input data (see "Data source settings" on page 370) to make sure that the source data is parsed correctly and divided into logical units of data the way you want. Then you can start building the data extraction workflow.
- **Data mapping Wizards**: Lets you create a new data mapping configuration with a wizard. See "With a wizard" on page 340.
- **Sample Projects**: Lists all "Sample Projects" on page 1185. A Sample Project generates a Connect solution consisting of a Workflow configuration as well as any templates, data mapping configurations, and Print Presets that are used in that configuration.
- **Templates**: Lists all types of templates that you can create with a wizard. See "Templates" on page 589.
    - **Email Template**: This starts the Basic Action Email wizard; see "Creating an Email template with a Wizard" on page 664.
    - **PDF-based Print Template**: The PDF-based Print template wizard creates a document from an existing PDF file; see "PDF-based Print template" on page 619.
    - **Print Template**: This starts the Formal Letter Print wizard; see "Creating a Print template with a Wizard" on page 618.

- **Web Page Template**: This starts the most basic Web Page wizard; see "Creating a Web template with a Wizard" on page 683.
- **Word-based Print Template**: This starts the Microsoft Word-based Print wizard; see "Creating a Print template with a Wizard" on page 618.
- **Banded Email Templates**: The contents of the email templates that these wizards create extend to the edge of the window in which it is read. See "Basic Email and Banded Email" on page 666.
- **Basic Email Templates**: The contents of the email templates that these wizards create extend to the email's margin. See "Basic Email and Banded Email" on page 666.
- **Basic Print Templates**: From this folder you can run the Formal Letter or Postcard Print wizard; see "Creating a Print template with a Wizard" on page 618.
- **Capture OnTheGo Starter Templates**: COTG templates are used to generate forms for the Capture OnTheGo mobile application. See "Capture OnTheGo template wizards" on page 717.
- **ERP Templates**: ERP template wizards are used to create business documents, such as a sales invoice, purchase order, collection letter, etc.. See "ERP templates" on page 623.
- **Foundation Web Page Starter Templates**: These template wizards create Web templates that make use of the Zurb **Foundation** front-end framework. See "Creating a Web template with a Wizard" on page 683.
- **Slate: Responsive Email Templates by Litmus**: These template wizards create responsive HTML email templates. See "Slate: Responsive Email Templates by Litmus" on page 666.

Note: When you create a new project you are prompted to specify a folder for the sample project. You may also be required to enter your email address to which will be used to identify your changes in the project history.

Package dialog

The Package dialog saves templates, data mapping configurations and Print Presets as a package file. Package files can be sent to other Connect Designer users.

- **Configurations**: Initially this list contains the Connect files that are currently open.
    - **Browse**: This button opens the Select Files dialog that lets you add one or more templates (*.OL-template) data mapping configurations (*.OL-datamapper) and Print Presets (*.OL-jobpreset, *.OL-outputpreset) to the list.
    - **Delete**: Deletes the selected file from the list of configurations.
- **Package...**: This button opens the Save dialog. All selected Connect files will be saved to one package file. You can use the topmost checkbox to select all of the listed files at once.
- **Cancel**: Click this button to close the dialog without creating a package file.

Paragraph Formatting dialog

The Paragraph Formatting dialog controls how the selected paragraph is formatted. It is accessed by placing the cursor within a paragraph and then selecting **Format > Paragraph** on the menu.

All settings in this dialog are in fact CSS properties. Cascading Style Sheets (CSS) were originally designed for use with web pages: HTML files. Since Designer templates are HTML files, they are styled with CSS. To learn how to use CSS in the Designer, see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881. For information about specific properties and their options, see MDN Web Docs - CSS Reference.

Click the **Advanced** button to enter CSS properties and values directly.

The **Apply** button lets you preview the template with the new settings.

For information about text and how to style it see "Text and special characters" on page 865 and "Styling text and paragraphs" on page 892.

Formats tab

- **General group**:
    - **Line-height**: Specify the height of each line in the element's text, in measure or percentage. Note that this is not spacing between lines, but rather the complete height of the line itself including the text. Equivalent to the `line-height` property in CSS.
    - **Align**: Select how text should be aligned, such as `left`, `center`, `right` or `justify`. Equivalent to the `align` property in CSS.
    - **First Indent**: Specify the indentation of the first line of each paragraph in the element. Equivalent to the `text-indent` property in CSS.

- **Display**: Select how to display the element. This can also be used to hide an element completely using the `none` option. See MDN Web Docs - display. Equivalent to the `display` property.
- **Direction**: Select in which direction text should be displayed (ltr, rtl, auto). Useful for certain languages such as Arabic, Hebrew, etc. Equivalent to the `dir` HTML attribute.

- **Breaks group**:
  - **Before**: Specifies whether a page break should occur **before** the paragraph. This is equivalent to the `page-break-before` property in CSS; see MDN Web Docs - page-break-before for an explanation of the available options.
  - **Inside**: Specifies whether a page break is allowed inside the paragraph. Equivalent to the `page-break-inside` property in CSS; see MDN Web Docs - page-break-inside for an explanation of the available options.
  - **After**: Specifies whether a page break should occur **after** the paragraph. Equivalent to the page-break-after property in CSS; see MDN Web Docs - page-break-after for an explanation of the available options.
  - **Widows**: Specifies how to handle widows within the paragraph (lines appearing alone on the next page if the paragraph does not fit on the current one). Equivalent to the `widows` property. Widows and orphans are ignored if the `page-break-inside` property is set to `avoid`.
  - **Orphans**: Specifies how to handle orphans within the paragraph (lines appearing alone at the end of a page if the paragraph does not fit on the current one). Equivalent to the `orphans` property.

Note: For more information on page breaks, widows and orphans, see MDN Web Docs - Paged Media.

Spacing tab

For information about spacing see "Spacing" on page 922.

- **Padding group**: Defines padding (spacing inside the element) in measure or percentage:
  - **All sides**: Check to set all padding to use the Top value. Equivalent to the CSS `padding` property.

    - **Top, Left, Bottom, Right**: Set padding for each side. Equivalent to the CSS `padding-left`, `padding-top`, `padding-right` and `padding-bottom` properties.
- **Margin group**: Defines margins (spacing outside the element) in measure or percentage:
    - **All sides**: Check to set all margins to use the Top value. Equivalent to the CSS `margin` property.
    - **Top, Left, Bottom, Right**: Set the margin for each side. Equivalent to the CSS `margin-left`, `margin-top`, `margin-right` and `margin-bottom` properties.

Border tab

For information about borders see "Border" on page 910.

- **Same for all sides**: Defines the border properties for all sides using the Top properties. Equivalent to the CSS `border` property.
- **Top, Left, Bottom, Right**: Each group defines the following properties:
    - **Width**: Specify the thickness of the border. Equivalent to the CSS `border-width`property.
    - **Style**: Specify the style of the border such as `solid`, `dashed` or `dotted`. Equivalent to the CSS `border-style` property.
    - **Color**: Specify the color of the border: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.Equivalent to the CSS `border-color` property.

Type tab

The Type tab has all the settings for text.

- The font, font size, color and background color:
    - **Font**: Select the font used to display text. See also: "Fonts" on page 917. This is equivalent to setting the `font-family` property in CSS.
    - **Font size**. Enter the size in a measure, named size or percentage. This is equivalent to setting the `font-size` property in CSS.

- **Color**: Specify the color of the text: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
  This setting is equivalent to the `color` property in CSS.
- **Background color**: Specify the background color of the text: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
  This setting is equivalent to the `background-color` property in CSS.

- The spacing between letters and words and the way the text is wrapped:
  - **Letter Spacing**: The space between characters in a text in measure or percentage. This is equivalent to the letter-spacing property in CSS.
  - **Word Spacing**: Set the space between each word in a text in measure or percentage. This is equivalent to the `word-spacing` property in CSS.
  - **Whitespace**: Specify how the text wraps. See MDN Web Docs: white-space for details. This is equivalent to the `white-space` property in CSS.
- The style of the text. Check any option to apply the selected style to text within the element. This list shows the CSS property and value for each of the options:
  - **Bold**: Sets the `font-weight` to `700`.
  - **Italic**: Sets the `font-style` to `italic`.
  - **Underline**: Sets the `text-decoration` to `underline`.
  - **Strikethrough**: Sets the `text-decoration` to `line-through`.
  - **Subscript**: Sets the `vertical-align` to `super`.
  - **Superscript**: Sets the `vertical-align` to `sub`.
  - **Capitalize**: Sets the `text-transform` to `capitalize`.
  - **Uppercase**: Sets the`text-transform` to `uppercase`.

- **Lowercase**: Sets the`text-transform` to `lowercase`.
- **Small-caps**: Sets the `font-variant` to `small-caps`.

PDF Attachments dialog

The PDF Attachments dialog defines options for the Email context that are used when generating email output with PDF attachments (see "Generating Email output" on page 1673).

To open this dialog, right-click the **Email** context on the **Resources** pane and select **PDF attachments**.
Alternatively, select **Context > PDF Attachments** on the main menu. This option is only available when an Email section is being edited in the Workspace.

- **Print Context Image Compression**: Defines the properties of the PDF when attaching the Print context to email output.
  - **Lossless**: Enables maximum quality in the PDF. Note that this will produce a larger PDF.
  - **Quality**: Disabled when Lossless is checked. Determines the quality (aka compression) of the attached PDF.
  - **Tile Size**: Use the drop-down to select the size of the tiles used in the image. When low Quality values are used to optimize images smaller than 1024 x 1024 pixels, using the largest tile size will produce better results.

Pie Chart Properties dialog

The Pie Chart Properties dialog appears when a Pie Chart is right-clicked and the **Chart...** option is clicked. It determines how the Pie Chart is displayed when generating output and in Preview mode (see "Business graphics" on page 827).

General tab

Settings on the General tab correspond to properties of the `AmPieChart` class in the amCharts library; see: AmPieChart.

- **Text Group**: Determines how text is displayed in labels and legends.
  - **Font**: Type in the font-face to use to display text. The font must be installed on the system and defaults to Verdana if the font is not found.
    (Equivalent to the `fontFamily` property; see: fontFamily.)
  - **Size**: Type in the size of the font. Defaults to 11.
    (Equivalent to the `fontSize` property; see: fontSize.)

- **Color**: Type the color in which to display text: a valid hexademical color code (HTML Hex Color) or a valid color name (see MDN Web Docs - named color), or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
  (Equivalent to the `color` property; see: color.)

- **Slice Colors Group**

  Note: Settings made in this group override the settings made in the Chart Wizard (see "Selecting data for a Business Graphic" on page 830).

  - **Base Color**: Enter a base color: a valid HTML hexadecimal color code (HTML Hex Color) or a valid color name (see MDN Web Docs - named color), or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
    When a Base Color is set, it will be the color of the first slice and the colors of all other slices are based on this color.
    (Equivalent to the `baseColor` property; see: baseColor.)
    To set a color for each data field individually, leave this option empty and edit the **script** that fills the Pie Chart (see "Selecting data for a Business Graphic" on page 830).
  - **Brightness Step**: Enter the amount of brightness to change on each new slice. Positive values increase the brightness (max: 100), negative values decrease the brightness (minimum: -100). Default is 10.
    (Equivalent to the `brightnessStep` property; see: brightnessStep.)

- **Slice Outline Group**: Determines whether an outline should be added to each slice of the chart.
  - **Width**: Select the width of the outline for each pie slice.
    (Equivalent to the `outlineThickness` property; see: outlineThickness.)
  - **Color**: Enter a color for the outline: a hexadecimal color code (HTML Hex Color) or a valid color name (see MDN Web Docs - named color), or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
    (Equivalent to the `outlineColor` property; see: outlineColor.)
  - **Opacity**: Enter the opacity of the outline. 100 is fully opaque, 0 is transparent.
    (Equivalent to the `outlineAlpha` property; see: outlineAlpha.)

Pie tab

Settings on the Pie tab correspond to properties of the `AmPieChart` class in the amCharts library; see: AmPieChart.

- **Pie Group**: Defines how the pie chart is displayed in the template.
  - **Radius**: Enter the radius of the Pie Chart as a percentage of the shortest length of the containing object (e.g. 30%) or in pixels.
    (Equivalent to the `radius` property; see: radius.)
  - **Hole Radius**: Enter the radius of the center of the Pie Chart to remove, in pixels or as a percentage. The hole radius removes the center of the chart, creating a doughnut hole pie chart.
    (Equivalent to the `innerRadius` property; see: innerRadius.)
  - **Start Angle**: Enter the starting angle of the first slice of the chart, between 0 and 360 (decimals are allowed). This essentially rotates the Pie Chart. Note that if a 3D effect is added to the chart, the only accepted values are 90 or 270 degrees.
    (Equivalent to the `startAngle` property; see: startAngle.)
- **3D Group**: Creates a 3D effect if both settings in this group are higher than 0.
  - **Depth**: The depth of the Pie.
    (Equivalent to the `depth3D` property; see: depth3D.)
  - **Angle**: The Pie lean angle. This must be a value between 0 - 90.
    (Equivalent to the `angle` property; see: angle.)

Labels tab

Settings on the Label tab correspond to properties of the `AmPieChart` class in the amCharts library; see: AmPieChart.

- **Labels Group**: Defines how the label text is shown.
  - **Text**: Enter the text to use to display labels. The default text is [[title]]: [[percents]]%. Variables can be used to display specific data, <br> can be used to create a new line:
    - [[title]] : Refers to the field label.
    - [[percents]] : Contains the percentage of the Pie chart the value represents.
    - [[value]] : Contains the numerical value of the field.
    - Any text: Adding text (such as a dollar sign or column, etc) will make it appear in each label.

    (Equivalent to the `labelText` property; see: labelText.)

- **Radius**: The distance between the label and the slice, in pixels. You can use negative values to put the label on the slice.
  (Equivalent to the `labelRadius` property; see: labelRadius.)

- **Tick Group**: Defines how ticks (line between the Pie chart and its labels) is shown.
  - **Color**: Enter a valid HTML hexadecimal color code (HTML Hex Color) or a valid color name (see MDN Web Docs - named color), for the color of the tick.
    (Equivalent to the `labelTickColor` property; see: labelTickColor.)
  - **Opacity**: Enter a percentage of opacity for the tick to be displayed. The default is 20 (20% opacity). 100 is fully opaque, 0 is transparent.
    (Equivalent to the `labelTickAlpha` property; see: labelTickAlpha.)

- **Grouping Group**: Defines how smaller percentages are grouped together into an individual "Other" category.
  - **Less than %**: If there is more than one slice whose percentage of the pie is less than this number, those slices will be grouped together into one slice. This is the "other" slice. It will always be the last slice in a pie.
    (Equivalent to the `groupPercent` property; see: groupPercent.)
  - **Slice Title**: Enter a name for the label of the "Other" category.
    (Equivalent to the `groupedTitle` property; see: groupedTitle.)
  - **Color**: Enter a color for the slice: a hexadecimal color code (HTML Hex Color) or a valid color name (see MDN Web Docs - named color), or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
    (Equivalent to the `groupedColor` property; see: groupedColor.)

Legend tab

Settings on the Legend tab correspond to properties of the AmLegend class in the amCharts library; see: AmLegend.

- **Show Legend**: Specifies if legend is enabled or not.
  (Equivalent to the `enabled` property; see: enabled.)

- **Legend Group**: Defines how the legends are shown.
  - **Equal label widths**: Check so that all labels are of equal width in the Legends box. The Legend's width will accommodate the largest value.
    (Equivalent to the `equalWidths` property; see: equalWidths.)
  - **Position**: Use the drop-down to select where the legend is shown: at the Right, Left, Top or Bottom.
    (Equivalent to the `position` property; see: position.)

- **Align**: Use the drop-down to select how to align the text in the labels: Left, Middle or Right.
  (Equivalent to the `align` property; see: align.)
- **Horizontal Space**: Horizontal space between legend items, in pixels.
  (Equivalent to the `spacing`property; see: spacing.)
- **Vertical Space**: Enter a numerical value (in pixels) to define the vertical space between legend items, and also between the legend border and the first and last legend item.
  (Equivalent to the `verticalGap` property; see: verticalGap.)
- **Max Columns**: Enter a numerical value to define the maximum number of columns in the legend. If the Legend's position is set to "right" or "left", this is automatically set to 1.
  (Equivalent to the `maxColumns` property; see: maxColumns.)

- **Labels Group**: Defines if and how labels are shown in the Legend.
  - **Text**: Enter the text used to display the labels; `[[title]]` is a variable that will be replaced with the title of the graph.
    (Equivalent to the `labelText`property; see: labelText.)

- **Markers Group**: Defines how the Legend's Markers look. Markers are icons with a color matching the Legend with its corresponding line.
  - **Type**: Use the drop-down to select in which shape the Markers are displayed; "none" hides the Markers completely.
    (Equivalent to the `markerType` property; see: markerType.)
  - **Size**: Enter the size (in pixels) for the Markers to be displayed.
    (Equivalent to the `markerSize` property; see: markerSize.)
  - **Label Gap**: Enter the distance (in pixels) between the legend marker and legend text.
    (Equivalent to the `markerLabelGap` property; see: markerLabelGap.)
  - **Border Width**: Use the drop-down to define the thickness of the border added to the Markers. The default value (0) means the line will be a "hairline" (1 px). In case the Marker type is line, this style will be used for the line thickness.
    (Equivalent to the `markerBorderThickness` property; see: markerBorderThickness.)
  - **Border Color**: Color of the Legend's border. Enter a hexademical color code (see Hex Calculator) or a valid color name (see MDN Web Docs - named color), or

click the colored square to open the Color Picker dialog ("Color Picker" on page 1143).
(Equivalent to the `markerBorderColor` property; see: markerBorderColor.)

- **Border Opacity**: Enter a numerical value between 0 and 100 to define the opacity (in percentage) of the border.
(Equivalent to the `markerBorderAlpha` property; see: markerBorderAlpha. When specified on the Source tab, the value should be between 0 and 1, e.g. 0.8.)

Source tab

The JSON on the Source tab reflects the choices made in the other tabs and, more importantly, provides the possibility to add in any amCharts configuration option that is unavailable via the other tab menus. For more information see: "Adding and editing properties manually" on page 832.

Profile Scripts dialog

The Script Profiler is accessible through the **Context > Profile Scripts** menu option. It runs the scripts in the template, using the current record, in order to verify the speed at which scripts in the "Scripts pane" on page 1256 execute. It helps greatly in troubleshooting performance issues caused by scripting (see also: "Testing scripts" on page 1074).

When the dialog opens, the script profiler runs automatically, on 1000 instances of all the scripts by default (this can be changed through the "Scripting preferences" on page 1059).

The script profiler can take a while, so please be patient.

The results are shown as follows (the first in the line is indicated as **Total** and represents the totals of all the scripts underneath, representing a good overview of the scripts performance in the template):

- **Name**: The name of the script being executed.
- **Count**: As the profiler runs, Count shows the current number of iterations that have been run. This goes up to the total number of set instances and then stops. Hover with your mouse to display a tooltip indicating in which sections the scripts has run (and in which contexts).
- **Elapsed**: Displays the total elapsed time since the start of the session. The table entries are initially sorted based on the values in this column, from high to low. Hovering the mouse over it will display a tooltip that indicates the breakdown of the execution time across different execution stages.
- **Delta**: Displays the estimated difference in performance between the current session and the previous session. Uses average values, so should still work if the previous

session was stopped after a different number of iterations. Will be empty if no previous data is available. Hover with your mouse to display a tooltip indicating the breakdown of the execution time across different execution stages.

Rasterization Options

Before creating Email or Web output, you may want to **rasterize** certain elements, such as business graphics, to ensure that the output will be visible to most viewers. Rasterizing an element means converting it into a JPG or PNG image. For a JPG image you can set the quality of the resulting image in a percentage.

Note: Rasterization options are only available for Boxes (elements); see "Boxes" on page 823.

Note: A business graphic in an Email section is rasterized by default and output as PNG image, because email clients usually don't support SVG images.
SVG images in an Email section give an error in the Preflight window (see "Doing a Preflight" on page 1075).

See also:

- "Before generating Email output" on page 1674
- "Before generating Web output" on page 1683

Script Debugger

The Script Debugger allows you to test personalization scripts by setting breakpoints and stepping through the scripts. (See also: "Testing scripts" on page 1074.)

There are two ways to start the Script Debugger:

- Click the **Debug Scripts** button in the toolbar of the Scripts pane. The Script Debugger will pause and accept input as soon as it processes the first script.
- Right-click an enabled script in the Scripts panel and choose **Debug** from the contextual menu. The Script Debugger will pause and accept input as soon as it processes the selected script.

The Debugger simulates an output run with only the current record and appears as soon as the merge engine processes the selected script, or the first script if no script was selected. You can then add breakpoints and/or step through the code.
Meanwhile, the Workspace will display the partially merged document.

Scripts and call stack

The left side of the window displays all enabled scripts that apply to the current context and the name of the script that is displayed in the source editor at the top right and the current line number.

If the selector of a standard script or post pagination script has no matches in the current resource (Master Page or section) the script will be skipped.

If an error is thrown, the rest of the current script is skipped and execution continues with the next script.

In the call stack, frames are ordered from new to old, top to bottom. The current stack frame has an arrow indicator. You will only see multiple stack frames if execution is suspended inside a function.
You can click a frame to make it the current frame. The Variables panel always reflects the state of the current frame. Note that this does not move the instruction pointer. The script remains suspended at the most recent frame, even if that frame is not the current frame.

Source code

You can click any script at the left to view it in the source editor at the top right. An instruction pointer in the left margin marks the current line.
The code is read-only, but clicking to the left of a line of code adds a breakpoint to it (except when a line has no debuggable content, for example if it only contains a declaration or comment). Note that breakpoints are discarded when the dialog is closed.

Hovering over a variable in the code shows the value of that variable. If the variable is nested, like fields in record.fields, you can click on the variable, its parent, child elements and/or siblings (if any) to see their value.
The value is read-only, but you can select it and copy it by pressing **Ctrl+C**.

Find code

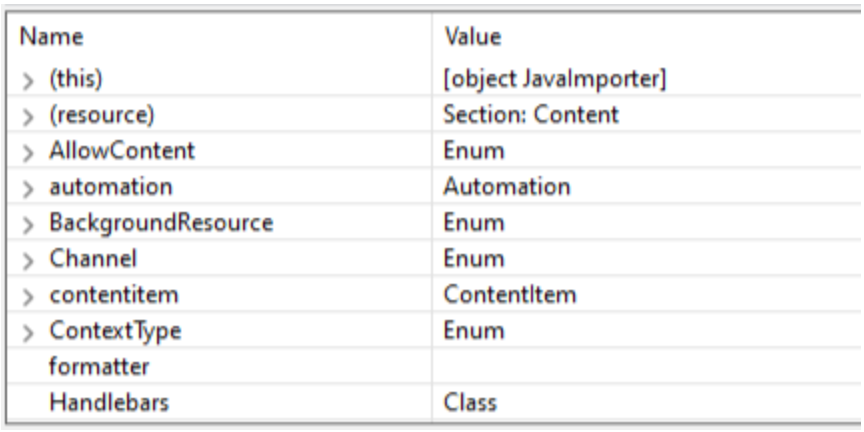
To search for a text in all of the scripts, starting with the currently selected script, press **Ctrl+F** and start typing the text. While the Find field is visible, you can either use the arrow buttons next to it, or press **Enter** / **Shift+Enter** to go to the next or previous match, respectively.

When the Find field is not visible, you can press **F3** / **Shift+F3** to go to the next or previous code that matches your last search text.

> **Tip:** When you click on an identifier (such as a variable name) in a script, the overview ruler at the right highlights all other occurrences of that identifier straightaway.

Variables

At the bottom right the Script Debugger shows a hierarchical overview of all variables in the scope chain and their state.



Name	Value
(this)	[object JavaImporter]
(resource)	Section: Content
AllowContent	Enum
automation	Automation
BackgroundResource	Enum
Channel	Enum
contentitem	ContentItem
ContextType	Enum
formatter	
Handlebars	Class

This information is always relative to the current stack frame, which is the one that's selected on the left.

The overview includes the special entry "(this)", which represents the JavaScript "this" object.

If the current script is not a Control Script, the overview also includes the special entry "(resource)". When expanded, this entry shows the HTML of the current Master Page or section, allowing you to see exactly how each line in the script affects the DOM.

Use the **Copy** button (or press Ctrl+C) to copy the label and contents of the selected variable and all of its children to the clipboard.

Expressions

You can add custom expressions. This is especially useful when you want to monitor the value of a variable that is situated at a lower level in the hierarchy. For example, if you'd want to directly see the height of the top margin of a section as you step through the code, you could add the following expression: `merge.section.margins.top`.
Expressions can include function calls. For example, you could monitor the height of an element that has the ID my_div with the expression: `query("#my_div").height()`. Note that the query in this example will be limited to the current resource (Master Page or section).
You could also change the value of a variable through an expression to see how that affects the document. Let's suppose execution is suspended before an if-statement: `if (foo == 3)`. If foo is not equal to 3, but you'd still like to see what would happen if it were, you could add the expression `foo = 3`.
Expressions are evaluated automatically when execution is suspended.

- The **Add** button lets you add an expression to the overview. Expressions must be unique and cannot be empty; empty expressions will be removed.
- To **edit** an existing expression, just click the entry.
- The **Delete** button removes the selected expression.

Step-through buttons

The buttons at the bottom of the Script Debugger let you step through the code. When you step through the code it always steps relative to the most recent stack frame.

- **Into** (F5): Step into the function on the current line, if possible. A common use case for this is to step into a `results.each()` call. Note that you can only step into functions that are defined in user scripts. For other statements this behaves like (Step) Over.
- **Over** (F6): Step over the current line.
- **Return**: Step out of the current function. This only works if execution is suspended inside a function defined in one of the user scripts. Otherwise it behaves like (Step) Over.
- **Resume** (F8): Continue up to the next breakpoint or until the output run is finished.
- **Restart**: Start the debugging session over from the beginning. Execution will be suspended on the first line of the output run, even if that line does not have a breakpoint.
- **Close**: Close the Script Debugger.

Sample Projects

A Sample Project generates a small Connect solution that is ready to be tested and deployed. The solution contains a specific Workflow configuration, as well as the Connect templates, data mapping configurations, and any Job Presets and Output Presets that are used in that configuration.

This chapter describes the Sample Projects and the projects that they install. It will help you install, comprehend and customize the projects.

For generic information about Connect solutions and their components, see "OL Connect projects" on page 250.

> **Note:** The projects require that the Connect Server and Connect Workflow be installed on the local machine.

These are the projects that can be installed with a wizard.

- **Print promotional jobs**. This project generates a batch of basic personalized letters, in the format that was selected in the wizard: PDF, PCL or PostScript Level 3. (See: "Sample Project: Print Promotional Jobs" on page 279.)
  The Workflow configuration uses the All In One plugin (see "Print processes with OL Connect tasks" on page 306).
- **Print transactional jobs**. This project creates print content items - invoices - and uses multiple Create Output tasks to generate various print output variants:
  - A single PDF for the entire job (in which the invoices are grouped per customer).
  - One PDF per customer.
  - One PDF per invoice.

  (See: "Sample Project: Print Transactional Jobs" on page 285.)

  The Workflow process implements the typical Print plugins (see "Print processes with OL Connect tasks" on page 306).
- **Basic web page**. This project serves a simple web page, personalized via URL parameters. (See: "Sample Project: Serving a Web Page" on page 296.)
- **Submitting data with webforms**. This project illustrates how to serve multiple web pages, one of them a web form, using a single Workflow process. It stores submitted data in the Data Repository. (See "Sample Project: Submitting Data with Web Forms" on page 291.)
- **Basic email setup**. This project sends email messages with attachments. Pre-rendered PDFs are attached based on the provided sample data. (See "Sample Project: Basic Email" on page 265.)
- **Capture OnTheGo Timesheets**. This project deploys and serves COTG Timesheet forms. Additional processes capture the submitted data and output them in JSON, XML and PDF. In order to create the PDF, the data are merged with a Connect template. (See: "Sample Project: COTG Timesheets" on page 272.)

Tip: All projects created with the Sample Projects wizard are *local versioned projects*. Any time you *commit* files in the project, a snapshot of the project is saved as a *version*. See "Versioned projects" on page 253 for more information about this feature.

Basic Email - Sample Project

The Basic Email Sample Project generates a Connect solution that can send **emails**.
To start the Sample Project, select **File > New > Sample Projects > Basic Email** from the menu.

For more detailed information about this project, and instructions on how to test and customize it, see "Sample Project: Basic Email" on page 265.

The wizard contains the following options.

- **Workflow**
  - **Project folder**: Select a folder. In this folder, the Project Wizard will create two folders: Configurations and Workspace. The project's resource files are saved to the Configurations folder. The Workspace folder is meant to be used when debugging the solution (see "Testing and running the project" on page 266).
  - **Send Templates and resources to Workflow**: Check this option to send the necessary resource files to the active Workflow instance on the local machine. Without these files the Workflow configuration can be inspected, but it cannot be tested; in order to test the configuration, you'll have to send the resources manually (see "Sending files to Workflow" on page 593).

    > **Note:** In order for this option to work, Workflow and the Designer must be installed on the same machine. In addition, the PPMessenger service must be running. By default, this service is always started automatically from the moment Workflow has been installed.

- **Email Info**
  - **Sender Address**: The email address entered here will be used as the sender as well as the recipient of all emails.
  - **Mail Host**: The address of the SMTP server through which the emails should be routed. This may include a port number, e.g. :25. The default port is 25. Use 587 for TLS.
  - **Use Encryption (TLS)**: Check this option to connect to the SMTP server using TLS (Transport Layer Security, also called SSL). The TLS option should be enabled if your SMTP server requires it.
  - **Use Authentication**: Check to enable authentication on the SMTP server.
    - **Username**: Enter a user name that has permission to send email through the SMTP server.
    - **Password**: Enter the password for the above user name.

COTG Timesheets - Sample Project

The COTG Timesheets Sample Project generates an entire Capture OnTheGo solution.
To start the Sample Project, select **File > New > Sample Projects > COTG Timesheets** from

the menu.
For more detailed information about this project, and instructions on how to test and customize it, see "Sample Project: COTG Timesheets" on page 272.

The wizard contains the following options.

- **Workflow**
  - **Project folder**: Select a folder. In this folder, the wizard will create two folders: Configurations and Workspace. The project's resource files are saved to the Configurations folder. The Workspace folder is meant to be used when debugging the solution (see "Testing the project" on page 273).
  - **Host**: Enter the address of the host running the Workflow server. This address will be used by the COTG app to download forms and submit data to the OL Connect Workflow service.
    The given IP address is that of the machine on which the Sample Project (i.e. the Designer) runs. The port number is 9090 since the project's Workflow configuration uses the NodeJS Server Input task, and Workflow's NodeJS Server listens on port 9090 by default.
    As long as all components (the COTG app, Workflow and Designer) run on the same computer you could use `http://localhost:9090`.
    If the COTG app runs on a mobile device, make sure to enter an IP address that can be reached from outside your network. Also be aware that a computer can have multiple network cards, each with their own IP address or addresses, or that it may have only an IP version 6 address, for example.

    Note: Your network setup may make it impossible for the COTG app to communicate with the OL Connect Workflow service in order to download forms and submit data. Network and firewall settings may block these requests. Please contact your local IT department for advise on how this can be resolved in your setup.

    Note: On Windows 10, built-in restrictions prevent applications from sending data to the same computer, making it impossible for the COTG app to submit form data to Workflow on the same computer. The COTG user guide explains how to change these restrictions; see Capture OnTheGo Help Center.

- **Send Templates and resources to Workflow**: Check this option to send the necessary resource files to the active Workflow instance on the local machine. Without these files the Workflow configuration can be inspected, but it cannot be tested; in order to test the configuration, you'll have to send the resources manually (see "Sending files to Workflow" on page 593).

  Note: Workflow has to be installed on the host in order to use this option. In addition, the PPMessenger service must be running. By default, this service is always started automatically from the moment Workflow has been installed.

- **Capture OnTheGo**
  - **Repository ID**: Enter the ID of your Capture OnTheGo Server repository.
  - **Password**: Enter the password for your Capture OnTheGo Server repository.

    Tip: If you own OL Connect, free COTG trial licenses may be available to you; see http://www.captureonthego.com/en/promotion/.

  - **COTG Account**: Enter the email address that will be used in the COTG app to log in. The forms - the time sheets - will be sent to this account.

Promotional Print Jobs - Sample Project

The Print Promotional Jobs Sample Project generates a simple Connect solution that can produce **promotional print output**.
To start the Sample Project, select **File > New > Sample Projects > Print Promotional Jobs** from the menu.
For more detailed information about this project, and instructions on how to test and customize it, see "Sample Project: Print Promotional Jobs" on page 279.

The wizard contains the following options.

- **Workflow**
  - **Project folder**: Select a folder. In this folder, the Project Wizard will create two folders: Configurations and Workspace. The project's resource files are saved to the Configurations folder. The Workspace folder is meant to be used when debugging the solution (see "Promotional Print Jobs - Sample Project" above).
  - **Send Templates and resources to Workflow**: Check this option to send the necessary resource files to the active Workflow instance on the local machine. Without these files the Workflow configuration can be inspected, but it cannot be tested;

in order to test the configuration, you'll have to send the resources manually (see "Sending files to Workflow" on page 593).

> **Note:** In order for this option to work, Workflow and the Designer must be installed on the same machine. In addition, the PPMessenger service must be running. By default, this service is always started automatically from the moment Workflow has been installed.

- **Production**
  - **Output preset**: Select the output format: **PDF**, **PCL** or **PostScript Level 3**. The variants with **Virtual Stationery** will add the stationery file (stored in the Connect template) as background image to the pages. In any case, the output will be printed to a file.
    The Sample Project will set the respective Output Preset for the Output Creation tab in the All In One plugin, and it adds the correct file extension (.pdf, .pcl or .ps) to the File name setting in the Send To Folder task.

Transactional Print Jobs - Sample Project

The Print Transactional Jobs Sample Project generates a Connect solution that can produce **transactional print output**; in other words, it lets you print invoices.
For more detailed information about this solution and instructions on how to test and customize it, see "Sample Project: Print Transactional Jobs" on page 285.

To start the Sample Project, select **File > New > Sample Projects > Print Transactional Jobs** from the menu.

It contains the following options.

- **Workflow**
  - **Project folder**: Select a folder. In this folder, the Project Wizard will create two folders: Configurations and Workspace. The project's resource files are saved to the Configurations folder. The Workspace folder is meant to be used when debugging the solution (see "Transactional Print Jobs - Sample Project" above).
  - **Send Templates and resources to Workflow**: Check this option to send the necessary resource files to the active Workflow instance on the local machine. Without these files the Workflow configuration can be inspected, but it cannot be tested; in order to test the configuration, you'll have to send the resources manually (see "Sending files to Workflow" on page 593).

> **Note:** In order for this option to work, Workflow and the Designer must be installed on the same machine. In addition, the PPMessenger service must be running. By default, this service is always started automatically from the moment Workflow has been installed.

Submitting Data with Web Forms - Sample Project

The Serving a Web Page Sample Project generates a Connect solution that can serve **a web page**.
To start the Sample Project, select **File > New > Sample Projects > Submitting Data with Web Forms** from the menu.
For more detailed information about this project, and instructions on how to test and customize it, see "Sample Project: Submitting Data with Web Forms" on page 291.

The wizard contains the following options.

- **Workflow**
  - **Project folder**: Select a folder. In this folder, the Project Wizard will create two folders: Configurations and Workspace. The project's resource files are saved to the Configurations folder. The Workspace folder is meant to be used when debugging the solution (see "Submitting Data with Web Forms - Sample Project" above).
  - **Send Templates and resources to Workflow**: Check this option to send the necessary resource files to the active Workflow instance on the local machine. Without these files the Workflow configuration can be inspected, but it cannot be tested; in order to test the configuration, you'll have to send the resources manually (see "Sending files to Workflow" on page 593).

    > **Note:** In order for this option to work, Workflow and the Designer must be installed on the same machine. In addition, the PPMessenger service must be running. By default, this service is always started automatically from the moment Workflow has been installed.

Serving a Web Page - Sample Project

The Serving a Web Page Sample Project generates a Connect solution that can serve **a web page**.
To start the Sample Project, select **File > New > Sample Projects > Serving a Web Page** from the menu.

For more detailed information about this project, and instructions on how to test and customize it, see "Sample Project: Serving a Web Page" on page 296.

The wizard contains the following options.

- **Workflow**
  - **Project folder**: Select a folder. In this folder, the Project Wizard will create two folders: Configurations and Workspace. The project's resource files are saved to the Configurations folder. The Workspace folder is meant to be used when debugging the solution (see "Serving a Web Page - Sample Project" on the previous page).
  - **Send Templates and resources to Workflow**: Check this option to send the necessary resource files to the active Workflow instance on the local machine. Without these files the Workflow configuration can be inspected, but it cannot be tested; in order to test the configuration, you'll have to send the resources manually (see "Sending files to Workflow" on page 593).

    Note: In order for this option to work, Workflow and the Designer must be installed on the same machine. In addition, the PPMessenger service must be running. By default, this service is always started automatically from the moment Workflow has been installed.

Script wizards

Wizard types

Script wizards are simplified interfaces for common scripts in templates. These are the available Script wizard types.

- **Text Script**: This type of script inserts variable data in the text. See "Variable data in text: scripts and placeholders" on page 945.
- **Dynamic Image Script**: Provided that its selector refers to an image, this script dynamically changes the image for each record. See "Dynamic images" on page 972.
- **Dynamic Background Script**: This type of script dynamically swaps a Print section's background. See "Dynamic Print section backgrounds" on page 994.
- **Email Scripts**. The Email To Script is automatically added to any new Email context; it defines where the email should be sent for each record. Other Email scripts define other recipients, the subject of the email that is sent, and the PDF password. See "Email header settings" on page 672.

- **Barcode Script**: This script controls the contents of a Barcode. It is automatically added when a barcode is added to a template. See "Barcode" on page 768
- **Business Graphic Script**: This script controls the contents of a Pie Chart, Bar Chart or Line Chart. See "Business graphics" on page 827.
- **Conditional script**: This script shows or hides content elements depending on certain conditions and values. See "Showing content conditionally" on page 958.

The result of the script can be either text appearing on the page, an email address or subject, the barcode data, or a JSON string that is written to the attribute of an HTML element.

Options

For the options in Conditional Script wizards see "Conditional Content script dialog" on page 1197.

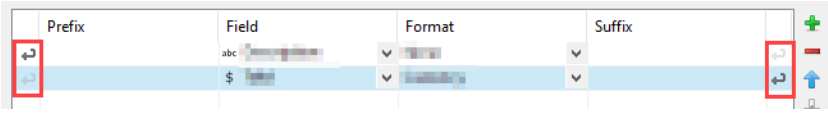
In other Script wizards, the following options are available.

- **Name**: The name of the script, making it easier to identify it.
- **Selector/Find text**: The Selector or Text to apply the result of the script to.
  - **Selector**: Uses CSS selectors to find the element to which the script applies
  - **Text**: Uses text as a trigger for the script. The script applies to all instances of the text found in the template.
  - **Selector and Text**: Uses text as a trigger for the script but only applies to text within the specified Selector.

  Note: Email script wizards don't have the Selector or Find text option. Instead they select the header of the specified **Section** (the current one, by default) or **All email sections**. Note that the Section name is case-sensitive.To see the actual (HTML) selector that an email script uses to select a section, you can click the Expand button.
- **Wizard Results**: Displays a list of the data that replaces the content that matches the script's selector:
  - **Prefix**: Static text to use before the set field. For example in Dynamic Image scripts, the default prefix is **images/**.
  - **Field**: A drop-down to select which field contents to use in the script. The field should contain a valid value. For an email script, for example, the field would

have to contain an email address. Note that you can't select a field that belongs to a detail table.

- **Format**: A special formatting modifier applied to the Field; see "Formatting variable data" on page 952.
- **Suffix**: Static text to use after the set field. For Dynamic Image Scripts, the default suffix is **.jpg**and refers to the file extension.
- In the Text Script Wizard and Helper Wizard, to add one **line break**, activate the Line break button in the last column.



- **[+]**: Adds a new line to the Wizard Results. Note that by default there is no line return between fields in the list. Adding <br> in the Suffix or Prefix field can establish a line return.
- **[-]**: Removes the currently selected line in the Wizard Results list.
- **Arrow Up**: Moves the currently selected line up one position.
- **Arrow Down**: Moves the currently selected line down one position.
- **Refresh Types**: Updates field types to match the types in the active data model.
- **Options** (only available in the Text Script wizard and the Dynamic Image wizard): specifies where and how the script inserts its results:
  - As **HTML**. HTML elements in the results are processed and displayed as HTML elements. For instance, `<b>this is bold</b>` will be displayed as **this is bold**. This is the default setting.
  - As **text**. This inserts the results as-is, meaning HTML tags and elements are displayed as text in the output. In this scenario, "<br>" shows up in the text and does not insert a line break.
  - As the value of an **attribute** of an HTML element. The selector of the script should be an HTML element. Which attributes are available depends on the selected HTML element. If the script's selector is an image (<img> element) for example, and the attribute is `src`, the script will modify the image's source. The script's results should be a valid value for the chosen attribute.
  - When checked, the option **Convert fields to JSON string** writes the results from

the script into an attribute or text as a JSON string. This is useful for Web contexts where a front-end script can read this value easily.

- **OK**: Click to save any changes made to the script, apply the changes in the template, and close the dialog.
- **Cancel**: Click to close the dialog without saving changes.
- **Expand**: Click to convert the script generator to a regular script. Note that this action is not reversible once the regular script has been saved. The expanded script will use either double or single quotation marks, depending on the Preferences (**Window > Preferences > Scripting**; see "Scripting preferences" on page 1059).
- **Apply**: Saves changes made to the script and applies the changes in the template without closing the dialog.

Expanded Script window

When expanded, the Script window replaces all parts of the wizard below the Selector by a box in which the script can be typed. See "Writing your own scripts" on page 1064.

For an overview of keyboard shurtcuts that can be used in this script editor, see "Keyboard shortcuts" on page 1223.

Chart Script dialog

The Chart Script dialog specifies which variable data will be displayed in a chart and shows a preview of both the chart (on the right hand side) and the data (at the bottom). This dialog is one of the parts of the Insert Chart wizards. It can be reopened by double-clicking the relevant script in the Scripts pane.

For more information about charts, see: "Business graphics" on page 827.

These are the options in the Chart Script dialog:

- **Name**: The name of the script, making it easier to identify it.
- **Selector**: The Selector or Text to apply the script to.
  - **Selector**: Uses CSS selectors to find the element to which the script applies. (See also: "Selectors in OL Connect" on page 1083.)
  - **Text**: Uses text as a trigger for the script. The script applies to all instances of the text found in the template.
  - **Selector and Text**: Uses text within the specified Selector as a trigger for the script.
- **Input Type**: Use the drop-down to select the source of the data to add to the Chart.

    - **Data fields**: Select to use fields at the record level in the Data Model.
    - **Data table**: Select to use fields in a detail table. At least one detail table must be available in the Data Model Pane for this option to be functional.
        - **Detail table**: Use the drop-down to select one of the detail tables that are part of the Data Model.
        - **Category**: Use the drop-down list to select the data field (in the selected detail table) of which the values will appear under the bars or the line; in other words, on the **x** axis.
- **Values**: Select data fields with a numerical value.
    - **Column**: Removing the topmost check deselects all data fields.
    - **Labels**: Click the label to enter a new label for a field. It depends on the Input Type where the labels appear: with Data Fields, the labels appear under the x-axis of the chart; with a Data Table, labels appear in the Legend.
    - **Color**: Click on the color to open the Color Picker dialog ("Color Picker" on page 1143) and select a different color.

  Buttons next to Values:

    - The **Toggle non-numeric fields** button filters non-numeric fields from the list. The list will then display only Integer, Float and Currency data fields.
    - The **Move Up** and **Move Down** buttons change the order of the fields, which is reflected in the chart.
- **OK**: Click to save any changes made to the script, apply the changes in the template, and close the dialog.
- **Cancel**: Click to close the dialog without saving changes.
- **Expand**: Click to convert the script generator to a regular script. Note that this action is not reversible once the regular script has been saved.
  The expanded script will use either double or single quotation marks, depending on the Preferences (**Window > Preferences > Scripting**; see "Scripting preferences" on page 1059).
- **Apply**: Saves changes made to the script and applies the changes in the template without closing the dialog. Note that the effect will only be visible on the Preview tab in the Workspace.

Expanded Script window

When expanded, the Script window replaces all parts of the wizard below the Selector by a box in which the script can be typed. See "Writing your own scripts" on page 1064.

For an overview of keyboard shortcuts that can be used in this script editor, see "Keyboard shortcuts" on page 1223.

Conditional Content script dialog

Conditional Content scripts can show or hide elements depending on certain conditions and values. They can be added by right-clicking any element in a template and clicking **Make Conditional**. If the current element has neither an ID nor a class, an ID will be automatically generated. See "Showing content conditionally" on page 958.

The general options in the Conditional Script wizard are:

- **Name**: The name of the script, making it easier to identify it.
- **Selector**: The Selector or Text to apply the result of the script to.
  - **Selector**: Uses CSS selectors to find the element to which the script applies.
  - **Text**: Uses text as a trigger for the script. The script applies to all instances of the text found in the template.
  - **Selector and Text**: Uses the specified selector as a trigger for the script. The script applies to all instances of the text found within the specified Selector.

  For more information about Selectors see "Selectors in OL Connect" on page 1083.
- **Action**: Use the drop-down to select whether to **Show** or **Hide** the element when the condition below is true. If, conversely, the condition evaluates to false, the opposite action will be performed.

A **condition** is made up of groups and rules.

The ✚ **Add** button adds a rule that evaluates a **data field** to a group.
To add either a **group**, or a rule that evaluates a **runtime parameter**, click the downward pointing arrow next to this button and select **Group** or **Parameter Rule**.

- **Group**:

  A group consists of one or more rules with a logic operator. Four logic choices are available at the Group level. These are:

  - **All of the following**.
    This equates to the logical operator (... AND ...).

If *all* of the associated criteria are met, then this group resolves to TRUE.

- **Any of the following**.
  This equates to the logical operator (... OR ...).
  If *any* of the associated criteria are met, then this group resolves to TRUE.
- **Not all of the following**.
  This equates to the logical operator (NOT(... AND ...)).
  If *any (but not all)* of the associated criteria are met, then this group resolves to TRUE.
- **Not any of the following**.
  This equates to the logical operator (NOT (... OR ...)).
  If *none* of the associated criteria are met, then this group resolves to TRUE.

- **Rule**:
  - **Data Field** / **Parameter**: Use the drop-down to select which data field in the record, or which runtime parameter in the template, the condition will be based upon. (See "Runtime parameters" on page 608.)
  - **Operator**: Select which kind of comparison is applied. The options available will depend upon the Data Field or Parameter type (they are "type aware"), but most have at least "*is equal to*" and "*is not equal to*" as an option.
  - **Value**: The value(s) used for the conditional check.
    - **String** data field specific selections are whether the alphanumeric string "*is empty*" or "*is not empty*", or "*contains*", "*does not contain*", "*starts with*" or "*ends with*" another string value.
      Values are **case sensitive** by default. You can click the button next to the value to make them case insensitive, which means that upper- and lowercase letters are treated as being the same.
    - **Date** data field specific selections are whether the date field is "*before*" or "*after*" another date entry.
      Dates should be entered in ISO standard notation (yyyy-mm-dd) (see ISO 8601). It is best to select the date using the date selection option.
    - **Boolean** data field specific selections are whether the selected Boolean data field "*is true*" or "*is false*".
    - **Numeric** data field specific selections: Whether the numeric fields is "*less*

than", "*less than or equal to*", "*greater than*" or "*greater than or equal to*" another number entry.

The **script** is displayed below the condition and updated while the condition is being edited.

Section properties dialogs

The respective section properties dialogs are opened via the contextual menu:

- Right-click the section in the **Resources** pane and choose **Properties**.

Email section properties

For information about Email sections, see: "Email templates" on page 669.

Properties tab

The properties for an Email section are minimal and contain the following options:

- **Name**: Enter the name of the Section in the Email Context. This has no effect on output.
- **CSS mode**: In the Designer, CSS files can be used to style email templates (see "Styling templates with CSS files" on page 881) but eventually, the formatting must be applied in a different way since email clients do not read CSS files. This property determines if and how the styles in linked CSS files are applied to the output.
  - **Write CSS to <head> section**: The content of linked style sheet files is copied to <style> elements in the <head> of the HTML. (This is also known as "embedded CSS".) With this option email gets sent faster; however, email clients may not support <style> elements in the <head>.
    This is the recommended setting, and the default setting for new templates.
  - **Apply CSS properties to elements**: The relevant CSS properties are copied to the <style> attribute of the various HTML elements.
    This is the default setting for templates made with OL Connect versions prior to 2020.1.
  - **Do not inline styles**: No styles are copied from linked CSS style sheets. It is assumed that you format elements locally or use a CSS inliner tool.

Note: It can't be guaranteed that content will look the same in all email clients, particularly when it comes to older client versions, so it is recommended to always test and validate the output. See: "Testing Email output for different email clients" on page 1677.

- **Evaluate Handlebars expressions**: If this option is enabled for a section, dragging data fields to the editor will insert Handlebars expressions instead of placeholders and scripts. See "Variable data in the text" on page 924.
  To prevent issues, the section cannot be edited in Preview mode when this option is enabled.
- The **Meta Information** group lists all <meta> tags that will be added to the header of the email. See "Meta information" on page 677.
- When the option **Append plain-text copy of the HTML** is checked, a plain-text version of the HTML is added to each email that is sent. With new templates this option is checked by default.

Includes tab

This tab lists the style sheets that can be applied to the email section when producing the output. Style sheets are loaded in the order shown, and styles in later style sheets overwrite earlier ones when the same selector is used. (See "Includes dialog" on page 1157 and "Styling templates with CSS files" on page 881.)

Attachments tab

The **Attachments** tab lets you select files and delete attachments. For more information, see: "Email attachments" on page 679.

Print section properties

For information about Print sections, see: "Print sections" on page 629.

Properties

- **Section** group:
  - **Name**: The name of the section.
  - **Minimum number of pages**: The default minimum number of pages is 1. If content is too long to fit on one page, overflow will automatically appear on the next page. Changing the minimum number of pages can be useful when you want certain content to appear on a specific page.
  - **Evaluate Handlebars expressions**: If this option is enabled for a section, Handlebars expressions will be evaluated using the current record in Preview mode and when the output is generated. In Design mode, dragging data fields to the editor will insert Handlebars expressions instead of placeholders and scripts.

See "Variable data in the text" on page 924.
To prevent issues, the section cannot be edited in Preview mode when this option is enabled.

Note: This setting also affects the Master Page with which a Print section is used.

These properties can be used in a Control Script; see the topic "Control Scripts" on page 1098 and the page about the section object: "section" on page 1627.

For an explanation of other settings on the Properties tab, see "Page settings: size, margins and bleed" on page 642.

Includes Tab

The Includes tab defines which style sheets and JavaScript files should be applied to the section when generating output, and in which order; see "Includes dialog" on page 1157.
For more information about stylesheets, see "Styling templates with CSS files" on page 881.
For more information about using JavaScript, see "Using JavaScript" on page 703.

Finishing tab

This tab defines finishing options for this section when it is printed. For an explanation of all Binding and Hole making options, see "Finishing options" on page 1366.

Sheet Configuration tab

The Sheet Configuration tab defines how different Print context sections output on different Media (see "Media" on page 652) and using different Master Pages (see "Master Pages" on page 648). For an explanation of all settings on this tab, see "Sheet Configuration dialog" on page 1208.

Background tab

This tab defines the background of the current Print section; see "Using a PDF file or other image as background" on page 634.

Numbering tab

The Numbering tab defines how page numbers are configured in the current Print section; see "Configuring page numbers" on page 645.

Web Section Properties

The Web section properties define some of the web page properties, especially details appearing in the header.
For information about Web sections, see "Web pages" on page 688.

Properties tab

- **Name, Page title, Shortcut icon**: See "Setting the title, meta data and a shortcut icon" on page 692.
- **Evaluate Handlebars expressions**: If this option is enabled for a section, dragging data fields to the editor will insert Handlebars expressions instead of placeholders and scripts. See "Variable data in the text" on page 924.
  To prevent issues, the section cannot be edited in Preview mode when this option is enabled.

Includes Tab

The Includes tab defines which style sheets and JavaScript files should be included in the section when generating output, and in which order; see "Includes dialog" on page 1157.
For more information about stylesheets, see "Styling templates with CSS files" on page 881.
For more information about using JavaScript, see "Using JavaScript" on page 703.

Arrange Sections

The Arrange dialog is used to change the order of sections within a context. This can have an effect on how they are outputted. See: "Print sections" on page 629, "Email templates" on page 669 and "Web pages" on page 688.

To access the Arrange dialog, on the Resources pane, right-click on any section or the context containing them, and click **Arrange**.

- **Name**: Displays the name of each section within the context.
- **Move Up**: Click to move the currently selected section up one position.
- **Move Down**: Click to move the currently selected section down one position.

Confirm page size change

This dialog is used to confirm a change to Page size in the Print Section Properties.

Select what **Media** settings should be applied, from the drop down list.

The options are dynamic but usually start with **Create new ...** for Media print section

changes. If one or more Print page options already exist, then **Update linked media** or the named Media options will become available.

Send COTG Test

The Send COTG Test dialog is used to send a document to the Capture OnTheGo app without the need to go through the Output to Capture OnTheGo task in OL Connect Workflow (see Workflow Help: Output to Capture OnTheGo).

For more information about Capture OnTheGo see: "Capture OnTheGo" on page 707.
For more information about testing COTG templates, see "Testing a Capture OnTheGo Template" on page 735.

Note that to the contrary of the Output to Capture OnTheGo task in Workflow, when using the Send COTG Test dialog, the *contents* of the file will be sent to the Nu-Book server. The Output to Capture OnTheGo task only sends document meta data to the Nu-Book server. The contents of the file are transmitted (by Workflow) when requested by the COTG app.

A test template automatically appears in the app's Repository for 4 days from the moment it is sent. Once downloaded it remains accessible in the app's Library for 2 days.

> **Note:** The dialog is only available on templates containing a Web context. It does not, however, verify whether any Capture OnTheGo form elements have been added to the page.

The dialog contains the following options:

- **General group**:
  - **Server**: Select the Capture OnTheGo server. COTG servers can be added via the preferences; see "COTG Servers preferences" on page 1060.
  - **Repository ID**: The Nu-Book Repository ID. If you don't have one, you can get a trial account for this purpose; please see this page for more details: http://www.captureonthego.com/en/promotion/.
  - **Password**: The password to the above Nu-Book Store.
  - **Recipient(s)**: The user name(s) that should receive the document. One or more emails and/or user groups, separated by a semicolon.
  - **Category**: The category under which the document appears. You can enter multiple values, separated by a semicolon. If a category does not exist, it will be created on the server.

- **Send as Blank Form**: Check this option to send the template as a reusable form. The form will not be deleted from the app's Library when it is submitted. To manually delete it from the Library, swipe it to the left.

- **Document Information group**:
  - **Title**: The title that appears both on the Nu-Book management interface (https://config-us.captureonthego.com), as well as on the Capture OnTheGo application on the mobile device. Defaults to the name of the template and the currently active section.
  - **Author**: The name of the author or company.
  - **Description**: The title that appears both on the Nu-Book management interface (https://config-us.captureonthego.com), as well as on the Capture OnTheGo application on the mobile device when looking at the document's details.

Send (Test) Email

The Send Email dialog is used to generate mail output and send it to each recipient in the Record Set. To open this dialog, select **File > Send Email**, on the menu.
Note that the subject, recipients etc. must be specified before sending the email; see "Email header settings" on page 672.
For more information about the process of sending out email and the possible settings, see "Generating Email output" on page 1673.

Options for this dialog:

- **From** group:
  - **Name**: Enter the name that should appear when sending emails. The name is optional.
  - **Email**: Enter the email address that will appear as a Sender to the email recipient. A single email address should be written.
- **To** group - only when sending a test email:
  - **Email address(es)**: Enter one or more email addresses where the test emails are sent. Multiple emails can be separated using a semicolon (;) or a comma, and can be in the same format as above. Note that every email address here will receive all the emails for the record-range below.
  - **Use Litmus**: Check to also send the emails to the Litmus test email set in the Email Preferences (to go to the Email Preferences, select **Window > Preferences**,

click the arrow next to **Email**, and then click **General**). Disabled if no Litmus email is set.

- **Records** group:
  - Select **All**, or click **Selection** and enter the range of records that should be sent. Removing the range disables the selection and sends emails to all records in the record set.
- **Attachments**:
  - **Print context as PDF**: If a Print Context exists in the template, its output will be generated and a PDF version of it will be attached to the outgoing email.
  - **Web Page context as HTML**: If a Web Page Context exists in the template, its output will be generated as a single HTML file with all required resources embedded in the file. This HTML file is then added as an attachment to the outgoing email.
- **Outgoing mail settings:**
  - **Presets**: Use the drop-down to select a preset. These presets are configured in the Email (SMTP) preferences; see "Email SMTP settings" on page 674.

    Note: It is recommended to use an Email Service Provider to get access to tools that give you full control over your mailings, like open rates, click through rates etc. See "Using an ESP with OL Connect" on page 1678.
  - **Host**: The SMTP server through which the emails are to be sent. Can be a host (mail.domain.com) or an IP address. The port number can be appended to the host name, separated by a colon, for example: smtp.mandrillapp.com:465. Port 25 will be used if no port number is specified.

    Note: If the mail server supports it, the connection will be encrypted without the need to send the server a STARTTLS instruction when port 465 is used.

    Tip: Gmail only allows Connect to be used as an SMTP client when "Access for less secure apps" is enabled in the Google account settings.
  - **Use authentication**: Check if a user name and password are needed to send emails through the host.

- **Send STARTTLS**: Enabled if authentication is checked. With STARTTLS the client negotiates with the mail server to use some form of encryption, usually a version of Transport Layer Security (TLS). Since this improves security it is recommend to enable this option if you use port 25 (the default port), 2525, or 587.
  Note that the email will not be sent if the SMTP server does not support TLS or SSL (an older encryption type).
  This option is ignored when port 465 is used.
- **User**: Enter the user name used to connect to the SMTP server.
- **Password**: Enter the password for the above user name.

Send to Workflow dialog

The Send to Workflow dialog sends selected templates, data mapping configurations and Print Presets to the OL Connect Workflow server.

- **Configurations**: Initially this list contains the Connect template and data mapping configuration that are currently open, and the Job Preset and Output Preset that were last modified.
  - **Browse**: This button opens the Select Files dialog that lets you add one or more templates, data mapping configurations and Print Presets to the list.

    Note that as of version 2023.1, Print Presets are no longer automatically stored in OL Connect's workspace folder. For more information, see "Creating, opening and saving Print Presets" on page 1645.
  - **Delete**: Deletes the selected file from the list of files to send.
- **Workflow Server**: Use the drop-down to select where to send the files. This lists all the OL Connect Workflow installations detected on the network.

  Note: If Workflow is installed on another computer, Workflow must be configured to accept files coming from another host (via the Access Manager).
  The Messenger service, which is used for inter-module communication, uses ports 5863/5864 by default (see "Firewall/Port considerations" on page 31).
- **Send**: Sends all selected Connect files to the selected Workflow Server. You can use the topmost checkbox to select all of the listed files at once.
- **Cancel**: Click this button to close the dialog without sending files to Workflow.

Select Image dialog

The Select Image dialog lets you select an image, depending on where the image is located. Whenever possible, a preview of the selected image will be shown.

Click **Resources**, **Disk** or **Url**, depending on where the image is located.

- **Resources** lists the images that are present in the **Images** folder on the **Resources** pane.
- **Disk** lists image files that reside in a folder on a hard drive that is accessible from your computer. Click the **Browse** button to select a folder (or an image in a folder).
  As an alternative it is possible to enter the path manually. You can give a local path (e.g. C:\Images\Test.jpg) or use the "file" protocol.

  The complete syntax of a fully qualified URL with the "file" protocol is: `file://<host>/<path>`. Note: if the host is "localhost", it can be omitted, resulting in `file:///<path>`, for example: `file:///c:/resources/images/image.jpg`.
  If the file is located on another server in your network, the path must contain five slashes after "file:".

  > **Note:** Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file://///myserver/myfolder/file.txt).

- **Url** lists image files from a specific web address. Select the protocol (**http** or **https**), and then enter a web address (for example, http://www.mysite.com/images/image.jpg).

> **Note:** If a URL doesn't have a file extension, and the option Save with template is **not** selected, the Select Image dialog automatically adds the `filetype` parameter with the file extension as its value (for example: `?filetype=pdf` (if it is the first parameter) or `&filetype=pdf`).
> The `filetype`, `page` and `nopreview` parameters are not sent to the host; they are used internally. Therefore, URLs that rely on one of these parameters cannot be used.

With an external image, you can check the option **Save with template**. If this option is checked, the file will be inserted in the **Images** folder on the **Resources** pane at the top left. If it isn't saved with the template, the image remains external. Note that external images need to be available when the template is merged with a record set to generate output, and that their location should be accessible from the machine on which the template's output is produced. External images are updated (retrieved) at the time the output is generated.

Insertion Point group box:

Selects where the image is to be inserted.

- **Absolute** checkbox: Select to make the inserted image an Absolutely positioned one. Selecting **Absolute** prevents subsequent selection of **Location** and **Element**.

- **Location** pull down option: Select the Location at which to insert the image.
- **Elements** pull down option: Select the **Element** that the **Location** option is to apply to. Not available for **Location** selection of *At cursor position*.

Sheet Configuration dialog

The Sheet Configuration dialog defines how different Print context sections output on different Media (see "Media" on page 652) and using different Master Pages (see "Master Pages" on page 648).

> **Note:** Master Pages, Media and Duplex printing options can also be set in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). This is especially useful when you need identical sections with different settings.

General options

The first option defines **Duplex** printing, which also enables or disables the settings for the **Back** side of each sheet.
If Duplex is enabled, you can also:

- Check **Omit empty back side for Last or Single sheet** to reset a page to Simplex if it has an empty back side (no content and no Master Page). Thus changing a Duplex job into a Mixplex job may reduce volume printing costs as omitted back sides aren't included in the number of printed pages. On the other hand, depending on the printer type it may reduce the print speed as well.
- Check **Tumble** to duplex pages like in a calendar. (On Portrait output, this would be equivalent to short-edge duplex.)
- Check **Facing pages** to have the margins of the section switch alternately, so that pages are printed as if in a magazine or book.

The **Media rotation** setting rotates the Media 0, 90, 180, or -90 degrees.

Sheet position options

With the option **Same for all positions** checked, the same Master Page and Media will be applied to every page in the Print section.
When this option isn't checked, there are multiple groups, each defining the settings for pages grouped by their position within the section as it outputs: **First**, **Middle**, **Last** and **Single** sheets.

Each group defines:

- **Allow content on**: Selects on which face of the sheet content is allowed.
  If **Front only** or **Back only** is selected, the page acts as a Simplex page even though Duplex printing is enabled. The other page may contain a Master Page, but no contents will be printed on it. As such it does not count in the Content Page Number and Content Page Count markers which can be inserted via the Insert menu (see "Page numbers " on page 644).
- **Media**: Defines the media that is used. If the Media has Virtual Stationery defined, the selected image is shown as a background to each page that corresponds to the media's sheet position.
  - **Edit Script**: Click this button to create a script that defines which Media is used. See "Dynamically switching the Media" on page 657.
- **Master Page Front**: Defines the Master Page used for the front of the selected sheet's position. (Disabled if Back only is selected under Allow content on).
- **Master Page Back**: Defines the Master Page used for the back of the selected sheet's position. (Disabled if Front only is selected under Allow content on, or if Duplex is unchecked.)
  Omit Master Page Back in case of an empty back page omits the specified Master Page on the last backside of a section if that page is empty. That page will then also be skipped from the page count *unless* the page numbers continue on the next section (see "Configuring page numbers" on page 645).
  Note that if the **Omit empty back side for Last or Single sheet** option (see "General options" on the previous page) is checked as well, the empty backside will not appear in the output at all and will not be counted in any case.

The **Repeat sheet configuration** option can be used to have the sheet configuration repeat every *n* number of pages. This is useful when documents can be so long that they cannot fit into one envelope.

Style sheets dialog

The Stylesheet editor dialog is used to edit CSS style sheet resources. For information on the use of style sheets, see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881.

This dialog lets you edit the Global style sheet (context_all_styles.css, which by default applies to all contexts), and the style sheet that applies to the context that is currently being edited in the workspace: Print (context_print_style.css), Email (context_email_style.css) or Web (context_web_style.css). It also allows to edit any stylesheets imported with a Word file.

To open this dialog, select **Edit > Stylesheets...**.

- **Scope**: Use the drop-down to select **Global** (all contexts) or the context that is open in the workspace, such as **Print**. Selecting a context shows all it's CSS rules in the **Rule List**.
- **Show**: Use the drop-down to select whether to show all CSS rules or limit to certain types: **Class**, **ID** or **Element** rules.
- **Rule List**: Displays the list of rules in the currently selected style sheet.
- **Rule Display**: Displays the contents of the currently selected rule in the **Rule List**.
- **New**: Click to create a new rule with the Edit Rule dialog. See "New/Edit Rule dialog" below.
- **Edit**: Click to edit the currently selected rule in the Rule List using the Edit Rule dialog. See "New/Edit Rule dialog" below.
- **Delete**: Click to delete the currently selected rule in the Rule List.
- **Duplicate**: Click to create a duplicate of the currently selected rule in the Rule List using the Edit Rule dialog. The default name for the new rule is the name of the current one plus "-duplicated". See "New/Edit Rule dialog" below.
- **Move Up**: Move the currently selected rule in the Rule List up one position in the list.
- **Move Down**: Move the currently selected rule in the Rule List down one position in the list.
- **Save**: Click to save all changes to the style sheet and close the dialog.
- **Cancel**: Click to close the dialog without saving any changes.

New/Edit Rule dialog

The New/Edit Rule dialog shows the properties for a specific CSS selector.
Click the **Apply** button to see how a setting affects the elements that are subject to that selector. (You may have to move and resize the Stylesheet dialog before opening the Edit Rule dialog, in order to be able to see the template that you are working on.)
At any point you can click on the **Advanced** button to see the Advanced Stylesheet Rule. See "Advanced Stylesheet Rule" on page 1214.

- **Name**: The CSS Selector to which this rule applies. This can be a selector that is specific to Connect (see "Selectors in OL Connect" on page 1083) or any selector used in regular CSS, since they can also be used here (see CSS Selectors on MDN Web Docs for a simple reference page).

Type Tab

- **General group**:
  - **Font**: Select the font used to display text. See also: "Fonts" on page 917.
    This is equivalent to the CSS `font-family` property.
  - **Size**: Enter the size in a measure, named size or percentage.
    This is equivalent to the CSS `font-size` property.
  - **Color**: This is the color of the text. Select a named font color as defined in the Edit Colors dialog (see "Colors" on page 913) or click the colored square to create a new color or to enter a color manually. The color value must be a valid HTML color name or hexadecimal color code.
    This is equivalent to the CSS `color` property.
  - **Background Color**: This is the background color of the text. Select a named font color as defined in the Edit Colors dialog (see "Colors" on page 913) or click the colored square to create a new color or to manually enter a color value (a valid HTML color name or hexadecimal color code).
    This is equivalent to the CSS `background-color` property.
- **Spacing group**:
  - **Letter Spacing**: Set the space between characters in a text in measure or percentage.
    This is equivalent to the CSS `letter-spacing` property.
  - **Word Spacing**: Set the space between each word in a text in measure or percentage.
    This is equivalent to the CSS `word-spacing` property.
  - **Whitespace**: Specify how to handle white spaces inside of an element. See CSS White-Space for details.
    This is equivalent to the CSS `white-space` property.
- **Style group**: Check any option to apply the selected style to text within the element. This list shows the CSS property and value for each of the options.
  - **Bold**: Sets the `font-weight` to `700`.
  - **Italic**: Sets `font-style` to `italic`.
  - **Underline**: Sets `text-decoration` to `underline`.
  - **Strikethrough**: Sets `text-decoration` to `line-through`.
  - **Subscript**: Sets `vertical-align` to `super`.

- **Superscript**: Sets `vertical-align` to `sub`.
- **Capitalize**: Sets `text-transform` to `capitalize`.
- **Uppercase**: Sets `text-transform` to `uppercase`.
- **Lowercase**: Sets `text-transform` to `lowercase`.
- **Small-caps**: Sets `font-variant` to `small-caps`.

Formats Tab

- **General group**:
  - **Line-height**: Specify the height of each line in the element's text, in measure or percentage. Note that this is not spacing between lines, but rather the complete height of the line itself including the text.
    Equivalent to the `line-height` property.
  - **Align**: Select how text should be aligned, such as `left`, `center`, `right` or `justify`.
    Equivalent to the `align` property.
  - **First Indent**: Specify the indentation of the first line of each paragraph in the element.
    Equivalent to the `text-indent` property.
  - **Display**: Select how to display the element. This can also be used to hide an element completely using the `none` option. See CSS Display.
    Equivalent to the `display` property.
- **Breaks group**:
  - **Before**: Specifies whether a page break should occur before the element.
    Equivalent to the `page-break-before` property.
  - **Inside**: Specifies whether to accept page breaks within the element.
    Equivalent to the `page-break-inside` property.
  - **After**: Specifies whether a page break should occur after the element.
    Equivalent to the `page-break-after` property.
  - **Widows**: Specifies how to handle widows within a paragraph (lines appearing alone on the next page if the paragraph does not fit on the current one).
    Equivalent to the `widows` property.
    Widows and orphans are ignored if the `page-break-inside` property is set to `avoid`.

- **Orphans**: Specifies how to handle orphans within a paragraph (lines appearing alone at the end of a page if the paragraph does not fit on the current one). Equivalent to the `orphans` property.

Spacing Tab

See also: "Spacing" on page 922.

- **Padding group**: Defines padding (spacing inside the element) in measure or percentage:
  - **All sides**: Check to set all padding to use the Top value. Equivalent to the `border` property.
  - **Top, Left, Bottom, Right**: Set padding for each side. Equivalent to the `border-left`, `border-top`, `border-right` and `border-bottom` properties.
- **Margin group**: Defines margins (spacing outside the element) in measure or percentage:
  - **All sides**: Check to set all margins to use the Top value. Equivalent to the `margin` property.
  - **Top, Left, Bottom, Right**: Set the margin for each side. Equivalent to the `margin-left`, `margin-top`, `margin-right` and `margin-bottom` properties.

Border Tab

See also: "Border" on page 910.

- **Same for all sides**: Defines the border properties for all sides using the Top properties. Equivalent to the`border`property.
- **Top, Left, Bottom, Right**: Each group defines the following properties:
  - **Width**: Specify the thickness of the border. Equivalent to the `border-width` property.
  - **Style**: Specify the style of the border such as `solid`, `dashed` or `dotted`. Equivalent to the `border-style` property.
  - **Color**: Specify the color of the border. The color value must be a valid HTML color name (see standard colors, on MDN Web Docs), or a hexadecimal color code (see Hex Calculator). Equivalent to the `border-color` property.

Advanced Stylesheet Rule

The Advanced editor is used to manually input rules. Note that to use this dialog, basic knowledge of CSS rules is a pre-requisite, as no check is currently done to verify that properties and values are correct.

- Property List: Lists all the currently available properties for the selector.
  - **Property**: The name of the property. This must correspond exactly to a known property (see CSS Reference). An autocompletion drop-down displays to show possible values when typing.
  - **Value**: The value for the given property. The values must be valid for that property, see the CSS Reference link above and check the property for valid values.
- **New**: Click to create a new line and type in the property.
- **Delete**: Click to delete the currently selected property in the Property List.
- **Move Up**: Move the currently selected property in the Property List up one position in the list.
- **Move Down**: Move the currently selected property in the Property List down one position in the list.

Sync Summary dialog

The Sync Summary dialog reports any discrepancies between the elements in the template that are tagged for translation (see "Tagging elements for translation" on page 1122) and the translation entries on the "Translations pane" on page 1260. To open it, click the **Sync** button on the **Translations** pane.

Remote snippets and content added by scripts are not taken into account.

For more information about template translation, see "Translating templates" on page 1120.

- **New Strings**: Lists the texts, found in elements that are tagged for translation, for which there is no matching translation entry.
- **Orphaned Strings**: Lists the translation entries that seem to be unused: currently there is no matching element in the content of the template for them. Note that seemingly unused translation entries may have been added for content in a remote snippet or for content that will be added by a script.

Per translation entry you have the following options.

- **Source**: This is the text that will be presented to the translator, and that will be replaced with a translation when OL Connect generates output from the template. It is case-sensitive.
- **Context** (optional): Describe where the source text is used, for example: "This text is in the head of a table", or "Table|Head", to help the translator make the best choice when the source text appears more than once and should be translated differently in different locations.
  The description is added to the <data-translate> attribute of the HTML element in which the source text is located, as its value, for example: <data-translate="Table">. Note that a translation will only be applied if the value of the <data-translate> attribute of the respective HTML element is *exactly* the same as the given Context (case-sensitive).

Click **OK** to update the Translations pane and add the new translation entries.

Tip: The **pluralization** option and the option to write **instructions for the translator** are only available in the Translation String Options dialog. After closing the Sync Summary dialog, double-click on a translation entry in the Translations pane to open it.

Table Formatting dialog

The Table Formatting dialog defines how a table looks. Note that the settings are applied to the table as a whole. For example, when you change the border of the table, the borders of cells inside the table will not be changed. For more information see "Styling a table" on page 902.

All settings in this dialog are in fact CSS properties. Cascading Style Sheets (CSS) were originally designed for use with web pages: HTML files. Since Designer templates are HTML files, they are styled with CSS. To learn how to use CSS in the Designer, see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881. For information about specific properties and their options, see MDN Web Docs - CSS Reference.

Note: Using viewport based units (vw, vh, vmin, vmax) in Print sections is not recommended. This may cause differences between the preview in the Designer and the printed output.

Table Tab

- **General group**:
  - **Width**: Set the width of the table in measure or percentage.
    Equivalent to the CSS `width` property.
  - **Height**: Set the height of the table in measure or percentage.
    Equivalent to the CSS `height` property.
  - **Angle**: Set the rotation angle of the table in clockwise degrees.
    Equivalent to the CSS `transform:rotate` property.
  - **Corner radius**: Set the radius of rounded border corners in measure or percentage.
    Equivalent to the CSS `border-radius` property.
  - **Display**: Use the drop-down or type in the value for how to display the table.
    Equivalent to the CSS `display` property.
  - **Overflow**: Use the drop-down or type in the value for how to handle overflow (text that does not fit in the current size of the box).
    Equivalent to the CSS `overflow` property.
- **Text wrap group**:
  - **Float**: Use the drop-down or type in the value for how to float the table, if the table is not in an absolute position.
    Equivalent to the CSS `float` property.
  - **Clear**: Use the drop-down or type in the value for clearing pre-existing alignments.
    Equivalent to the CSS `clear` property.
- **Positioning**:
  - **Position**: Use the drop-down or type in the value for the type of positioning for the table.
    Equivalent to the CSS `position` property.
  - **Top**: Set the vertical offset between this table and its parent's top position.
    Equivalent to the CSS `top` property.
  - **Left**: Set the horizontal offset between this table and its parent's left position.
    Equivalent to the CSS `left` property.

- **Bottom**: Set the vertical offset between this table and its parent's bottom position.
  Equivalent to the CSS `bottom` property.
- **Right**: Set the horizontal offset between this table and its parent's left position.
  Equivalent to the CSS `right` property.
- **Z-index**: Set the z-index of the table. The z-index defines in which order elements appear.
  Equivalent to the CSS `z-index` property.

- **Breaks group**:
  - **Before**: Specifies how to handle page breaks before the table.
    Equivalent to the CSS `page-break-before` property.
  - **Inside**: Specifies whether to accept page breaks within the table.
    Equivalent to the CSS `page-break-inside` property.
  - **After**: Specifies how to handle page breaks after the table.
    Equivalent to the CSS `page-break-after` property.
  - **Widows**: Specifies how to handle widows within the table (rows appearing alone on the next page if the table does not fit on the current one).
    Equivalent to the CSS `widows` property. Widows and orphans are ignored if the `page-break-inside` property is set to `avoid`.
  - **Orphans**: Specifies how to handle orphans within the tables (rows appearing alone at the end of a page if the table does not fit on the current one).
    Equivalent to the CSS `orphans` property.

Spacing Tab

For information about spacing see "Spacing" on page 922.

- **Padding group**: Defines padding (spacing inside the element) in measure or percentage:
  - **All sides**: Check to set all padding to use the Top value.
    Equivalent to the CSS `padding` property.
  - **Top, Left, Bottom, Right**: Set padding for each side.

Equivalent to the CSS `padding-left`, `padding-top`, `padding-right` and `padding-bottom` properties.

- **Margin group**: Defines margins (spacing outside the element) in measure or percentage:
  - **All sides**: Check to set all margins to use the Top value.
    Equivalent to the CSS `margin` property.
  - **Top, Left, Bottom, Right**: Set the margin for each side.
    Equivalent to the CSS `margin-left`, `margin-top`, `margin-right` and `margin-bottom` properties.

Border Tab

For information about borders see "Border" on page 910.

- **Same for all sides**: Defines the border properties for all sides using the Top properties.
  Equivalent to the CSS `border` property.
- **Top, Left, Bottom, Right**: Each group defines the following properties:
  - **Width**: Specify the thickness of the border.
    Equivalent to the CSS `border-width`property.
  - **Style**: Specify the style of the border such as `solid`, `dashed` or `dotted`.
    Equivalent to the CSS `border-style` property.
  - **Color**: Specify the color of the border: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    Equivalent to the CSS `border-color` property.

Background Tab

For information about backgrounds see "Background color and/or image" on page 908.

- **General group**:
  - **Color**: Specify the color of the table background:select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexa-decimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    Equivalent to the CSS `background-color` property.
- **Background image group**:
  - **Source**: Click the **Select Image** button to select an image via the "Select Image dialog" on page 1206.
    Equivalent to the CSS `background` property.
  - **Size**: Select `auto`, `cover` or `contain` (for an explanation see MDN Web Docs - background-size), or type the width and height of the image in a measure (e.g. `80px 60px`) or as a percentage of the parent element's size (e.g. `50% 50%`).
    Equivalent to the CSS `background-size` property.
  - **Position**: Select the position for the background-image.
    Equivalent to the CSS `background-position` property.

Table Cell Formatting dialog

The Table Cell Formatting dialog defines how a particular cell in a table looks. For more information see "Styling a table" on page 902.

All settings in this dialog are in fact CSS properties. Cascading Style Sheets (CSS) were originally designed for use with web pages: HTML files. Since Designer templates are HTML files, they are styled with CSS. To learn how to use CSS in the Designer, see "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881. For information about specific properties and their options, see MDN Web Docs - CSS Reference.

Note: Using viewport based units (vw, vh, vmin, vmax) in Print sections is not recommended. This may cause differences between the preview in the Designer and the printed output.

Cell Tab

- **Width**: Set the width of the table in measure or percentage. Equivalent to the CSS `width` property.
- **Height**: Set the height of the table in measure or percentage. Equivalent to the CSS `height` property.
- **Vertical Align**: Specify how text is vertically aligned in the cell: top, middle, bottom or baseline. With the baseline value all the table data share the same baseline. Often this has the same effect as the bottom value. However, if the fonts are in different sizes, baseline looks better.

Type Tab

- **General group**:
  - **Font**: Select the font used to display text.
    Equivalent to the CSS `font-family` property.
  - **Size**: Enter the size in measure, named size or percentage.
    Equivalent to the CSS `font-size` property.
  - **Color**: You may select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    Equivalent to the CSS `color` property.
- **Spacing group**:
  - **Letter Spacing**: Set the space between characters in a text in measure or percentage.
    Equivalent to the CSS `letter-spacing` property.
  - **Word Spacing**: Set the space between each word in a text in measure or percentage.
    Equivalent to the CSS `word-spacing` property.
  - **Whitespace**: Specify how to handle white spaces inside of an element.
    Equivalent to the CSS `white-space` property. See CSS white-space for details.
- **Style group**: Check any option to apply the selected style to text within the element:

- **Bold**: Sets the `font-weight` to `700`.
- **Italic**: Sets the `font-style` to `italic`.
- **Underline**: Sets the `text-decoration` to `underline`.
- **Strikethrough**: Sets the `text-decoration` to `line-through`.
- **Subscript**: Sets the `vertical-align` to `super`.
- **Superscript**: Sets the `vertical-align` to `sub`.
- **Capitalize**: Sets the `text-transform` to `capitalize`.
- **Uppercase**: Sets the `text-transform` to `uppercase`.
- **Lowercase**: Sets the `text-transform` to `lowercase`.
- **Small-caps**: Sets the `font-variant` to `small-caps`.

Spacing Tab

For information about spacing see "Spacing" on page 922.

- **Padding group**: Defines padding (spacing inside the element) in measure or percentage:
  - **All sides**: Check to set all padding to use the Top value.
    Equivalent to the CSS `border` property.
  - **Top, Left, Bottom, Right**: Set padding for each side.
    Equivalent to the CSS `padding-left`, `padding-top`, `padding-right` and `padding-bottom` properties.

Border Tab

For information about borders see "Border" on page 910.

- **Same for all sides**: Defines the border properties for all sides using the Top properties.
  Equivalent to the `border` property.
- **Top, Left, Bottom, Right**: Each group defines the following properties:
  - **Width**: Specify the thickness of the border.
    Equivalent to the CSS `border-width` property.
  - **Style**: Specify the style of the border such as `solid`, `dashed` or `dotted`.
    Equivalent to the CSS `border-style` property.
  - **Color**: Specify the color of the border: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square

to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
Equivalent to the CSS `border-color` property.

Background Tab

For information about backgrounds see "Background color and/or image" on page 908.

- **General group**:
  - **Color**: Specify the color of the table cell background: select a named color (defined in the "Colors Properties" on page 1144) from the drop-down, or click the colored square to open the Color Picker dialog ("Color Picker" on page 1143). Alternatively you could type a name or value in the Color field directly. It must be a valid CSS color name (CSS color names), a hexadecimal color code (HTML Hex Color), an RGB color value, for example `rgb(216,255,170)` or a CMYK color value, for example `cmyk(15%, 0%, 33%, 0%)`.
    Equivalent to the CSS `background-color` property.
- **Background image group**:
  - **Source**: click the **Select Image** button to select an image via the "Select Image dialog" on page 1206.
    Equivalent to the CSS `background`property.
  - **Size**: select `auto`, `cover` or `contain` (for an explanation see MDN Web Docs - background-size), or type the width and height of the image in a measure (e.g. `80px 60px`) or as a percentage of the parent element's size (e.g. `50% 50%`).
    Equivalent to the CSS `background-size` property.
  - **Position**: select the position for the background-image.
    Equivalent to the CSS `background-position` property.

Translation String Options dialog

The Translation String Options dialog sets the options for a translation entry. To open it, double-click a translation entry or click the Add Empty String button on the **Translations** pane.

For information about template translation, see "Translating templates" on page 1120.

- **String**:
  - **Source**: This is the text that will be presented to the translator, and that will be replaced with a translation when OL Connect generates output from the template. It is case-sensitive.
    When a new translation entry is created by tagging an HTML element for translation (see "Tagging elements for translation" on page 1122), the text of that element gets copied to this field.
  - **Context** (optional): Describe where the source text is used, for example: "This text is in the head of a table", or "Table|Head", to help the translator make the best choice when the source text appears more than once and should be translated differently in different locations.
    The description is added to the <data-translate> attribute of the HTML element in which the source text is located, as its value, for example: <data-translate="Table">.
    Note that a translation will only be applied if the value of the <data-translate> attribute of the respective HTML element is *exactly* the same as the given Context (case-sensitive).
- **Pluralization**: Check this option to make it possible for the translator to enter (text with) plurals. See "Pluralization" on page 1125.

  Note: The Singular and Plural given in this dialog are only used when there is no .po file that matches the locale.

  - **Singular**: Enter the text to use when the value of the selected data field is 1.
  - **Plural**: Enter the text to use when the value of the selected data field is **not** 1.
  - **Data field**: Select a data field that contains a numerical value. It depends on the value in this field whether the singular or plural is used.
- **Instructions for the translator**:
  - **Comment**: Enter a comment that helps the translator translate the Source text correctly, for example when the Source text contains "Data placeholders in translation entries" on page 1124 or "HTML tags in translation entries" on page 1124.

Keyboard shortcuts

This topic gives an overview of keyboard shortcuts that can be used in the Designer.

Although some of the keyboard shortcuts are the same, this isn't a complete list of Windows keyboard shortcuts. To find that, refer to the documentation of the Windows version that you are using.

Menu items

The following key combinations activate a function in the menu.

Key combination	Function
Alt	Put the focus on the menu. (Alt + the underlined letter in a menu name displays the corresponding menu.) The menu can then be browsed using the Enter key, arrow up and arrow down buttons.
Alt + F4	Exit
Alt + Shift + D	"Design tab" on page 1261
Alt + Shift + E	Preview HTML
Alt + Shift +L	Live view
Alt + Shift + P	"Preview tab" on page 1263
Alt + Shift + S	"Source tab" on page 1263
Alt + Shift + T	Tag element (see "Tagging elements for translation" on page 1122)
Ctrl + C or: Ctrl + Insert	Copy
Ctrl + F	Find
Ctrl + I	Properties
Ctrl + N	New
Ctrl + O	Open file
Ctrl + P	Print
Ctrl + R	Show/hide rulers
Ctrl + S	Save file
Ctrl + V or: Shift + Insert	Paste

Key combination	Function
Ctrl + X or: Shift + Delete	Cut
Ctrl + W	Close file
Ctrl + Y or: Ctrl + Shift + Z	Redo
Ctrl + Z	Undo
Ctrl + Alt + ;	Lock guides
Ctrl + Shif + R	Clear preview cache
Ctrl + Shift + S	Save all
Ctrl + Shift + W	Close all
Ctrl + Shift + ;	Snap to guides
Ctrl + ;	Show guides
Ctrl + '	Show/hide virtual stationery
Ctrl + \	Highlight Master Page items
Ctrl + F5	Revert
Ctrl + F10	Save as
Ctrl + F11	"Send COTG Test" on page 1203
Ctrl + F12	"Send to Workflow dialog" on page 1206
Ctrl + Shift + F12	Package

Workspace

The following key combinations activate a function in the Workspace.

Key combination	Function
Alt + **-**	Open system menu

Key combination	Function
Alt + F7 or: Alt + Page Down	Next tab
Alt + Shift + F7 or: Alt + Page Up	Previous tab
Ctrl + A	Select all
Ctrl + Shift + E	Switch to Editor
Ctrl + F	Find (opens the Find and Replace dialog: "Find/Replace Dialog" on page 1151)
Ctrl + F6	Next editor (when there is more than one file open in the Workspace)
Ctrl + Shift + F6	Previous editor (when there is more than one file open in the Workspace)
Shift + F10 or: Ctrl + Shift + F10	Open context menu

Design and Preview tab

In addition to the keyboard shortcuts for menu items and the Workspace, the following key combinations have a special function in the **Design** tab and **Preview** tab of the Workspace.

Key combination	Function
Ctrl + B	Bold (works on a text selection)
Ctrl + E	Open Paragraph formatting dialog
Ctrl + I	Italic (works on a text selection)
Ctrl + H	Show Edges
Ctrl + K	Delete browser element
Ctrl + M	Open Box formatting dialog
Ctrl + T	Open Text formatting dialog
Ctrl + U	Underline ((works on a text selection)

Key combination	Function
Ctrl + **+** or: Ctrl + Shift + **+** or: Ctrl + = or: Ctrl + Shift + =	Zoom in **Tip:** You can also use the mouse's scroll wheel in combination with the Ctrl button to gradually zoom in or out.
Ctrl + **-** or: Ctrl + Shift + **-**	Zoom out
Ctrl + 0	Zoom to page width
Ctrl + 1	Zoom to page content width
Ctrl + arrow up	Select parent
Ctrl + Shift + R or: F5	Refresh

Text editors: Source tab, JavaScript, CSS, Script Editor

The following key combinations have a special function in the **Source** tab of the Workspace (see also: "Source tab" on page 1263), and when editing a JavaScript or CSS file in the Workspace, and in the Script Editor (expanded view).

Key combination	Function
Alt + Down arrow	Moves the current line down
Alt + Up arrow	Moves the current line up
Alt + Ctrl + Down arrow	Makes a copy of the current line and inserts it below the current line
Alt + Ctrl + Up arrow	Makes a copy of the current line and inserts it above the current line
Ctrl + space	Content assist (auto-complete)
Ctrl + A	Select all
Ctrl + D	Duplicate line
Ctrl + F	Find
Ctrl + I	Indent (Tab)

Key combination	Function
Ctrl + J	• Script Editor: Add a line break. • Other editors: Incremental find; start typing a search string directly after pressing this key combination.
Ctrl + K	Find next
Ctrl + L	Go to line; a prompt opens to enter a line number.
Ctrl + Shift + D	Delete line
Tab	Expand Emmet abbreviation (see "Emmet preferences" on page 1049)
Shift + Tab	Shift selected lines left
Tab	Shift selected lines right
Ctrl + /	Comment out / uncomment a line in code
Ctrl + Shift + /	Comment out / uncomment a code block

Scripts pane and Resources pane

The following keys or key combinations have a special function when a file is selected in the **Resources** pane and when a script is selected in the **Scripts** pane.

Key combination	Function
F2	Rename
Alt + Enter	Open Properties dialog
Delete	Delete

Data Model pane

You can use the following keys to browse records in the Data Model pane:

- Page Up: next record.
- Page Down: previous record.
- Home: first record.
- End: last record.

Menus

The following menu items are shown in the Designer menu. For a list of keyboard shortcuts, see "Keyboard shortcuts" on page 1223.

File Menu

- **New...**: Opens the **New (Select a Wizard)** dialog that lets you create a data mapping configuration (see "Data mapping configurations" on page 337), a template (see "Templates" on page 589), a Print Preset (see "Print Presets" on page 1644) or run a Sample Project (see "Sample Projects" on page 1185).
- **Open**: Opens a standard File Open dialog. This dialog can be used to open templates (see "Templates" on page 589) and Print Presets (see "Print Presets" on page 1644). It can also be used to open data mapping configurations; see "Data mapping configurations" on page 337.
- **Open Recent**: Lists the most recently opened templates and data mapping configurations.
  The Designer can have one data mapping configuration and one template open at the same time. Use the tabs at the top of the workspace to switch between the two. Click the synchronize button on the Data Model pane to make sure that the Data Models are the same in both.
- **Close**: Closes the currently active template or data mapping configuration. If the file needs to be saved, the appropriate Save dialog will open.
- **Close All**: Closes any open template or data mapping configuration. If any of the files needs to be saved, the Save Resources dialog opens.
- **Close Others**: Closes all files except the one that is currently active in the workspace.
- **Save**: Saves the current file to its current location on disk. If the file has never been saved, the Save As dialog appears instead.
- **Save All**: Saves the open files. If a file has never been saved, the Save As dialog opens for it.
- **Save As...**: Saves the current file to a new location on disk.
- **Save a Copy**: Save a copy of the current template in the selected Connect version's format. See "Saving a copy / down-saving a template" on page 592.
- **Revert**: For templates only. Reverts all changes to the state in which the file was opened or created.

- **Data**: Adds data either to the current data mapping configuration or to the open template. See "Loading data" on page 926.
    - **Add Data Sample...**: Add a data sample to the currently open data mapping configuration.
    - **Open Data File...**: Opens the dialog to add a new data file. Not available if a data mapping configuration is currently open.
    - **Connect with Database...**: Opens the Edit Database Configuration dialog. Not available if the currently loaded data mapping configuration is file-based.
    - **Generate Counters**: Opens the Generate Counter Wizard to create a custom counter as a data source.
    - **Open JSON Sample Data...**: Opens the dialog to load JSON sample data in the Data Model pane; see "JSON sample data dialog" on page 1158. This option is only available when no data mapping configuration is open.
- **Send to Workflow...**: Opens a dialog to send files to a local Workflow configuration tool. See "Send to Workflow dialog" on page 1206.
- **Send to Server**: Opens the Send to Server dialog to send files to a Connect Server or to another server.
- **Package...**: Opens the dialog that lets you package files; see "Package dialog" on page 1171.
- **Export report**: Opens the wizard to save a template report. See "Exporting a template report" on page 594.
- **Properties**: Opens the "File Properties dialog" on page 1150.
- **Print**: Opens the "Print options" on page 1380 dialog.
- **Proof Print**: Opens the "Print options" on page 1380 dialog as a Proof Print dialog which limits the number of records output. The options themselves are identical to the regular Print Output dialog.
- **Send Email**: Opens the Send Email dialog; see "Send (Test) Email" on page 1204 and "Generating Email output" on page 1673.
- **Send Test Email**:Opens the Send Test Email dialog; see "Send (Test) Email" on page 1204.
- **Send COTG Test**: Opens the "Send COTG Test" on page 1203 dialog, to send the current Web section to the Capture OnTheGo Application.

- **Exit**: Closes the software. If any of the files needs to be saved, the Save Resources dialog opens.

Edit Menu

- **Undo <action>**: Undoes the previous action that was done.
- **Redo <action>**: Redoes an action that was previously undone.
- **Cut**: Cuts the currently selected text, object or element and puts it on the clipboard.
- **Copy**: Copies the currently selected text, object or element to the clipboard.
- **Copy to snippet**: Creates a new snippet from the selected text, object or element.
- **Paste**: Takes the current clipboard content and pastes it at the pointer location.
- **Paste as Plain Text**: Takes the current clipboard content and pastes it at the pointer location without any HTML styles or formatting.
- **Delete Browser Element**: Removes the currently selected element in the workspace.
- **Find/Replace**: Only active while inside the Workspace. Opens the Find/Replace dialog.
- **Stylesheets...**: Open the "Style sheets dialog" on page 1209. See "Styling and formatting" on page 879 and "Styling templates with CSS files" on page 881.
- **Colors...**: Opens the Colors Editor dialog. See "Colors" on page 913.
- **Fonts...**: opens the "Font Manager" on page 1152. See "Fonts" on page 917.
- **Locale...**: Opens the Locale Settings dialog. See "Locale" on page 921.
- **Color Settings...**: Opens the Color Settings dialog. See "Colors" on page 913.

Insert Menu

- **Image**: Inserts an image using a resource that is local to the template, a resource on disk or a URL. See "Images" on page 852.
- **Text**:
    - **Wrap in span**: Wraps selected text in a element. The ID or class of the span can be used as a selector for scripts and styles.
- **Special Characters**: Displays a categorized list of special HTML characters that can be inserted at the current pointer location. When a character is clicked, its HTML Entity is inserted. This includes:
    - **Symbols**: Use the list to insert a special symbol such as Copyright, Trademark, or Ellipsis.

- **Markers**: Use the list to insert pagination markers that are replaced with specific page numbering:
    - **Page Number**: This marker is replaced by the current page number in the document. Even if the page number is not used on certain pages, those page are still added to the page count.
    - **Page Count**: This marker is replaced by the total number of pages in the document, including pages with no contents.
    - **Content Page Number**: This marker is replaced by the current page number (with contents) in the document.
    - **Content Page Count**: This marker is replaced by the total number of pages that have contents in them, in the document. A page with contents is a page that is part of a section that has variable data on it. A page with a Master Page but no contents (set in the Sheet Configuration tab of the "Print section properties" on page 1200) is not included in the Content Page Count.
    - **Sheet Number**: This marker is replaced by the current sheet number (physical piece of paper with two sides, or pages) in the document. This is equivalent to half the page number, for example if there are 10 pages, there will be 5 sheets.
    - **Sheet Count**: This marker is replaced by the total number of sheets in the document, whether or not they have contents.
- **Dashes and Spaces**: Use the list to insert special dashes, such as an em-dash, and spaces, such as non-breaking spaces or an en-space. (The HTML code inserted for the dash or space is visible on the Source tab of the workspace.)
- **Arrows**: Use the list to insert directional arrows (in one of four directions).
- **Geometric Shapes**: Use the list to insert a special geometric shape, such as circles, triangles and squares.

- **Date**: Opens the "Date" on page 841 dialog to add a date to the template based on the current system's date and time.
- **Wrap in box**: Puts the element in which the cursor is located in an inline box (a).
- **Table**
    - **Thead, tbody, tfoot**: Insert a header, body or footer (if not already present) in the current table.

    - **Standard**: Inserts a table with a specific number of columns and rows through the Standard Table Wizard; see "Table" on page 860.
    - **Dynamic** : Inserts a Dynamic Table in which the number of rows is determined by a detail table in the record, through the Dynamic Table Wizard; see "Dynamic Table" on page 974.
- **Table Elements**:
    - **Insert Row Above**: Inserts a row above the current one. The row configuration, such as merged cells and cell styles, is duplicated, but contents is not.
    - **Insert Row Below**: Inserts a row below the current one. The row configuration, such as merged cells and cell styles, is duplicated, but contents is not.
    - **Insert Column Before**: Inserts a column to the left of the current one. The column configuration, such as merged cells and cell styles, is duplicated, but contents is not.
    - **Insert Column After**: Inserts a column to the right of the current one. The column configuration, such as merged cells and cell styles, is duplicated, but contents is not.
- **Common Elements**:
    - **Paragraph...**: Opens a dialog to add a <p> element; see "Text and special characters" on page 865.
    - **H1 through H6...**: Opens a dialog to add a <h1> to <h6> element; see "Text and special characters" on page 865.
    - **Address...**: Opens a dialog to add an <address> element.
    - **Preformatted...**: Opens a dialog to add a <pre> element.
- **Structural Elements**:
    - **Div**...: Opens a dialog to add a element; see "Boxes" on page 823
    - **Span**...: Opens a dialog to add a element; see "Boxes" on page 823
    - **Article**...: Opens a dialog to add an <article> element
    - **Section**...: Opens a dialog to add a <section> element (the HTML element, not a section in a context).
    - **Header**...: Opens a dialog to add a <header> element.
    - **Footer**...: Opens a dialog to add a <footer> element.
    - **Nav**...: Opens a dialog to add a <nav> element.

- **Aside**...: Opens a dialog to add an <aside> element.

  > **Note:** Article, Section, Header, Footer, Nav and Aside are content sectioning elements; see MDN Web Docs - content sectioning elements

- **Form Elements** (see "Form Elements" on page 847)
  - **Form...**: Opens a dialog to add a Form Element; see "Forms" on page 842.
  - **Fieldset...**: Opens a dialog to add a Fieldset Element; see "Fieldset" on page 848.
  - **Text Field...**: Opens a dialog to add a Text Field; see "Text" on page 848.
  - **Email Field...**: Opens a dialog to add an Email Field; see "Email" on page 848.
  - **URL Field...**: Opens a dialog to add a URL Field; see "URL" on page 848.
  - **Password Field...**: Opens a dialog to add a Password Field; see "Password" on page 848.
  - **Text Area...**: Opens a dialog to add a Text Area; see "Text area" on page 848.
  - **Date Field...**: Opens a dialog to add a Date Field; see "Date" on page 848.
  - **Number Field...**: Opens a dialog to add a Number Field; see "Number" on page 848.
  - **Hidden Field...**: Opens a dialog to add a Hidden Field; see "Hidden field" on page 848.
  - **Label...**: Opens a dialog to add a Label; see "Label" on page 848.
  - **Checkbox Field...**: Opens a dialog to add a Checkbox; see "Checkbox" on page 849.
  - **Radio Button...**: Opens a dialog to add a Radio Button; see "Radio Button" on page 849.
  - **Select Field...**: Opens a dialog to add a Select (drop-down); see "Select" on page 849.
  - **Button...**: Opens a dialog to add a Button; see "Button" on page 850.
  - **Help text**: Opens a dialog to insert a paragraph (<p>) for help text.
- **COTG Form Elements** (see "COTG Elements" on page 835).
  - **Signature...**: Opens a dialog to add a Signature element, see "Signature" on page 840.

- **Date...**: Opens a dialog to add a Date Element, see "Date and Formatted Date" on page 838.
- **Date Formatted...**: Opens a dialog to add a Formatted Date Element, see "Date and Formatted Date" on page 838.
- **Time...**: Opens a dialog to add a Time element, see "Time and Formatted Time" on page 840.
- **Time Formatted...**: Opens a dialog to add a Formatted Time element, see "Time and Formatted Time" on page 840.
- **Geolocation...**: Opens a dialog to add a Geolocation element, see "Geolocation" on page 839.
- **Locale...**: Opens a dialog to add a Locale element, see "Locale" on page 840.
- **Camera...**: Opens a dialog to add a Camera element, see "Camera" on page 836.
- **Image and annotation**: Opens a dialog to add an image that can be annotated by the user; see "Image & Annotation" on page 839.
- **Barcode Scanner...**: Opens a dialog to add a Barcode Scanner element, see "Barcode Scanner" on page 835.
- **User Account...**: Opens a dialog to add a User Account element, see "User Account" on page 841.
- **Device Info...**: Opens a dialog to add a Device Info element, see "Device Info" on page 838.
- **Repository ID**: Opens a dialog to add a Repository ID element, see "Repository ID" on page 840.
- **Document ID**: Opens a dialog to add a Document ID element, see "Document ID" on page 839.
- **Fields Table**: Opens a dialog to add a Fields Table element, see "Fields Table" on page 839.

- **Form Wizard**: Opens the Form Wizard to add a form to a Web context; see "Forms" on page 842
- **Validation Wizard**: Opens the Validation Settings dialog to change the validation settings on the currently selecting tools; see "Changing a Form's validation method" on page 846.
- **Business Graphic**: Displays a list of available business graphic object to be inserted:

    - **Insert Pie Chart**: Opens the Pie Chart script dialog to insert a new Pie Chart.
    - **Insert Bar Chart**: Opens the Bar Chart script dialog to insert a new Bar Chart.
    - **Insert Line Chart**: Opens the Line Chart script dialog to insert a new Line Chart.
- **Barcode**: Displays a list of available barcodes. Click on one to insert it in the page. See "Barcode" on page 768.

Format Menu

- **Size**: When text is selected, choose a predefined or custom font size in this submenu to change the size of the selected text.
    - **Other...**: Opens the Text Formatting dialog for advanced style selection; see "Styling text and paragraphs" on page 892.
    - **7pt - 72pt**: Sets the size of the selected text to the chosen font size.
- **Style**: When text is selected, sets the text style by applying or removing the following attributes: Plain, Bold, Italic, Underline, Strikethrough, Subscript, Superscript, Capitalize, Uppercase, Lowercase, Small-caps. This is the same as opening the Text Formatting dialog (**Format > Text**) and checking the appropriate style. See "Styling text and paragraphs" on page 892.
- **Color**: When text is selected, sets the text color by applying the color attribute to the text. The color submenu lists all the colors in the Colors Editor.
- **Text...**: Opens the Text Formatting dialog to modify the current text selection. See "Styling text and paragraphs" on page 892.
- **Align**: When an element is selected, determines how its contents is aligned inside the element. Options are Align Left, Align Right, Align Center and Justify.
- **Paragraph...**: Opens the "Paragraph Formatting dialog" on page 1172 to modify the paragraph where the cursor is located. See "Styling text and paragraphs" on page 892.
- **Paragraph Format**: Displays a list of generic element types that can be used for a text element. Selecting one of them converts the element where the cursor is located into the appropriate element (for example <p> for Paragraph, <h3> for Heading 3, etc).
- **Float**
    - **Left**: Floats the current element to the left. This is equivalent to setting the CSS `float` property to `left`.
    - **Right**: Floats the current element to the right. This is equivalent to setting the CSS `float` property to `right`.
    - **None**: Removes any float style applied to the currently selected element.

- **Align Objects**:
  - **Top**: Aligns the top side of the selected objects with the top edge of the last selected object.
  - **Middle**: Aligns the middle of the selected objects with the middle of the last selected object. Objects may move up or down.
  - **Bottom**: Aligns the bottom side of the selected objects with the bottom edge of the last selected object.
  - **Left**: Aligns the left side of the selected objects with the left edge of the last selected object.
  - **Center**: Aligns the center of the selected objects with the vertical center of the last selected object. Objects may move to the left or to the right.
  - **Right**: Aligns the right side of the selected objects with the right edge of the last selected object.
- **Box...**: Opens the "Box Formatting dialog" on page 1138 to modify the box where the cursor is located.
- **Image...**: Opens the "Image Formatting dialog" on page 1153 to modify the image that is currently selected.
- **Table...**: Opens the "Table Formatting dialog" on page 1215 to modify the table in which the cursor is located. If the cursor is within a table embedded within another, the innermost table's formatting is the one modified.
- **Table Cell...**: Opens "Table Cell Formatting dialog" on page 1219 to modify the cell where the cursor is located.
- **Hyperlink**
  - **Insert...**: Creates a hyperlink on the currently selected text or element and opens its properties; see "Hyperlink and mailto link" on page 850.
  - **Edit...**: Opens the properties for the currently selected hyperlink; see "Hyperlink and mailto link" on page 850.
  - **Remove**: Removes the currently selected hyperlink. The text or element that was the hyperlink is not removed.

Context Menu

- **Add**:
  - **Print Context**: Adds a new Print context to the template if one does not exist.
  - **HTML Email Context**: Adds a new Email context to the template if one does not exist.
  - **Web Page Context**: Adds a new Web context to the template if one does not exist.
- **Delete**: Deletes the currently selected context. The last remaining context cannot be deleted.
- **Go to**: Opens the first section in the selected context. This is the same as double-clicking on the first section of any context in the Resource Pane.
- **Color Output**: Opens the Print context's Color Output options dialog (see "Overprint and black overprint" on page 628). This option is only available when editing a Print section in the Workspace.
- **Finishing**: Opens the Print context's Finishing options dialog (see "Setting the binding style for the Print context" on page 627). This option is only available when editing a Print section in the Workspace.
- **PDF Attachments**: Opens a dialog to set the compression for PDF attachments; see "PDF Attachments dialog" on page 1176. This option is only available when editing an Email section in the Workspace.
- **Includes**: Opens the Includes dialog; see "Includes dialog" on page 1157. This option is only available when editing a Web section in the Workspace.
- **Preview HTML**: Opens the currently selected section in the default system browser to preview it. This feature works in all contexts.
- **Profile Scripts**: Opens the "Profile Scripts dialog" on page 1181 to test script performance (see "Testing scripts" on page 1074).
- **Preflight**: Opens the Preflight dialog. Preflight checks the template for common errors (see "Testing scripts" on page 1074).

Section Menu

- **Add**: Adds a new section to the currently selected context.
- **Delete**: Deletes the currently selected section.
- **Arrange**: Opens the "Arrange Sections" on page 1202 dialog.
- **Go to**: Lists the sections in the currently selected context. Open one by clicking it.

- **Properties...**: Opens the appropriate section properties. See "Section properties dialogs" on page 1199.
- **Includes...**: Opens the "Includes dialog" on page 1157.
- **Attachments**: Opens the "Attachments tab" on page 1200 in the Email section properties.
- **Finishing...**: Opens the Finishing tab in the "Print section properties" on page 1200.
- **Sheet Configuration...**: Opens the "Sheet Configuration dialog" on page 1208.
- **Master Pages**: Lists the available Master Pages in the template (see "Master Pages" on page 648). Open one by clicking it.
- **Master Page Properties**...: Opens the currently selected Master Page's properties dialog; see "Master Pages" on page 648.

View Menu

- **50/75/100/150/200%**: Zooms the Workspace at the selected level.
- **Source View**: Shows the HTML source for the template, including CSS and HTML code.
- **Design View**: Shows the template including all styles, text and images as well as the placeholders used for variable data.
- **Preview View**: Shows the template as it will output with the current record, with the personalized content (see "Personalizing content" on page 923).
- **Refresh**: Reloads the view, including static external images, remote stylesheets, and remote scripts, and re-runs the scripts (the latter in Preview Mode only).
- **Show Edges**: Shows or hides a colored border around elements on the page.
- **Rulers**: Shows or hides the rulers in the Workspace. Rulers only appear for Print contexts.
- **Guides**:
    - **Show Guides**: Shows or hides the margin lines and guides in a Print section (see "Print" on page 616, "Page settings: size, margins and bleed" on page 642 and "Guides" on page 899). The colors of margin lines and guides are adjustable; see "Editing preferences" on page 1044.
    - **Lock Guides**: Locks the guides, so that they cannot accidentally be moved while working on the Print context.
    - **Snap to Guides**: Enables or disables snapping to guides and to margins when moving objects.

- **Virtual Stationery**: Enables or disables the visibility of the PDF Background image set in the Media.
- **Highlight Master Page Items**: Enables or disables a yellow border around Master Page items in a section.
- **Object Resizing**: Enables or disables the ability to resize elements on the page. See "Editing preferences" on page 1044 for more fine-tuned control.
- **Shared Content Editing**: Enables or disables the ability to edit shared content in locations where it is used. See "Snippets" on page 867.

Window Menu

- **Show View**>: Use the options in this menu to show or hide different panes of the UI.
  - **Translations**: Shows the "Translations pane" on page 1260.
  - **Outline**: Shows the "Outline pane" on page 1250.
  - **Messages**: Shows the Messages pane, see "Preflight Results and Messages" on page 1248.
  - **Data Model**: Shows the "Data Model pane" on page 1244.
  - **Preflight Results**: Shows the results of a Preflight; see "Doing a Preflight" on page 1075.
  - **Scripts**: Shows the "Scripts pane" on page 1256.
  - **Parameters**: Shows the Parameters pane. See "Runtime parameters" on page 608.
  - **Styles**: Shows the "Styles pane" on page 1260.
  - **Attributes**: Shows the "Attributes pane" on the next page.
  - **Resources**: Shows the "Resources pane" on page 1250.
- **Reset Perspective**: Resets all toolbars and panes to the initial configuration of the module.
- **Clear Recent Files Lists**: Clears the lists of the most recently opened templates and data mapping configurations under File > Open Recent and in the Welcome screen.
- **Preferences**: Click to open the Preferences dialog.

Help Menu

- **Software Activation**: Displays the Software Activation dialog. See "Activating a License" on page 64.
- **Online Help**: Opens the Online Help in the default web browser.

- **Release Notes**: Opens the Release Notes of the installed version in the Online Help.
- **Contact Support**: Opens the Objectif Lune contact page in the default system web browser.
- **About OL Connect Designer**: Displays the software's About dialog. this allows you to copy the product name and version number.
- **Welcome Screen**: Re-opens the Welcome Screen.

Panes

Panes are windows containing user interface elements (such as information or properties), which can be docked and undocked, moved around and merged together through tabbed panes.

Here is a list of all panes:

Attributes pane

The Attributes pane displays all of the properties of the currently selected object in the Workspace. These properties vary greatly depending on the object that has been selected.

General

These attributes are common to all elements in the template and will always appear.

- **ID**: A unique identifier for the selected element. Used for CSS selections (see "Styling templates with CSS files" on page 881) as well as scripts (see "Writing your own scripts" on page 1064) affecting **single** elements.
- **Class**: One or more classes that can be common to more than one elements. Used for CSS selections and scripts that can affect **multiple** elements.

Note: Do not give an element the ID 'pages' or the class name 'dynamic'. These are reserved words. Using them as an ID or class name leads to undesirable effects.

Other

These attributes are available depending on the item selected.

- **Allow Resizing**: Allows columns in a **table** to be resized directly within the Workspace in Design mode.

- **Alternate text**: The "Alt" text used when hovering over the **image** in a browser. Also used for accessibility.
- **Cellspacing**: Defines the *cellspacing* attribute of the **table** which controls the spacing between cells in the table.
- **Cellpadding**: Defines the *cellpadding* attribute of the **table** which controls the padding inside each cell of the table.
- **colspan**: Number of columns that this Dynamic Table **cell** spans.
- **Expand table**: If this option is unchecked, the table will not be merged with data in the Preview mode or in output. This can be useful when debugging or optimizing a template.
- **Field**: Associates an **element** in a Dynamic Table body row with a data field.
  This setting corresponds to the `data-field` attribute (see "A Dynamic Table's data-attributes" on page 988).
- **Format**: Defines how a value must be formatted. Which formats are available depends on the data type of the corresponding field in the Data Model (see "Data types" on page 436). For a detailed description of the options see: "Formatting variable data" on page 952.
  This setting corresponds to the `data-format` attribute (see "A Dynamic Table's data-attributes" on page 988).
- **Hide when empty**: By default, a Dynamic Table gets hidden if it contains no data. Uncheck this option to keep the empty table in the output.
  This setting corresponds to the `data-hide-when-empty` attribute (see "A Dynamic Table's data- attributes" on page 988).
- **Repeat**: Associates a Dynamic Table body **row** with a detail table in the data. The row will be repeated once for each record in the detail table. See "Dynamic Table" on page 974.
- **Show Row**:
  - Repeats a row in the **body** of a Dynamic Table, if that row's sub-table runs over multiple pages. The options are:
    - **Initial** (default): The row will only appear once.
    - **All**: The row will appear on all pages.
    - **Before page break**: The row will appear on all pages except the last one.
    - **After page break**: The row will appear on all pages except the first one.

- Rows in the **header** or **footer** of a Dynamic Table have the following options:
  - **All**: The row will appear on all pages.
  - **Before page break**: The row will appear on all pages except the last.
  - **After page break**: The row will appear on all pages except the first.
  - **At end of table**: The row will only appear on the last page.

  This setting corresponds to the `data-show-row` attribute (see "A Dynamic Table's data- attributes" on page 988).

- **Source**: The location of the **image** file. For image resources in the template, the image path is often `images/<imagefile>.<extension>`
  When the source is a PDF, an addition button appears next to this box that opens the "Select Image dialog" on page 1206.
- **Sum**: Adds a running total (i.e. a subtotal) to a cell in a Dynamic Table. See "Adding subtotals" on page 982.
- **Target**: The target of a Hyperlink element. See "Hyperlink and mailto link" on page 850 and MDN Web Docs - Anchor element.
- **Type (form input element)**: Use the drop-down to select an input type. The drop-down lists all input types, including HTML5 input types (see MDN Web Docs - HTML Forms).
- **URL**: The URL of a Hyperlink element. See "Hyperlink and mailto link" on page 850.
- **Whitespace element**: Check to make the element a whitespace element, meaning it will only appear on the page if there is enough space for it. This is useful for templates with variable height elements or conditional elements, to fill empty spaces with transpromotional material. Note that only top-level elements (i.e. not inside another element such as a table or div) will function as whitespace elements.

Geometry

These attributes are available for certain elements that have position or size attributes such as images and boxes.

- **Top**: The horizontal distance from the top-left of the object to the left position of its parent. This is used only for relative and absolute positioned elements.
- **Left**: The vertical distance from the top-left of the object to the top position of its parent.
- **Width**: The width of the element, by default in pixels. For an image, this defaults to the original image width in pixels.

- **Height**: The height of the element, by default in pixels. For an image, this defaults to the original image height in pixels.
- The **Reset Image Size** button resets the selected image to its original size.

Note: When no unit is added to a geometry value, the default unit will be added to the value; see "Print preferences" on page 1057.

Page

These attributes appear when selecting the *Page* node in the Outline Pane.

- **Master Page**: Which of the "Master Pages" on page 648 to use for the template.

Data Model pane

The Data Model Pane displays a Data Model used to help design the template, along with (optional) data. When you open or create a data mapping configuration (see "Data mapping configurations" on page 337) or directly load data (see "Loading data" on page 926), the resulting record set is loaded in the Data Model pane. The information shown is the information for the current record within the record set.

Data is displayed as a tree view, with the root level being the record table, levels below it being detail tables, and any level below being called "nested tables".

Nested objects in JSON data are displayed on successive levels in the Data Model.

Filter

To find a certain table, group or field in a large Data Model, start typing characters in the Filter box. This immediately narrows down the list of displayed items to all tables, groups and fields whose **name** or **value** contains the characters that were typed. Note that the filtering is case-insensitive.

Data Model toolbar buttons

- **JSON Sample Data**: Opens a dialog to load JSON sample data in the Data Model pane; see "Adding JSON sample data" on page 937.

  Tip: If a data field contains JSON data you could use that data as sample data: right-click the field, select **Copy** to copy the JSON data to the clipboard; then open the JSON Sample Data dialog and paste the data there.

- **Synchronize Fields and Structure**: Click to synchronize the Data Model fields and structure in the currently loaded template and data mapping configuration. If you click this button when working on the data mapping configuration, the Data Model gets updated to the one in the template. If you click it when working on the template, the Data Model gets updated to the one in the data mapping configuration.
- **Sample data file**: Hover over the button to see the name of the currently active sample data file. Click the down arrow and select a sample data file to quickly switch between available sample data files.
- **Collapse All**: Collapse all fields on the record level, in all detail tables and in all groups.
    - **Import Data Model**: Click to browse to a file that contains a Data Model. This may be:
        - A Data Model file (*.OL-datamodel).
        - A JSON file (*.json; see the note below).
        - A Connect template (*.OL-template), data mapping configuration (*.OL-datamapper) or 'typed' JSON file (*.json; see the note below).

      The file's data model structure will be displayed in the Data Model pane without data.
    - **Export Data Model**: Click to browse to a location to save the Data Model file. The available file types are:
        - A Data Model file (*.OL-datamodel).
        - A JSON file (*.json): A JSON object or an array of JSON objects representing records.
        - A 'typed' JSON file (*.json).

      Note: Typed JSON follows the structure of a JSON Record Data List (see the REST API Cookbook). Data field types are determined by the `schema` object in the JSON.
      When a Data Model is exported to a JSON file or typed JSON file, detail tables are exported, but groups are not.
    - **Show/Hide ExtraData fields**: This icon is only visible if the View ExtraData option has been checked in the Preferences: **Window** > **Preferences** > **Data Model**.

      ExtraData fields are not meant to be filled via an extraction. They were originally

intended to be used in a Workflow configuration to add data to the Data Model (see "Adding fields and data via Workflow" on page 424).

Nowadays, **runtime parameters** are the preferred way to pass data at runtime to a data mapping configuration (see "Properties and runtime parameters" on page 387) or template (see "Runtime parameters" on page 608).

- **Minimize/Maximize**: Click to minimize or maximize the pane. See Moving and Merging Panes.

Browsing data

When a Data Model is loaded in the Data Model Pane, it can be used to design templates by dragging fields directly into the template; see "Variable data in the text" on page 924.

If data is present (from a data file or a data mapping configuration), it is possible to preview the resulting data in the template on the **Preview** tab (see Workspace).

You can use the following keys to browse records in the Data Model pane:

- Page Up: next record.
- Page Down: previous record.
- Home: first record.
- End: last record.

The way the Data Model is displayed can be changed via the following options in the contextual menu. Right-click the Data Model to open the contextual menu.

- **Collapse All**: Collapse all fields on the record level, in all detail tables and in all groups.
- **Expand All**: Expand all fields on the record level, in all detail tables and in all groups.

The record, detail tables and groups can also be expanded and collapsed using the arrow icon next to them.

Tip: Right-click a field and select **Copy** to copy its contents to the clipboard.

Changing the Data Model

By adding fields to a Data Model, setting their type and entering a (default) value, you may construct or modify a Data Model without having to create or open a data mapping configuration. This is especially useful when a solution does not require a data mapping configuration.

It isn't possible to enter data in a Data Model directly, but you could add JSON data (see "Adding JSON sample data" on page 937).

Modifying a Data Model can be done via the contextual menu. To open it, right-click the Data Model.

- **Add Field**: Click to add a new field at the current level (record or detail table). Enter the field name in the dialog; if you want you can change the field's data type and set a default value. Then click OK to add it to the Data Model.
- **Add Table**: Click to add a new detail table at the current level (record or existing detail table). Enter the table name in the dialog and click OK to add it.
- **Add Group**: Click to create a new group at the current level (record or existing group). Enter the group name in the dialog and click OK to add it. Any selected fields will be added to the group. (See also: "Ordering and grouping fields in the Data Model" on page 422.)

Note: Rename, **Delete** and **Set Type** are only available for newly added fields, and when data have been loaded from a file, without a data mapping configuration.

- **Rename**: Click to rename the selected table or field. Enter the new name and click OK to rename.
- **Delete**: Click to delete the selected table or field.
- **Set Type**: Use the list to select the field type (see "Data types" on page 436).
- **Ungroup**: Delete the selected group. The fields in the group will be moved up one level in the Data Model.
- **Default Value**: Click to set the default value for a field.
- **Move**: Click to move the selected field within the current level (record or group) in the Data Model. (See: "Ordering and grouping fields in the Data Model" on page 422.)
  - **Top**: Move the field to the first place.
  - **Up**: Move the field one step up.
  - **Down**: Move the field one step down.
  - **Bottom**: Move the field to the last place.
- **Copy**: copy the field's contents to the clipboard.

Preflight Results and Messages

Messages pane

The **Messages** pane is shared between the **DataMapper** and **Designer** modules and displays any warnings and errors from the data mapping configuration or template.

To open it in the Designer module, click the Messages button at the bottom right of the window (see "Designer User Interface" on page 1128).

Buttons

- **Export Log**: Click to open a **Save As** dialog where the log file (.log) can be saved on disk.
- **Clear Log Viewer**: Click to remove all entries in the log viewer.
- **Filters** : Displays the Log Filter.
- **Activate on new events**: Click to disable or enable the automatic display of this dialog when a new event is added to the pane.

Field headers

- **Message**: The contents of the message, indicating the actual error.
- **Component**: Whether the entry is a warning or an error.
- **Source**: The source of the error. This indicates the name of the step as defined in its step properties.
- **Date**: The date and time when the error occurred.

Preflight Results pane

The Preflight Results pane displays any notifications or errors related to the template, its scripts, its code or output generation. Double-click a script warning/error to open the script in the script editor. The relevant line will be highlighted.
See also: "Testing scripts" on page 1074.

Log Filter

The log filter determines what kind of events are shown in the Messages Pane.

- **Event Types** group:
  - **OK**: Uncheck to hide OK-level entries.
  - **Information**: Uncheck to hide information-level entries.
  - **Warning**: Uncheck to hide any warnings.
  - **Error**: Uncheck to hide any critical errors.
- **Limit visible events to**: Enter the maximum number of events to show in the Messages Pane. Default is 50.

Moving and merging panes

The OL Connect interface for both the Designer and DataMapper module is highly configurable. Each panel in the application can be moved, with the exception of the "The Data Viewer" on page 458 and"Workspace" on page 1261 which area always in a static location. All panels can be minimized or maximized.

To move a panel:

1. Click and hold the left mouse button on the panel title (tab) to move and keep the button pressed.
2. Start moving the mouse to the new location. A grey outline shows where the tab will show up:
   - A small grey outline next to a current panel tab indicates that both tabs will be at the same location and only the active tab will display its content.
   - At larger grey outline at one of the edges of the Workspace or Data Viewer indicates that the separate will be separate and always visible.
3. When the grey outline displays the location where the panel should be, release the mouse button.

To minimize a panel:

- Click the **Minimize panel** button at the top-right corner of the panel.

A minimized panel displays only as its icon wherever it was docked, generally on the left or right side, or the bottom.

To restore a minimized or maximized panel:

- Click the **Restore** button next to the panel's display icon.

The restored panel will return to its original docked location.

To temporarily display a minimized panel:

- Click the panel's display icon.

When another panel, menu or toolbar is clicked, the panel will be minimized again.

To maximize a panel:

- Click the **Maximize** button at the top-right corner of the panel.

A maximized panel takes the full available size for the panels. All other panels are minimized.

Outline pane

The Outline pane displays the current structure of the template, including all HTML tags present in each page.

- The display is in a treeview, the root being the *Page* node.
- At the top of the pane, a Text Filter box appears. Enter text in this box to only show elements which correspond to this inclusive filter. This can be class names, IDs, or element types (div, table, etc).
- Under the **Page** node, all top-level page elements are displayed. Each element under them is accessible by expanding (with the [+]) elements with children.
- Clicking on any element will select it in the Workspace, whether it displays the Source, Design or Preview tab.
- Dragging an element inside the Outline pane re-orders it in the actual HTML. Elements are executed top-to-bottom with lower elements appearing on top of previous elements (unless a CSS Z-Index is used).
- Right-clicking an element displays a contextual menu offering the following option:
    - **Delete Element**: Click to delete the element from the outline view. This also removes it in the template itself for the current section.

Resources pane

The Resources pane displays the resources that affect the template and its output.

Tip: The Text Filter box at the top of the Resources pane allows to look for text in the name of resources and in the source of any text-based files: HTML (including sections, master pages, and snippets), JSON, JS and CSS.

Tip: Images, fonts, stylesheets and snippets can be dragged or copied and pasted into the Resources pane to add them to your template.

Media

Media resources define paper handling configurations for Print output (see "Generating Print output" on page 1639 and "Print options" on page 1380) including page size and paper type. See "Media" on page 652 for more information.

Contextual menu

- **New Media**: Click to create a new Media and open its properties ("Media Properties" on page 1168).
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Properties**: Click to open the media properties.
- **Copy**: Click to copy the selected Media to the clipboard.
- **Paste**: Paste the copy into the Media folder.

Master Pages

Master Pages are layers of content that can be used by Print Contexts to provide a reusable static background of content. Only one Master Page can be selected for each page position in the context. See "Master Pages" on page 648 for more information.

Contextual menu

- **New Master Page**: Click to create a new Master Page and open its properties.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.
- **Properties**: Click to open the Master Page properties; see "Master Pages" on page 648 for more information.
- **Copy**: Click to copy the selected Master Page to the clipboard.
- **Paste**: Paste the copy into the Master Pages folder.

Contexts

Contexts hold the actual content of the template that is used to generate output. See "Context" on page 610 for more information.

Contextual menu (Context folder or individual contexts)

- **New Print Context**: Click to create a new Print Context with a single section.
- **New Web Page Context**: Click to create a new Web Page Context with a single section.
- **New HTML Email Context**: Click to create a new HTML Email context with a single section.
- **Properties...** (Print and Email Contexts): Click to open the Context's properties. See "Context" on page 610 for more information.
- **Paste**: Paste the copied section into the Context folder.

Sections

Sections hold part of the contents within a specific context. See "Sections" on page 612 for more information.

Contextual menu

- **Set as Default** (Email and Web contexts only): Click to set the default section that is output if none is selected in the output generation.
- **New Section**: Click to add a new section within the context.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.
- **Properties...**: Click to open the appropriate section properties. See "Section properties dialogs" on page 1199.
- **Includes...**: Click to open the "Includes dialog" on page 1157.
- **Finishing...** (Print sections only): Click to open the Finishing tab in the "Print section properties" on page 1200.
- **Sheet Configuration...** (Print sections only): Click to open the Sheet Configuration dialog; see "Master Pages" on page 648 and "Media" on page 652.

- **Copy**: Click to copy the selected section (including its settings) to the clipboard.
- **Paste**: Paste the copied section into the same Context.

Images

Images are graphical elements that can be added to the page for display, either statically or dynamically. See "Images" on page 852 for more information.

Contextual menu

- **New Folder**: Click to create a new folder to organize resources more easily.
- **Rename**: Click to open the resource's Rename dialog. This is the same as pressing the F2 key while the resource is selected.
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.
- **Copy**: Click to copy the selected file.
- **Copy Resource Location**: Click to copy the relative path of the resource to the clipboard. It may then be pasted into a script.

Fonts

Font Resources included in a template are transported with it, so they can be accessed even if the template is moved to a different computer. Fonts appear in the fonts drop-down menu but they can also be set through a CSS Stylesheet.

Currently supported font types: otf, woff, ttf, svg. Fonts must be set to *installable* to be useable in the output.

See "Fonts" on page 917.

JavaScripts

JavaScripts are scripted programs that can run on Web output when added to the page header. See "Using JavaScript" on page 703 for more information.

JavaScript files that are linked to (i.e. included in) a section show a chain icon.

Contextual menu

- **New Javascript**: Click to create a new JavaScript resource.
- **New Remote Javascript**: Click to add a Remote JavaScript resource. See "Using JavaScript" on page 703 for more information.

- **New Folder**: Click to create a new folder to organize resources more easily.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Download Resource** (remote JavaScript files only): Select to download the Remote JavaScript file (see "Adding a remote JavaScript file" on page 705) and add the downloaded copy to the Scripts folder. The copy will have the same name as the remote script file, but its extension will be .js instead of .rjs. Any section that used the JavaScript file will from then on use the downloaded file (see "Including a JavaScript file in a Web context" on page 706).
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.

Stylesheets

Style sheets control how contents appears on the page. They define spacing, color, size and other properties of elements on the page. See "Styling templates with CSS files" on page 881 for more information.

In case the CSS file has references to specific images, you can drag/drop or copy/paste those images into the Stylesheets folder as well.

Style sheets that are linked to (i.e. included in) a section show a chain icon. See "Applying a style sheet to a section" on page 888.

Contextual menu

- **New Stylesheet**: Click to create a new Stylesheet resource. Adding a new stylesheet will automatically include it in the currently active section.
- **New Remote Stylesheet**: Click to add a Remote Stylesheet resource. See "Styling templates with CSS files" on page 881 for more information.
- **New SCSS file**: Click to add a new, empty Sass file to the Resources (extension: .scss). **Sass** is a CSS Preprocessor integrated in Connect. For more information about Sass, see: Sass website.

  > **Note:** When the name of Sass file begins with an underscore, it is considered a partial .scss file (e.g. _mySass.scss). Partial files are typically imported in a base .scss file. They may include Sass variables or other directives declared in the base file, and they cannot be compiled.

- **New Remote SCSS file**: Click to add a Remote Sass file to the Resources.

- **Compile** (.scss files only): Select to compile the Sass (.scss) file into a Stylesheet which can then be applied to a section.
  The compiled style sheet will have the same name as the Sass file, but with the extension .css.
  Compiled CSS files can be recognized by their first line: /* Compiled by https://sass-lang.com/libsass */.
  Compiler options can be set in the Preferences; see "Editing preferences" on page 1044.

  > **Note:** Re-compiling a .scss file overwrites any manual changes made to the .css file.
  >
  > Partial .scss files cannot be compiled.

- **New Folder**: Click to create a new folder to organize resources more easily.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Download Resource** (remote style sheets only): Select to download the Remote Stylesheet (see "Using a remote style sheet" on page 884) and add the downloaded copy to the Stylesheets folder. The copy will have the same name as the remote style sheet, but its extension will be .css instead of .rcss. Any section that used the remote style sheet will from then on use the downloaded style sheet (see "Applying a style sheet to a section" on page 888).
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.

Snippets

A snippet is a small, ready-to-use piece of content in a file. Snippets can be re-used within the same template, in all contexts, sections and scripts.

Contextual menu

- **New HTML Snippet**: Click to create a new HTML Snippet resource; see "Snippets" on page 867.
- **New JSON Snippet**: Click to create a new JSON Snippet resource; see "JSON snippets" on page 872.
- **New Handlebars template**: Click to create a new Handlebars template; see "Handlebars templates" on page 1023.

- **New Folder**: Click to create a new folder to organize resources more easily.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.
- **Copy**: Click to copy the selected file.
- **Copy Resource Location**: Click to copy the relative path of the resource to the clipboard. It may then be pasted into a script.

Translations

A **PO file** holds translated texts for the template in one language. Each PO file has an entry by which OL Connect recognizes the target language.
PO files can be stored in the Translations folder.

See "Translating templates" on page 1120 and "Importing translations (.po)" on page 1127 for more information.

Contextual menu

- **New Translation**:
  - **PO files...**: Select a PO file to import it into the template.
  - **Remote PO file**: Enter the URL Click to add a remote PO file. See "Using JavaScript" on page 703 for more information.
- **New Folder**: Click to create a new folder to organize resources more easily.
- **Rename**: Click to open the resource's Rename. This is the same as pressing the F2 key while the resource is selected.
- **Download Resource** (remote JavaScript files or translation files): Select to download the remote file and import it into the template. The copy will have the same name as the remote script file, except for its extension: .rjs becomes .js, .rpo becomes .po.
  In case of a JavaScript file, any section that used the remote file will from then on use the downloaded file (see "Including a JavaScript file in a Web context" on page 706).
- **Delete**: Click to delete the resource. This is the same as pressing the Delete key while the resource is selected.

Scripts pane

The Scripts pane contains all of the scripts that are used to replace data in a template, or to modify its look; see "Personalizing content" on page 923.

The scripts are stored in three groups: Control, Standard and Post Pagination, depending on when they run (see "The script flow: when scripts run" on page 1082).

> **Note:** The scripts in each of the main folders - Control, Standard and Post Pagination - are executed top-to-bottom. They can be dragged up or down in the pane to change their order of execution. For example, content-loading scripts (snippets with variable data, for instance) must come before scripts that replace data within loaded contents.

The Script pane allows to add and edit scripts and to manage scripts, or instance by organizing them in subfolders and enabling/disabling them (see "Managing scripts" on page 1071).

Scripts can be exported and imported via the buttons or through drag & drop between the Scripts pane and any location on the computer.

Scripts have access to the whole OL Connect JavaScript API (see "Standard Script API" on page 1481), such as the `record` object.

> **Note:** Scripts included on the Scripts pane are completely distinct from the JavaScript resources found in the "Resources pane" on page 1250 (see "Using JavaScript" on page 703).
> Think of scripts as server-side in the sense that they are executed through the Connect modules (Server and Content Creation especially). JavaScript resources don't have access to the OL Connect JavaScript API. They are only executed **after** the content creation is done, generally in a browser.

Buttons

- **Debug Scripts...** : Opens the "Script Debugger" on page 1182.
- **View Menu** : Displays a drop-down that shows the following options:
  - **Import...**: Click to open a standard Open dialog to import a script. The script must have the .OL-script extension.
  - **Export...**: Click to open a standard Save As dialog to save the currently selected scripts to disk. These scripts can be re-used in other templates. If more than one script is selected, they are all saved to a single file. If some scripts are inside folders, this folder structure is kept and will be restored when the scripts are imported.
- **New**: Displays a drop-down that shows the following options:

- **Control Script**: A Control Script affects the output of a template per record as a whole, instead of parts of the content. See "Control Scripts" on page 1098 and "Control Script API" on page 1591.
- **Remote Control Script**: Opens a dialog to add a remote Control Script. See "Control Scripts" on page 1098 and "Control Script API" on page 1591.
- **Standard Script**: Adds a new empty standard script. Double-clicking the new script opens the script editor, where you can write code (see "Writing your own scripts" on page 1064).

  > **Tip:** Select **Text Script** instead of Standard Script to create a Standard script that is opened with the Text Script Wizard.

- **Post Pagination Script** (Print context only): Adds a Post Pagination script, which runs on the Print context after the output has been paginated (see "Post Pagination Scripts" on page 1114).
- **Folder**: Adds a folder in which scripts can be placed for easier management. See "Script folders" on page 1072.
- **Text Script**: This type of script inserts variable data in the text. See "Variable data in text: scripts and placeholders" on page 945.
- **Dynamic Image script**: Provided that its selector refers to an image, this script dynamically changes the image for each record. See "Dynamic images" on page 972.
- **Dynamic Background Script**: This type of script dynamically changes the background of a Print section. See "Dynamic Print section backgrounds" on page 994.
- **Email scripts** (Email sections only): Email scripts define the sender, recipients, subject, dynamic attachments etc. of the email that is sent, and the PDF password. See "Email header settings" on page 672.
- **Conditional Content Script**: This script can conditionally show or hide any element in the template. See "Showing content conditionally" on page 958 and "Conditional Content script dialog" on page 1197.
- **Conditional Print Section Script**: This script conditionally shows or hides any Print section in a template. See "Conditional Print sections" on page 962.

- **Collapse All**: Collapses all the folders, hiding the scripts inside of them.

Scripts Pane filter

To quickly find a certain script or piece of code, type a search text in the filter field located below the buttons in the Scripts pane.
The Scripts Pane will then only display scripts of which the name, selector or content matches the search text. This is case insensitive.

Scripts Pane column

- **Name**: The name added to better identify the script.
- **Selector**: Displays the initial text or selector that the script applies to. See "Selectors in OL Connect" on page 1083. Control Scripts and Post Pagination Scripts do not have a selector (see "Control Scripts" on page 1098 and "Post Pagination Scripts" on page 1114).

Tip: Fields from the Data Model pane can be dragged directly into the Scripts pane to create a Text Script. Additionally, Text scripts can be dragged into any section to add the script's placeholder at the insert location. See "Variable data in text: scripts and placeholders" on page 945.

Contextual menu options

- **New**: Click to create a new script or folder. See "Buttons" on page 1257 for a description of all options. Which script types are available depends on the current section and script folder.
- **Duplicate**: Click to create an exact copy of the script.
- **Delete**: Click to delete the selected script. This does not delete any element or text in the template itself.
- **Rename**: Click to open a dialog to rename the script. This is the same as changing the **Name** field in the Edit Script window, which can be opened by double-clicking the script.
- **Enable/Disable**: Click to trigger the script to be enabled or disabled. Disabled scripts are greyed out and italic and will not be executed. See "Enable/disable scripts" on page 1073
- **Import**: load a script from a Scripts file (*.OL-script).
- **Export**: save the script to a Scripts file (*.OL-script).
- **Properties** (Script folders only): edit the **name** and **execution scope** of the folder. See "Execution scope" on page 1072.

Styles pane

The Styles pane shows which CSS style rules apply to the currently selected element. Hovering over elements, parent elements, styles and selector parts in the Styles pane highlights the affected elements in the main editor.
A link next to a style rule will open the file where that particular style is defined. This can be either a CSS file or the source file of a section if local formatting was used (see "Styling and formatting" on page 879).

A crossed-out style rule signals that it was overruled by another style rule. This happens when:

- A more specific, and therefore more important rule, is encountered for the same element. See "Using a more specific CSS rule" on page 890 to learn more about the specificity of style rules.
- A rule with the same importance is read after the first rule. Not only is the order of the rules in a CSS file important, but also the order in which the style sheets are read. The style sheets that are included with a section are read in the specified order; see "Applying a style sheet to a section" on page 614.

Translations pane

The Translation pane contains translation entries. They can be exported to a POT file, which can be opened in a translation tool (see "Translating templates" on page 1120).

> **Tip:** Hover over a translation entry to see which HTML elements in the template have the same text and are tagged for translation (see "Tagging elements for translation" on page 1122).

Buttons

- **Tag Element...**: Tag the currently selected element in the template for translation (see "Tagging elements for translation" on page 1122). If no translation entry with the same text already exists, the text of the tagged element is copied to a new translation entry.
- **Add Empty String**: Adds a new, empty translation entry to the list and opens the "Translation String Options dialog" on page 1222.
- **Export**: Exports the translation entries to a POT file (see "Exporting and importing translation files" on page 1126).
- **Highlight Tagged Elements**: Turn highlighting on or off. Highlighting shows which HTML elements have the same text and are tagged for translation when you hover

over a translation entry.

- **Sync**: Scans the template (all sections and snippets) for elements that are marked for translation and compares them to the list of translation entries. The results are shown in the "Sync Summary dialog" on page 1214.
- **Edit locale**: Opens the "Locale Settings" on page 1167 dialog.

Columns

- **Source**: The text that will be presented to the translator, and that will be replaced with a translation when OL Connect generates output from the template.
- **Context**: Describes where the source text is used, to help the translator translate the source text correctly in case it should be translated differently in different locations (see "Same text, different translation: add a context" on page 1123).

Contextual menu options

- **Edit String**: Opens the "Translation String Options dialog" on page 1222 for the clicked translation entry.
- **Delete**: Deletes the clicked translation entry and (optionally) removes the translation tag from matching elements.

Workspace

The Workspace pane is where everything comes together. It is the contents of the page, the WYSIWYG editor that shows what the output will look like.

The Workspace contains three tabs, or four, when a Web section is open. To switch between the tabs, click on the tab at the bottom, or select **View > Design View**, **Preview View** or **Source View** on the menu.

A lock icon displayed next to the file name indicates that the file is read-only. Saving a read-only file opens the Save as dialog, allowing you to save the file under a different name. The new file is then opened and can be edited in the Designer.

For an overview of keyboard shortcuts, see "Keyboard shortcuts" on page 1223.

Design tab

The Design tab shows the template including all styles, text and images as well as the placeholders used for variable data. In this tab, the template's scripts are not executed and only placeholders are shown.

Normally you would only edit a template in Design mode.

The top of the Design tab contains an area with the following options and buttons:

- **Breadcrumbs**: Displays the element type where the cursor is located and any of its parent elements. Elements with classes or IDs show these details next to them, for instance `div #contents > ol.salesitems > li`. Click on an element in the Breadcrumbs to select it.
  When an element is selected in the breadcrumbs and the Backspace or Delete key is pressed, that element is deleted.
  If the deleted element was targeted by a script, you will be asked if you want to delete the script as well.
- **Context Selector**: Displays the current context. The drop-down lists available contexts. Clicking on a context switches to that context.
- **Section Selector**: Displays the currently active section. Clicking on another section switches to that section.
- **Media Selector** (Master Page editor only): Displays a list of Media resources. Clicking on a media will display its Virtual Stationery background while in Preview mode.
- **Zoom Level**: Displays the current zoom level and drops-down to change the level.
- **Buttons**
  - **Zoom in**: Zooms in by 25%
  - **Zoom out**: Zooms out by 25%
  - **Actual Size**: Zooms to 100%.
  - **Fit Width**: Adjusts zoom to fit the exact width of the template to the available workspace.
  - **Refresh**: Reloads the view, including static external images, remote stylesheets, and remote scripts, and re-runs the scripts (the latter in Preview Mode only).
  - **Responsive Design View**: Use the drop-down to select a specific screen width, to test the design for different devices. Not available in Print contexts.

Tip: You can also use the mouse's scroll wheel in combination with the Ctrl button to gradually zoom in or out.

When a template is open, the workspace also shows a ruler and guides (see "Guides" on page 899).

Preview tab

The Preview tab shows the section as it will output with the current record (see "Loading data" on page 926), showing the personalized content (see "Personalizing content" on page 923). Content added by a script isn't visible in the Design tab, but is visible and can be inspected in the Preview mode.

Content cannot be edited in Preview mode if the option **Evaluate Handlebars expressions** is enabled for a section (which is the case by default in a new template; see "Section properties dialogs" on page 1199).

Source tab

The source tab displays the HTML source of the section, including HTML Headers, CSS and HTML code. The source is displayed in a color-coded text editor, to quickly visualize the code.
In this tab changes and adjustments can be made to the code. When editing a template in Source mode you should write HTML. For shortcuts that can be used in this editor, see "Text editors: Source tab, JavaScript, CSS, Script Editor" on page 1227 and "Emmet" on page 662.

To the left of the Source tab, a bar helps visually identify the start and stop of an element. For example when clicking on the opening <table> element, this bar marks the whole <table> and all its contents, until the ending </table> tag.

Pretty print options

In the Source view the HTML source of the template is "pretty printed" (that is: formatted, adding new lines and indentation) to make it more readable.
When this is undesirable, the Source view formatting can be turned off for (parts of) a section by adding <!-- format:off --> in Source view, at the beginning of the text or in between two HTML elements. From that point on, pretty printing will be disabled for that section. Use <!-- format:smart --> or <!-- format:auto --> to turn the formatting back on.

These are all format options:

- <!-- format:off --> turns the formatting off.
- <!-- format:smart --> or <!-- format:auto --> turns the formatting back on.
- <!-- format:collapse --> forces the following HTML elements to be collapsed.
- <!-- format:expand --> forces the following HTML elements to be expanded.

Example

This example shows how to turn pretty printing off for one specific HTML element: a Div element.

```
<p>In a "pretty printed" paragraph
        <br>text is indented automatically on the Source tab.
</p>
<!-- format:off --><div anchor="page_media_0"
style="font-family: Lucida Console,monospace;
line-height: 1.2; white-space: pre-wrap; position: absolute; overflow: hid-
den;
-moz-box-sizing: border-box; width: 675px; height: 68px; top: 209.967px;
left: 65.4px;" offset-x="103.19999694824219" offset-y="247.76666259765625">
@Header1@ </div><!-- format:auto-->
```

Live tab

The Live tab shows the result of the template as rendered by the Gecko rendering engine. It is a good indication of how an HTML template would display in a visitor's browser, especially if they are using FireFox (which uses the Gecko engine).

Toolbars

This topic lists the buttons that are available in the top toolbar of the Designer module. For a description of the buttons at the top of the Workspace, see "Workspace" on page 1261.

File manipulation

- **New**: Displays the New Wizard where a new data mapping configuration or a new template can be created.
- **Open**: Displays the Open dialog to open an existing template.
- **Save**: Saves the current template. If the template has never been saved, the Save As... dialog is displayed.
- **Print**: Opens the Print Output dialog.
- **Proof Print**: Opens the "Print options" on page 1380 dialog as a "Proof Print" which limits the number of records output. The options themselves are identical to the regular Print Output dialog.

Output

- **Send Email**: Opens the Send Email dialog; see "Send (Test) Email" on page 1204.
- **Send Test Email**: Opens the Send Test Email dialog; see "Send (Test) Email" on page 1204.
- **Preview HTML**: Opens the current template's Preview in the system default browser. Useful for testing scripts and HTML output.
- **Send to Workflow**: Opens the "Send to Workflow dialog" on page 1206 to send files to a local Workflow software installation.
- **Send to Server**: Opens the Send to Server dialog to send files to a Connect Server.
- **Send COTG Test**: Click to open the Send COTG Test dialog, to send the current Web Context to the Capture OnTheGo Application. See "Testing a Capture OnTheGo Template" on page 735.
- **Get Job Data File on submit**: Click to enable/disable. When enabled, the Job Data File will be returned to Connect Designer directly after a COTG Form has been submitted (see also: "Using COTG data in a template" on page 712).
- **Add Dummy Data**: Populate empty form fields with dummy data. See "Testing a Capture OnTheGo Template" on page 735.

Forms

- **Insert Form**: Inserts a <form> element.
- **Insert Fieldset**: Insert a <fieldset> element.
- **Insert Text Field**: Inserts a <input type="text" /> element. A drop-down is available to insert other fields, such as a URL, Password etc.
- **Insert Text Area Field**: Inserts a <textarea> element.
- **Insert Label**: Inserts a <label> element.
- **Insert Checkbox**: Inserts a <input type="checkbox"> element.
- **Insert Radio Button**: Inserts a <input type="radio"> element.
- **Insert Select Field**: Inserts a <select> element and add multiple possible options to it.

- **Insert Button**: Inserts a <button type="submit"> element at the current cursor location.

Pagination (Print Context)

- **Page Number**: Inserts a placeholder for the current page number
- **Page Count**: Inserts a placeholder for the total number of pages in the current section.

Guides

- **Insert Horizontal Guide**: Click to insert a new horizontal guide; see "How to position elements" on page 897.
- **Insert Vertical Guide**: Click to insert a new horizontal guide; see "How to position elements" on page 897.

Form Wizard

- **Form Wizard**: Click to open the Form Wizard to add a form to a Web Context. See "Forms" on page 842 and "Form Elements" on page 847.
- **Validation Settings**: Click to open the Validation settings dialog to change the validation settings on the currently selecting tools. See "Forms" on page 842.

Table manipulation

- **Insert Standard Table...**: Inserts a table with a specific number of columns and rows through the "Table" on page 860 Wizard.
- **Insert Dynamic Table...**: Inserts a Dynamic Table where the number of rows is determined by a detail table in the data, through the "Dynamic Table" on page 974 Wizard.
- **Select**
  - **Select Table**: Selects the table where the cursor is located. If the cursor is within a table embedded within another, the innermost table is the one selected.
  - **Select Row**: Selects the innermost row where the cursor is located.
  - **Select Cell**: Selects the innermost cell where the cursor is located.
- **Delete**

- **Delete Table**: Deletes the innermost table where the cursor is located.
- **Delete Row**: Deletes the innermost row where the cursor is located.
- **Delete Column**: Deletes the innermost cell where the cursor is located.

- **Insert**
  - **Insert Row Above**: Inserts a row above the current one. The row configuration, such as merged cells and cell styles, are duplicated, but contents is not.
  - **Insert Row Below**: Inserts a row below the current one. The row configuration, such as merged cells and cell styles, are duplicated, but contents is not.
  - **Insert Column Before**: Inserts a column to the left of the current one. The column configuration, such as merged cells and cell styles, are duplicated, but contents is not.
  - **Insert Column After**: Inserts a column to the right of the current one. The column configuration, such as merged cells and cell styles, are duplicated, but contents is not.

Objects

- **Insert Image...**: Inserts an Image using a resource that is local to the template, at the current location of the pointer and opens its properties. See "Images" on page 852.
- **Insert Barcode**: Displays a list of available barcodes. Click on one to insert it on the page. See "Barcode" on page 768.
- **Insert Pie Chart**: Click to insert a new Pie Chart object and open the Chart Script wizard.
- **Insert Bar Chart**: Click to insert a new Bar Chart object and open the Chart Script wizard.
- **Insert Line Chart**: Click to insert a new Line Chart object and open the Chart Script wizard.

Hyperlinks

- **Insert Hyperlink...**: Creates a Hyperlink or mailto link on the currently selected text or element and opens its properties. See "Hyperlink and mailto link" on page 850.

- **Remove Hyperlink**: Removes the currently selected hyperlink. The text or element that was the hyperlink is not removed.

Boxes

- **Insert Positioned Box**: Inserts an absolute-positioned box on the page, which can be moved around freely.
- **Insert Inline Box**: Inserts an inline box that is set to float to the left, at the position of the cursor.
- **Wrap in Box**: Takes the current selection and wraps it inside a new box.
- **Float Left**: Floats the current element to the left using a *float:left* style.
- **No Float**: Removes any *float* style applied to the currently selected element.
- **Float Right**: Floats the current element to the right using a *float:right* style.
- **Rotate Counter Clockwise**: Rotates the currently selected box 90° counter-clockwise.
- **Rotate Clockwise**: Rotates the currently selected box 90° counter-clockwise.

Align Objects

- **Top**: Aligns the top side of the selected objects with the top edge of the last selected object.
- **Middle**: Aligns the middle of the selected objects with the middle of the last selected object. Objects may move up or down.
- **Bottom**: Aligns the bottom side of the selected objects with the bottom edge of the last selected object.
- **Left**: Aligns the left side of the selected objects with the left edge of the last selected object.
- **Center**: Aligns the center of the selected objects with the vertical center of the last selected object. Objects may move to the left or to the right.
- **Right**: Aligns the right side of the selected objects with the right edge of the last selected object.

Styles

- **Element Type**: Displays the element type of the selected element and drops down to show other element types in which it can be changed.
- **Style**: Displays the style of the selected element and drops down to show other available styles which can be applied to it.
- **Font Face**: Displays the font face of the selected text or element where the cursor is located and drops down to show other available font faces which can be applied to it. Fonts added to the Fonts folder of the Resources pane are shown automatically in the Fonts drop-down.
- **Font Size**: Displays the font size of the selected text or element where the cursor is located and drops down to show other available sizes which can be applied to it.
- **Font Color**: When text is selected, click to apply the shown color to the selected text, or use the drop-down to change the color and apply it.

Alignment

- **Align Left**: Aligns the currently selected element to the left.
- **Align Center**: Aligns the currently selected element to the center.
- **Align Right**: Aligns the currently selected element to the right.
- **Justify**: Aligns the currently selected element to stretch text lines to fill all available width.

Text editing

- **Bold**: Makes the currently selected text **bold**.
- **Italic**: Makes the currently selected text *italic*.
- **Underline**: Makes the currently selected text underline.
- **Strikethrough**: Makes the currently selected text ~~strikethrough~~.
- **Create Numbered List**: Makes the selected text element a numbered list (<ol>). If multiple paragraphs are selected, each becomes a list item (<li class="Bullet">).
- **Create Bulleted List**: Makes the selected text element a bullet list (<ul>). If multiple paragraphs are selected, each becomes a list item (<li class="Bullet">).

- **Indent**: Increases indentation of the selected text element. If the element is a paragraph, it is wrapped in a <blockquote> element. If it is a list item, it is moved to a child level, creating a new list if necessary.
- **Outdent**: Decreases indentation of the selected text element. If the element is wrapped in a blockquote element, one blockquote is removed. If the element is a list item, it is removed from one surrounding list.
- **Superscript**: Makes the currently selected text a $^{\text{superscript}}$.
- **Subscript**: Makes the currently selected text a $_{\text{subscript}}$.
- **Remove Formatting**: Remove any and all styles, text decorations and other formatting from the selected text. Indentation is not affected.

Miscellaneous

- **Insert Lorem Ipsum**: Inserts a paragraph of generic lorem ipsum text, useful for placeholder or template design.
- **Show Edges**: Shows a colored border around elements on the page and the type of element that is highlighted.
- **Welcome Screen**: Click to re-open the Welcome Screen.

Welcome Screen

The **Welcome Screen** appears when first starting OL Connect. It offers useful shortcuts to resources and to recent templates, data mapping configurations, and print presets.

If you are new to OL Connect and you don't know where to start, see "Welcome to OL Connect 2025.1" on page 15.

The Welcome Screen can be reopened in two ways:

- The **Home** button at the top right in the "Toolbars" on page 1264.
- From the Menus in **Help**, **Welcome Screen**.

To go back from the Welcome Screen to the template or data mapping configuration that you were working on:

- Close the Welcome Screen by clicking the cross next to the text 'Welcome' at the top.

Contents

Top bar:

- **OL Connect**: Opens the OL Connect website.
- **News** (bell icon): Displays a list of what's new in OL Connect.
- **Support**: Brings you to the customer portal login page.
- **Software activation**: Opens the Objectif Lune Web Activation Manager.

Menu on the left:

- **Home**: Resets the Welcome screen to its start page. The start page has:
  - One link to a blog post or tutorial in the Upland Objectif Lune resource center. To see more, click **Learn**.
  - A list of recently opened files.
- **New**: Lets you create a new OL Connect file or sample project.
  - **Print**: Start designing a template for print output. See "Creating a Print template with a Wizard" on page 618.
  - **Email**: Start designing a template for email output. See "Creating an Email template with a Wizard" on page 664.
  - **Web**: Start designing a template for a web page. See "Creating a Web template with a Wizard" on page 683.
  - **Data**: Start to build a data mapping configuration with a wizard; see "Data mapping configurations" on page 337.
  - **Project**: Displays a list of available Sample Projects, producing a complete Connect project; see "Sample Projects" on page 1185.
  - **Presets**: Start creating "Job Presets" on page 1646 and "Output Presets" on page 1648 to configure your print jobs.

  Tip: Scroll down to **Online resources** to download a ready-made file with existing demo content as a starter for your file.
- **Open**: Lets you open an existing template, data mapping configuration, job preset, output preset, or package file.

Online resources:

- **Learn**: Displays a list of the most recent blog posts and tutorials in the Upland Objectif Lune resource center. Click a link to open the Objectif Lune resource center and read the article online.
- **Forum**: Ask your questions and share tips and tricks in the user forum (https://olforum.uplandsoftware.com/).
- **Online Help**: Opens this documentation.

Files:

- **Printer configurations**: Lists printer definition files (PDC) available for download. The selected files will be downloaded to the current user's Connect\workspace\configurations\ folder and can be imported via the Print Wizard; see "Adding print output Models to the Print Wizard" on page 1652.
- **Finishing configurations**: Lists HCF files available for download. High Capacity Feeders (HCF) are also commonly referred to as Inserters or Folder-Inserters. The Inserter options in the print settings are dependent upon the HCF model selected; see "Inserter options" on page 1464.
- **My license**: Shows your current license's details.

Print options

The **Print Options** page is the first page of the **Advanced Print Wizard** (used for both Production and Proof printing in the Designer) and the second page of the "Output Presets" on page 1375 wizard.

This page is the most important of both the Advanced Print Wizard and the ***Output Preset*** Wizard. All other pages that appear in the Wizard are determined by the selections made on this page.

The options that don't appear in the ***Output Preset*** Wizard (as noted below), do appear in the ***Job Preset*** Wizard. Only the Advanced Print Wizard contains all the options, but it cannot save your settings to file like the preset wizards allow.
For more information about print preset files, see "Print Presets" on page 1644.

The choices can be broken down as follows.

Printer

- **Model**: Use the drop-down to select the printer language that will be generated.

  OL Connect comes with several bundled print output Models which cover a large range of industry standard print output types. Depending on the license, these include

AFP, IPDS, PCL (if licensed), PDF, PPML and PostScript (including the optimized VIPP and VPS variants), with a range of quality settings available for each.

> **Note:** AFP and IPDS are only available in OL Connect Enterprise edition.

The bundled print output Models do not cater for *all* types of printers, however. There are simply too many different printers with their own individual (often quirky) settings and capabilities to cater for all of them with just a few standardized print Models.

To use OL Connect with a particular printer Model you must obtain a tailored OL Connect Printer Definition that defines that printer and its capabilities for OL Connect. For example, to cater for IPDS or AFP printers that support Encapsulated Postscript resources included through an Include Object (IOB) structured field. Such a print output Model would need to be customized for you. These customized Printer Definitions will be created by Upland OL Support, in conjunction with information you provide about both your printer and your specific needs. Support will then provide you with a customized Printer Definition that supports the desired functionality with your specific printer. By default, OL Connect initially displays just a single generic *PDF output* option, to keep the interface clean. Other print output types can be added to the Printer Model drop-down list at any time via the **Import Definitions** button.

The **Import Definitions** options are:

- **Import Definition**: Used to add any customized or tailored Printer Definition that Upland Software, Inc. has prepared for you, specific to your printer.
- **Create Definition from PPD**: Used to create your own customized PostScript Printer Model through the printer manufacturer provided PostScript Printer Definition (PPD) file. This option launches the "Dynamic PPD Options" on page 1468 page later in the Printer Wizard, which allows you to setup the printer.
- **Edit available printers**: Add extra Printer Definitions from the selection that come installed with OL Connect. Adding a printer from this list is probably all you will ever need.
- **Export printer definition from output preset**: Use to extract a Printer Definition from a new or modified Printer Model. This option is only available when the Printer Model currently loaded differs from those previously saved or installed.

For more information on how to add Printer Definitions to OL Connect, see "Adding print output Models to the Print Wizard" on page 1652.

Output options

- **Output Local**: Select to have the output created using the local Print Server, instead of the OL Connect Server (see "The OL Connect server" on page 18 for an explanation).
- **Output Type** choices:
  - **Prompt for file name**: Select to output to a local file on the hard drive. When this option is selected, no other configuration is necessary. A Save As dialog will appear to allow selection of the folder and file name.
  - **Directory**: Select to output to a local folder on the machine.
    Selecting this will open the **Directory Options** sub-group, which has these options:
    - **Job Output Mask**: The name of the file that will output.
      You could write the Job Output Mask directly into this edit box (for a list of available variables see "Print output variables" on page 1659), or you could create a Mask via the Options button. This opens the custom dialog: "Job Output Mask Dialog " on page 1385.
      The Job Output Mask may contain (dynamic) folder names, for example: `${document.metadata['Country']}\${template}`.
      The evaluated value of the Job Output Mask is taken as a path **relative** to the folder specified by the Job Output Folder (the next option in this dialog). The Job Output Folder must exist, but folders specified in the Job Output Mask will get created if they don't exist.
      Spaces inside the Job Output Mask (i.e., not at the beginning or end) are allowed, but any leading and trailing spaces are trimmed from the Job Output Mask to prevent errors.
    - **Job Output Folder**: The path on the disk where the file is produced. Please note that the folder must exist, or output will fail when produced through the server.
  - **LPR Queue**: Select to send the print job to an LPR queue. It is assumed that the print technology is supported by the system receiving the LPR job.
    - **Local Printer**: The IP or host name of the printer or machine where the LPD is installed and will receive
    - **Queue Name**: The queue name that will accept the job on the LPD. Default is generally "auto".
    - **Job Owner Name**: Optional entry for adding the name of the job owner.

- **Job Name**: The name of the output file. You can use ${template} as a variable for the name of the Designer Template used to generate the output. (See also: "Print output variables" on page 1659.)
- **Timout (seconds)**: Set the timeout duration, in seconds. Default is 60 seconds.
- **Retry Attempts**: Set the amount of job retry attempts. Default is twice.

- **Windows Printer**: Select to send the Print Job to a printer queue (note that this is an actual printer queue, not a Workflow Printer Queue).
  For non-PDF output, the job will be first rendered as a PDF before being printed through the Windows driver.

  For PDF output the Adobe Library is used.

  - **Windows Printer**: Use the drop-down to select the Windows printer queue where the job will be sent.
  - **Job Owner Name**: Optional entry for adding the name of the job owner.
  - **Job Name**: The name of the output file. You can use ${template} as a variable for the name of the Designer Template used to generate the output.
  - **PDF Rendering Options** sub-group (only available for *PDF output*):
    - **Auto-rotate and center**: Check to automatically select the page orientation that best matches the content and paper.
    - **Choose paper source by page size**: Check to use the PDF page size to determine the output tray rather than the page setup option. This option is useful for printing PDFs that contain multiple page sizes on printers that have different-sized output trays.
    - **Scale**: Chose from one of the following scaling options:
      - **None**: Select to not scale any page, whether it fits or not.
      - **Expand to printable area**: Select to expand any page to fit the page area. Pages larger than the paper size are not re-sized.
      - **Shrink to printable area**: Select to shrink any page to fit the page area. Pages smaller than the paper size are not re-sized.
    - **Prompt for Printer Properties**: Select to run the Windows Driver

preferences dialog as part of the printing process, to allow selection of printer functionality.

Production Options

- **Runtime Parameters**: select "Runtime Parameter Options" on page 1360 to store information which can be used at runtime for comparisons against conditions within Job Creation, or for use with external sorting programs.
- **Override Finishing options** (not available in ***Output Preset*** Wizard): Check to configure custom "Finishing options" on page 1366, such as binding.
- **Booklet Imposition**: Check to tell the printer to generate a booklet for the print output. Booklet options are set in the "Booklet Options" on page 1454 page.
  This option is unselected by default unless selected in the Designer "Print section properties" on page 1200.
- **Separation**: Check to activate the "Separation options" on page 1477 page of the wizard.
- **Add Inserter marks**: Check to enable Inserter mark functionality, which is set in the "Inserter options" on page 1464 page.
- **Use grouping** (not available in ***Output Preset*** Wizard): Check to configure grouping of output into jobs, job segments or document sets. See "Grouping options" on page 1372.
- **Include meta data** (not available in ***Output Preset Creation*** Wizard): Check to add meta data to the output. This can be done at Job, Job Segment, Document, Document Set and Page level. See "Meta Data options" on page 1374.
- **Imposition**: Check to enable Cut & Stack Imposition, which is set in the "Imposition options" on page 1456 page.
- **Add additional content**: Check to activate the "Additional Content" on page 1409 page of the wizard.
- **Print virtual stationery**: Check to enable virtual stationery in the output.

Records

(Not available in ***Output Preset*** Wizard.)

- **Record Range**: Allows selection of a range of records or a custom selection.
  You can specify individual records separated by semi-colons (;) or ranges using dashes.
  For example: 2;4;6-10 would print pages 2, 4, 6, 7, 8, 9 and 10.

- **Apply filtering and sorting to record selection**: Check to activate the "Data filtering options" on page 1363 (to filter out certain records) and "Sorting options" on page 1368 (to sort the remaining records) pages of the wizard.

Copies

(Not available in ***Output Preset*** Wizard.)

- **Copies**: Enter the number of copies to print, of each record.
- **Collate**: When printing multiple copies you can check this checkbox to have the record copies printed together.
  For example, in a three record job the records would print out as 1-1-2-2-3-3 rather than 1-2-3-1-2-3.

Pure Color Thresholds

This section is valid for AFP, IPDS (Enterprise edition only)and PCL only. It applies to elements within the record that are shades of grey, rather than black or white.

- **Black Threshold Percentage**: The percentage of shading at which the element will appear as full black, rather than dark grey.
- **White Threshold Percentage**: The percentage at which the element will appear as full white, rather than light grey.

Advanced Print Wizard navigation options

- **Load** button (not available in ***Output Preset*** Wizard): Click to select a previously created Output Preset.
  This will change the Advanced Print Options to match the entries contained within the Preset.
- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1272 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings.
  This can be done at any point within the Wizard, whether or not *all* the options

selected in the "Print options" on page 1272 page have been completed or not.

- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Booklet Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Booklet Options** page defines how to generate booklets in the output. It is used in conjunction with Imposition settings, which will appear after the Booklet entries have been made.

This page includes a handy illustration that displays how the final binding would look, based upon the current selections.

Options:

- **Configuration**: Use the drop-down to select the type of binding to use:
  - **Saddle Binding**: This binding places all the pages in a stack, binds the middle and folds the stack as one.
  - **Perfect Binding**: This binding type is often used for books. Pages are folded in the middle and then set side by side. The pages are then bound along the folded "spine".
  - **1 up Perfect Binding**: This binding does not contain any folding. The pages are lined up side by side and bound along one edge.
- **Booklet Binding Edge**: Use the drop-down to select the side on which to bind the booklet.

Optional **Cover Page** selections are available to Saddle Binding only.

- **Cover Page** checkbox: Check to enable cover pages to be created with the options below:
  - **Media** selections:
    - **Cover Media Size**: Use the drop-down to select the media size for the cover page, or use a Custom size and select **Width** and **Height** values.
  - **Front Cover** selections:

- **Blank**: Select to add no data to the front cover.
- **First page on outside and second page on inside**: Select to use the first 2 pages as the inside and outside of the front cover.

- **Back Cover** selections:
  - **Blank**: Select to add no data to the back cover.
  - **Last two pages on inside and outside**: Select to use the final 2 pages as the inside and outside of the back cover.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Imposition options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

Imposition refers to the printing of multiple pages on a single sheet. This is also known as N-Up printing.

The options on the **Imposition Options** page allow for the setting of imposition repetition, order, margins and markings.

As imposition selections are very specific and can be quite confusing, we have provided a handy dynamic diagram that displays a representation of the current imposition selections in real time. Whenever selections are changed, this display changes to reflect the selections

made.

You can even select how many pages are to be represented, whilst page numbers appear on the pages in the diagram and change dynamically, like everything else.

It is important to note that if Impositioning is used then the only meta data that will be available to the job thereafter will be the meta data at Job Segment level.

When using Impositioning everything within the Segment is Impositioned, so all meta data below the Job Segment level is lost. If *Booklet Binding* were selected, some of the imposition settings will be determined by the options made within the "Booklet Options" on page 1454 Page and those imposition settings will thus be disabled on this page.

Note: During Impositioning, all hyperlinks - both external and internal - are removed from the document (see "Hyperlink and mailto link" on page 850).

The Imposition selections that can be made are as follows:

- **Sample Page** group: Use to choose a paper size for the logical page(s) which would be printed upon the actual physical Imposition sheet. This helps in decision making regarding what paper sizes and orientations to use, with the Sample Page shown in blue on the white Imposition Sheet in the output preview (and validation) diagram.

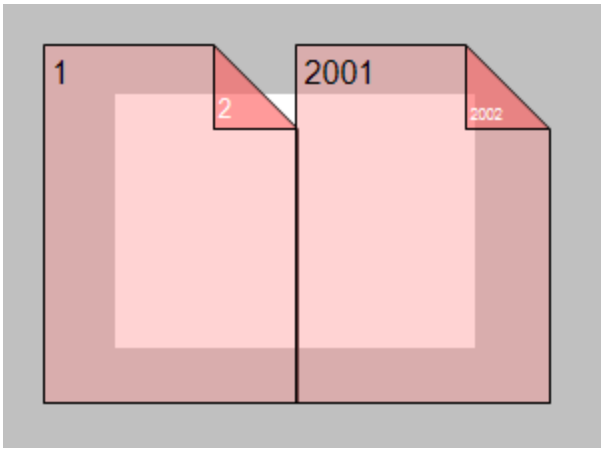
  Note: The logical page size can be set independently of any template open in the Designer, allowing for output presets to be created independently from the active template.

  If a template is currently loaded in Connect Designer, the size and portrait attribute of the first section of that template are used as defaults for the Sample Page.

  As a general rule, it makes little sense to modify the Sample Page size within the **Print Wizard** (as the template media size will already have been selected), but it does make sense when creating Print Output Presets for other templates.

  If no template is loaded or if the template does not have a Print section, then the Sample Page defaults to A4 portrait.

    - **Size**: Use the drop-down to select the media size of the logical template page.
    - **Orientation**: Select orientation (aspect ratio) of the logical template page (*Landscape* or *Portrait*).
    - The **Width** and **Height** values are real-only and appear for display and design purposes only.
- **Imposition Sheet** group: Use to select the physical media size that the logical pages will be printed upon. This appears as the white background in the output preview (and validation) diagram.
    - **Size**: Use the drop-down to select the size of the physical media the output is to be printed upon. The default is SRA3 (oversized A3).
    - **Orientation**: Select orientation (aspect ratio) of physical media sheet (*Landscape* or *Portrait*).
    - The **Width** and **Height** values are real-only and appear for display and design purposes only.
- Output diagram and validation. Use the settings here to control how the output diagram is to look and behave. It contains the following options:
    - **Display size** entry: Chose between a pre-set viewing percentage, or manually enter whatever percentage you would like, or select *Fit Page* or *Fit Width* to have the diagram display the page(s) within the diagram window dimensions.
    - Use the zoom in/out buttons / to increase or reduce the viewing percentage by 25% increments.
    - **Sample size** entry: Set the sample output size here. The output diagram will change dynamically to reflect the selection made here, if applicable.
      The maximum Sample size is 10,000 for booklet imposition and 10,000,000 for standard impositions.
    - Output diagram: A real-time display that reflects the selections made in the Imposition Options page.
      The blue page(s) are the individual logical page(s), upon a white background that represents the physical Imposition Sheet.
      If the settings are invalid, the display will appear as red, such as in the following diagram:



In this example the error is that the logical pages are too large to fit upon the Imposition Sheet that has been selected.

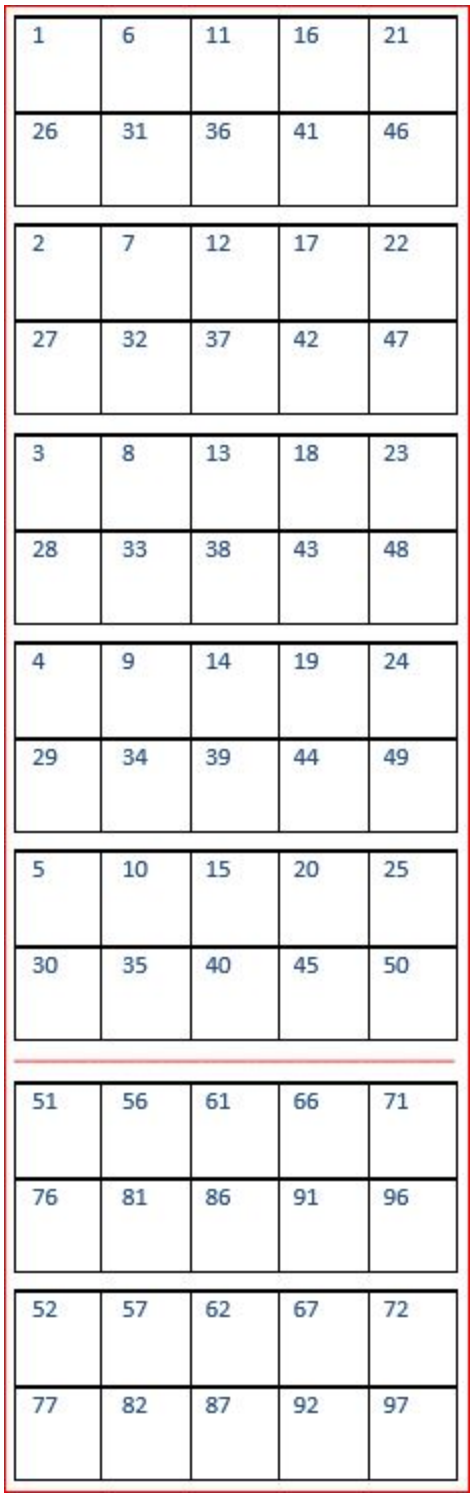
- **Position** group: These settings apply to the positioning of the logical page(s) within the Imposition Sheet.
    - **Method** selection: Set the position of the logical page(s) upon the Imposition Sheet. The choices are as follows:
        - *Center*: Have the logical page(s) centered within the Imposition Sheet. This is the default method.
        - *Scale to fit*: Scales the logical page(s) so that they fit the Imposition Sheet exactly.
        - *Offset*: Have the logical page(s) offset within the Imposition Sheet. The offset values can be set in the Left and Top entry boxes, which only become available for this positional method.
    - **Left** and **Top** values: These entry boxes are only enabled and their values applied when the Offset method has been chosen. They are used to set the offset of the logical page(s) upon the Imposition sheet.
    - **Rotate** selection: Select to rotate the logical page(s) within the Imposition Sheet.
- **Layout** group: Select how the logical page(s) are to be laid out on the Imposition Sheet.
    - **Horizontal** selection: Set how many logical pages are to be placed on one horizontal line within the Imposition Sheet.

      If the logic page was set to **Rotate** in the Position selections, then the Horizontal selection might appear as vertical, and the Vertical selection

might appear as horizontal, on the real time output diagram.

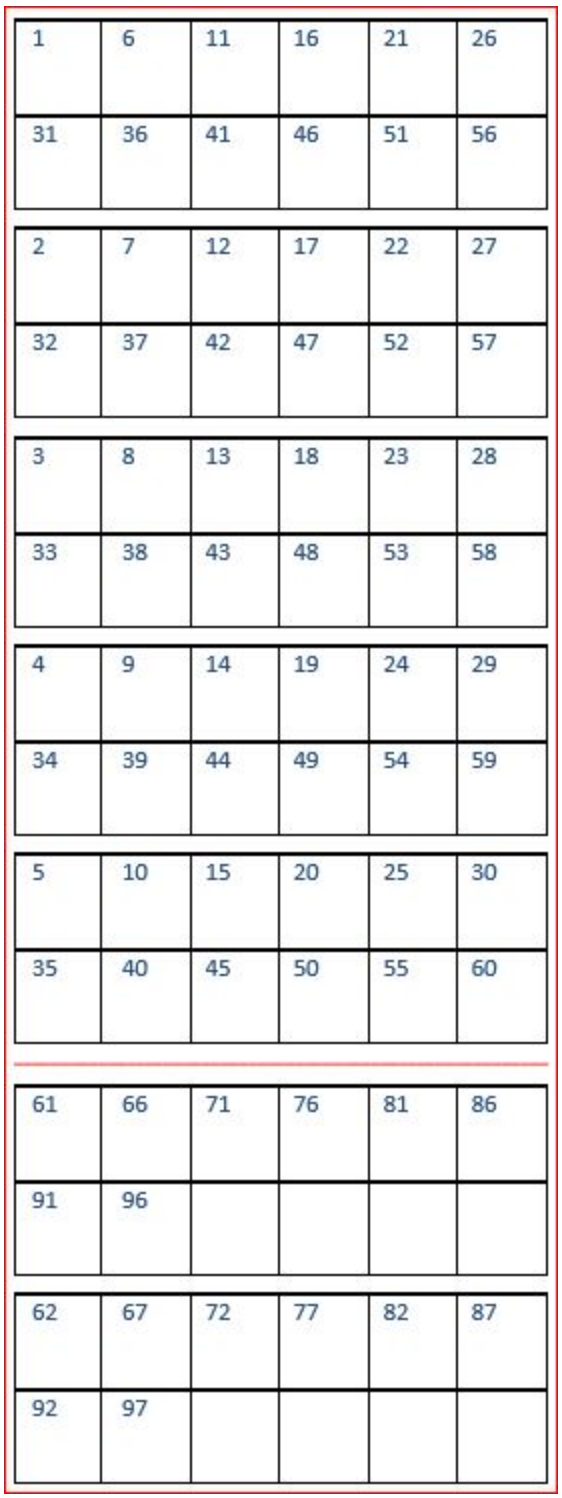
- **Vertical** selection: Set how many logical pages are to be placed on one vertical line within the Imposition Sheet.
- **Horizontal Gap** selection: Set the amount of blank space to add between each logical page on this axis.
- **Vertical Gap** selection: Set the amount of blank space to add between each logical page on this axis.
- **Stack Depth** selection: Stack Depth defines the number of Imposition Sheets that will be printed before drill sorting resets.
  Stack Depth works in conjunction with the Horizontal and Vertical page, Gap and Page Order selections in order to determine the printing order. This is best illustrated by way of example.

  The following "Cut and stack" example has a job with 100 single page documents and repetition settings of 5 Horizontal and 2 Vertical, plus a Stack Depth of 5. Each table represents a single Imposition Sheet, and the red line between the tables represents the Stack Depth:



Each set of 5 Imposition Sheets is a discrete drill sorted nUp block that can be guillotined and stacked. The first 5 sheets encompass records 1-50, whilst the next 5 sheets encompass records 51-100.

The following example is the same job but with a Horizontal repetition setting of 6:



As you can see, the outputs are *very* different.

> **Note:** If the Stack Depth is greater than the number of sheets in the entire job then you will get a single drill sorted block that can be guillotined and stacked to give the complete job in record order.

- **Sides** selection: Select how many sides of the Imposition Sheet are to be printed upon. Choice is between *Simplex* or *Duplex*.
- **Method** selection: In 2018.2 we introduced a new imposition page order for cut & stack impositioning that works with continuous feed devices: *Stack by Column*. This option is intended for impositions with more than 1 row. It places the next page in the next row of the same column on the current sheet, instead of in the same row and column of the next sheet like the pre-existing *Cut and Stack* does.
  The choices are between:
  - *Cut and Stack*: Any preset created in a version of Connect prior to Connect 2018.2 will default to this Layout Method. It allows total control of subsequent **Page Order** choices.
  - *Stack by Column*: For use with continuous feed printers. This selection limits subsequent **Page Order** choices to column based options only.
- **Page Order**: Select in which direction to go when adding sections to the output:
  - **Left to right, top to bottom**
  - **Right to left, top to bottom**
  - **Top to bottom, left to right**
  - **Top to bottom, right to left**
- **Reverse** selection: The reverse selections have been extended in Connect 2018.2, with the addition of the *Inverse page order* option. The options now available are:

- *Off*. Output is not reversed.
- *Stack upside down*: Essentially turns the stack upside down by reversing the output imposition page sides. The last sheet becomes the first, and in the case of duplex imposition, the back sides become the front sides.
  For presets created in a version prior to Connect 2018.2, a "Reverse Pages" selection will apply this *Stack upside down* setting.
- *Inverse page order*: This impositions the output in the ***exact opposite order*** from what has been selected.
  For example, if the selected order was **Top to bottom, left to right** (down the stack, left to right, top to bottom), then selecting **Inverse page order** would result in output that was up the stack, right to left, bottom to top.
  In the case of duplex impositioning, the front and back sides don't switch.

- **CropMarks** group: Select to add Crop Marks (also known as cut marks or trim marks) to the printed output.
  - **Type**: Use the drop-down to select whether to add Crop Marks to the page or not, and what type of Crop Marks to use. The Crop Mark types available are *Standard* or *Japanese*.
  - **Page side**: What side(s) of the page to put the Crop Marks. The choices are *Front*, *Back* or *Both*.
  - **Width**: Select the width of the crop mark lines.
  - **Offset**: How much separation (if any) to leave between the vertical and horizontal corner markings.
  - **Length**: Select the Length of the crop mark lines.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.

- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Inserter options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Inserter Options** page allows the selection of a High Capacity Feeder (HCF) model. These machines are also commonly referred to as Inserters or Folder-Inserters.

The options available on this page are dependent upon the model selected.

The options selected on this screen influence the position of the markings set on the next page: "Mark Position Options" on page 1466.

- **Model**: Use the drop-down to select from any previously loaded Inserter model, or use the Browse button to select a HCF file to load a new Inserter model.
  An image representing the chosen folder-inserter is displayed under the list, along with the HCF file details.
- **Options Group:**
  The options available here are all Inserter dependent, and thus will change based upon the Inserter model selection.
  To see how the selected Inserter markings would look on the printed page, click the Next button to move to the "Mark Position Options" on page 1466 page, which has a preview of the page. You can move back and forward between these two pages until you are entirely satisfied with the selections made.
  - **Mark Configuration**: Use the drop-down to select the type of markings to add. This selection basically equates to the amount of area the markings will take up on the printed page.
  - **Fold Type**: Use the drop-down to select the type of fold to apply to the paper. This will impact upon where on the page the markings will be placed.

- **Collation level**: Select whether the markings will be made at Document level, or Document Set level.
- **Print marks on back**: Check to place the Inserter Marks on the rear of the page.
- **Selective Inserts**: If selective inserts are supported by the chosen Mark Configuration you can select what markings to include and whether those markings are to included based upon some conditional setting. This can be done by highlighting the Mark Name entry and either pressing the **Edit** button, or using the right mouse click context menu, and selecting **Edit**.
  For information on how to edit the Selective Inserts settings, see "Selective Insert Dialog" on page 1467

- **Clear Background Area** Tab:
  Check the **Clear Background Area** checkbox to add a white background to the OMR, preventing background colors or elements interfering with the OMR Markings when they are read by the Inserter.
  - **Margins**:
    - **Same for all sides**: Check so that the Left margin selection is used to set all sides identically.
    - **Left, top, right, bottom**: Enter measurements for the margins on each side of the OMR Marks.
- **Custom OMR mark sizing** Tab:
  If supported by the currently chosen Mark Configuration you can select a Custom OMR size by checking the **Custom OMR mark sizing** checkbox.
  Select from any of the following, or leave the entries blank to use default values:
  - **Line length**: Enter a value between 10.16mm and 20mm.
  - **Line thickness**: Enter a value between 0.254mm and 0.63mm.
  - **Gap distance**: Enter a millimeter value 2.91mm and 4.2mm.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on

page 1380 page) and add or remove printing options from the print run.

- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Mark Position Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Mark Position Options** page displays a Preview of the output and the possible locations to place the inserter marks. The initial settings are determined by the selections made within the "Inserter options" on page 1464 page.
You can move back and forward between these two pages to perfect the settings, or you could move the inserter mark box to the desired location on the preview.

Preview box:

- The *pink area* displays the areas of the page where inserter marks can be positioned.
- The *small checkered box* displays the current location of the inserter marks. This box is selectable and can be dragged to the desired location within the printable (pink) areas.
  If the box is placed outside the printable areas the page will display an error and prevent attempts at leaving the page.

Below the Preview box are buttons which allow control of the Preview box. The selections that can be made are:

- **First Page**: Click to jump to the first page.
- **Previous Page**: Click to move to the previous page.
- **Next Page**: Click to move to the next page.
- **Last Page**: Click to jump to the last page.

- **Show Page**: Use the up and down arrows or type a page number to display a specific page within the document.
- **Zoom in/out**: Click to zoom in or out by 25%
- **Zoom Level**: Use the drop-down to select a predefined level or enter a zooming percentage.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Selective Insert Dialog

The Selective Insert Dialog provides control over individual inserts. In this dialog the selected can be set to be either selected ("Yes") or not ("No"), or selected based upon some conditional criteria ("Conditional").

For example, you could add a marking to the third page of a document by making the selection Conditional and then setting the Condition entry to "`page.nr == 3`".

Both Metadata and Informational fields can be used in the Conditionals settings.

For more details on how to create a conditional, see the "How to set up conditions" on page 1453 page.

Separation options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Separation Options** page defines how to separate a job into discreet portions. In addition the printer can add banner pages, slip sheets, and/or jog the output.

Note: Jogging is not an option on Continuous Form printers.

- **Separate by Count** group:
  This group allows for the splitting of output based upon a specified number of *Sheets*, *Documents*, *Document Sets* or *Job Segments*.
  - **Split**: Use the drop-down to select whether or not to split the job based on number of *Sheets*, *Documents*, *Document Sets* or *Job Segments*.
    - **None**: Select this option to ignore count splitting.
    - **At Exactly**: Select to create a split based upon a predetermined number of *Sheets*, *Documents*, *Document Sets* or *Job Segments*.
  - **Every**: Enter the number of *Sheets*, *Documents*, *Document Sets* or *Job Segments* at which the output is to be split.
    This option is only available if the **Split** entry has been set to *"At Exactly"*.
  - Use the drop-down box to chose what the separation option is to be.
    The selection is either **Sheets**, **Documents**, **Document Sets** or **Job Segments**.
    The default separation option is *Sheets*, which was the only option available for splitting prior to Connect 2019.1.

  Note: If the "**Separation by Count**" option is enabled, you can use the `segment.metadata` variable to set a dynamic file name. See the Segment section in the "Print output variables" on page 1659 for more information on setting this variable.

- **Separation Settings** group:
  This setting is only available if no **Separate by Count** split has been specified.
  - **Separation**: Use the drop-down to select when a job separation occurs, which is either **None** (no separation) or at the **Job**, **Job Segment**, **Document** or **Document Set** level.
- **Banners** (banner page) group:
  This setting adds banner pages to print jobs, at the selected level.

- **Every**: Use the drop-down to select where the banner page is to be inserted. Choices are is either **None** (no banner page, which is the default) or at the **Job**, **Job Segment**, **Document** or **Document Set** level.

Note: To add content to the banner page, the content can be made conditional on `sheet.banner`, or `page.banner` in the "Additional Text Settings" on page 1411.

Note: If the sheet following the banner is simplex, the banner will be simplex, but it the sheet following the banner is duplex, the banner will default to duplex. This ensures that a duplex print job doesn't accidentally become mixed-plex, which is not supported by all printers.

Tip: When banners are enabled, the overview of output preset properties shows a property "Banners", with a value that reflects the setting.

- **Slip Sheets** group:
  - **Add slip sheet**: Use the drop-down to select whether to add a slip sheet before or after a specific separation, or whether to use none.
  - **Every**: Use the drop-down to select at which separation to add a slip sheet, at the **Job**, **Job Segment**, **Document** or **Document Set** level.
  - **Media Size**: Use the drop-down to select the media size of the slip sheet. If a custom Media Size was chosen:
    - **Width**: enter slip sheet page width.
    - **Height**: enter slip sheet page height.

Note: When both banners and slip sheets are enabled for the same level, the slip sheet will go in front of the banner.

- **Jog** group:

Note: Jogging is not an option on Continuous Form printers.

  - **Jog after every**: Use the drop-down to select when to jog the printer, which is either **None** (no forced jogging) or at the **Job**, **Job Segment**, **Document** or **Document Set** level.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Additional Content

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Additional Content** page allows you to add content at the time of printing. There are four different types of Additional Content that can be added at print time: **Text**; **Images**; **Barcodes** and **OMR Marks**.
They can be used to add either static content or variable content.

The Additional Content option is particularly useful when you might need to drive custom processes on production machines using either Barcodes or OMR Marks, or if you need to add some last minute additions to the print job via text and/or images.
Additional content can also be used on banner pages.

Additional Text and Barcodes can contain `template` and `file` variables (see "File name variables" on page 1660), making it possible, for example, to display the output file name on banner pages.

Tip: Having a barcode added at the time of printing is also a time saver in an automated process where only a barcode needs to be added to an existing PDF and a template is actually unnecessary. In this case, content creation can be skipped. (See "Print processes with OL Connect tasks" on page 306.)

Page breakdown

The Additional Content table displays any Additional Content that has been set. Additional Content can be added, edited, removed, duplicated or moved within the table via the buttons to the right of the table.

The **Additional Content options** are:

- **Add**: Click to open some Additional Content.
  The choices are as follows:
  - "Additional Text Settings" on page 1411 entry.
  - "Additional Barcode Options" on page 1415 entry, from the selection presented.
  - "Additional Image Settings" on page 1413 entry.
  - "Additional OMR Mark Settings" on page 1449 entry.
- **Edit**: Click to edit the currently selected entry. This will launch the appropriate edit box, for the selected entry type.
  Double clicking on an entry in the table has the same effect as this button.
- **Delete**: Click to delete the currently selected entry or entries.
- **Duplicate**: Click to create a duplicated copy of the entry. This creates a new entry that is exactly the same as the original.
- / **Move Up / Move Down**: Click to move the selected entry (or entries) up or down within the table.

Additional Content Table: The table is split up into the following columns

- **Include in Output**: Indicates whether the entry is to be included in the output or not. The settings made in the Additional Content dialog box can be over-ridden here. This allows you to create a virtual "library" of Additional Content options in an Output Preset, from which you can then pick and choose from at time of printing.
- **Type**: Displays an icon showing what type the Additional Content entry is.
- **Description**: Displays the Description" that was entered in the Additional Content dialog.

  Note: The entered text Description might stretch over a couple of lines, so it is possible that not all of the Description will be displayed here.
- **Left**: Displays the Additional Content Left Positional value.
- **Bottom**: Displays the Additional Content Bottom Positional value.

- **Content**: Displays the actual Additional Content that is to be inserted.
  For *Text*, this would be the actual text (or data fields) that are to be inserted.
  For *Image*, this would be the name of the image file.
  For *Barcode*, this would be the data fields used for creating the barcode.
  For *OMR*, this would be the number of marks set for this entry.
- **Condition**: Displays the condition that is used to determine whether this Additional Content instance is to be added to the output or not (see "How to set up conditions" on page 1453). Document information variables that you can use in conditions are listed in the topic: Variables Available in Output.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings.
  This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Additional Text Settings

The **Additional Text Settings** dialog displays the property of Text added in the "Additional Content" on page 1409 page.

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have

included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Text added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: Option relates to Imposition (also known as N-Up) printing. Select this box to have the Text printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Text, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Text, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.

- **Font** group:
  - **Font Name**: Use the drop-down to select which font type to apply to the Text. The drop-down displays all the fonts installed on the system.

> **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

  - **Style**: Choose between "Regular", "*Italic*", "**Bold**" or "***Bold Italic***".
  - **Font Size**: Enter the font size in points (pt).
  - **Color**: Select what color the Text will be.
- **Text**: Enter the actual Text to appear on the page in the selected location. The Text can be spread over multiple lines, but no additional formatting can be added within this edit box. The entire Text will be printed using the formatting options selected in the **Font group**.
  Use the **Add** button ( ) to display a list of variable data that can be added to the Text. This includes metadata fields added in the Metadata Options, as well as some document information fields (see "Print output variables" on page 1659 for information on those).
- **Condition**: Enter the condition which determines whether or not the Text will be added to the document at print time.
  Use the **Add** button ( ) for selection options.
  For details on how to create a conditional, see the "How to set up conditions" on page 1453 page.

  > **Note:** To add the content to the banner page, type the condition sheet.banner, or page.banner.

Additional Image Settings

The **Additional Image** dialog displays the properties of the image added in the "Additional Content" on page 1409 page.

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the image.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Layer**: Whether this image will appear behind the text (the text will print over the image) or in front of the text (the text behind will be blanked out by the image, as transparent images are not supported).
    - **Ouput once per sheet**: Option relates to Imposition (also known as N-Up) printing. Select this box to have the Image printed once per sheet rather than once per document page.

      Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the image, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the image, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Filename**: Use the browse button to launch a Browse box to select an image file. This is a mandatory field.

  Note: Transparent images are not supported.

- **Preview** group: This displays a preview of the selected Image.

- **Scaling** group:
  Scaling the image expands the image but keeps the aspect ratio. The amount of scale and specific limitations can be applied used a combination of the following options:
  - **Max Width**: Enter the absolute maximum width the image can be scaled to, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **Max Height**: Enter the absolute maximum height the image can be scaled to, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **Scale**: What scale to apply to the image. The maximum scale is 10.0 to 1. Decimal values are allowed for this field.
- **Condition**: Enter the condition which determines whether or not the image will be added to the document at print time. Use the Add button for selection options.

  For details on how to create a conditional, see "How to set up conditions" on page 1453 page.

Additional Barcode Options

When adding Barcodes in the "Additional Content" on page 1409 page you can select from a series of predetermined Barcode types.
To create dynamic barcodes, "Meta Data options" on page 1374 must first have been set. Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

The options for each of these types are described on the following pages:

- "Australia Post Settings" on page 1302
- "Codabar Settings" on page 1417
- "Settings" on page 1422
- "Code 128 Settings" on page 1419
- "Data Matrix settings" on page 1425
- "EAN-8 Settings" on page 1433
- "EAN-13 Settings" on page 1431
- "GS1-128 Settings" on page 1428
- "Interleaved 2 of 5 Settings" on page 1435
- "Japan Post Settings" on page 1324
- "KIX Code (Dutch Post) Settings" on page 1326

- "PDF417 Settings" on page 1438
- "QR Code Settings" on page 1440
- "Royal Mail 2D Mailmark Settings" on page 1335
- "Royal Mail 4 State (CBC) Settings" on page 1337
- "Royal Mail 4 State Mailmark C Settings" on page 1339
- "Royal Mail 4 State Mailmark L Settings" on page 1342
- "UPC-A Settings" on page 1444
- "UPC-E Settings" on page 1446
- "US Postal Service IMb Settings" on page 1349
- "US Postal Service IMpb SettingsUS Postal Service Intelligent Mail Package Barcodes (IMpb) are used by the United States postal service for parcel deliveries. They contain parcel destination information along with other information about the delivery.To create dynamic barcodes, Metadata options must first have been set (see Metadata Options). Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.Application Identifiers in the barcode value need to be surrounded by round brackets, otherwise the barcode will not be generated correctly. For example, this value contains both a Postal Code Application Identifier (420) and a Channel Application Identifier 93: 420123459302234567890123456788. It should be written as follows: (420)12345 (93)02234567890123456788.More information about Application Identifiers can be found in the barcode specification here: https://postalpro.usps.com/shipping/impb/BarcodePackageIMSpec (8.0 Appendix D).Use the following options to configure the output Barcode settings:General group:Include in output checkbox: This determines whether of not this Additional Content entry should be included in the output or not. It is checked by default for all new entries.This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.Description edit box: An edit box for adding an optional Description to the Additional Content entry.Position group:Orientation: Use the drop-down to select the orientation of the Barcode added to the page.The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document: PortraitLandscapeInverse LandscapeInverse PortraitOutput once per sheet: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.If Imposition options such as auto-positioning and scaling were

selected, these options won't apply to the Additional Content added to the physical N-Up sheet.Left: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.Bottom: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.USPS IMPB Properties group:Barcode height: The height of the bars in the barcode, in inches (in). Value must be between 0.075 and 1.0 inches.Module width: The thickness of thin bars in the barcode, in inches (in). Thicker bars are a multiple of this value. Value must be between 0.013 and 0.021 inches.Text to barcode clearance: The space between the barcode bars and the top and bottom text, in inches (in). Value must be between 0.125 and 0.25 inches.Text height: The height of the text, in inches (in). Value must be between 0.09 and 0.125 inches.Line to text clearance: The space between the guard lines and text, in inches (in). Value must be between 0.031 and 0.04 inches.Guard bar height: The thickness of the guard lines, in inches (in). Value must be between 0.031 and 0.063 inches.Process Tilde:This selection is not applicable to this barcode type.Text: Enter the text used to generate the Barcode. Add button: Click to display a list of variable data that could be used for generating the Barcode. This includes metadata fields which must previously have been added in the Metadata options, likely at the Document Tags level (see Metadata Options), as well as some information fields.Condition: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see Conditionals. Add button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition." on page 1

Australia Post Settings

Australia Post 4 State barcodes are specific to Australian postal addresses. Every address in Australia has a numeric Delivery Point IDs (DPID) associated with it. Addresses must first be matched against an official Postal Address File (PAF), to obtain the unique DPID, and the DPIDs must be available to the job as "Meta Data options" on page 1374.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets,

libraries from which you can pick and choose what entries you wish to have included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Australia Post 4 State Properties** group:
  - **Track height**: The height of the small vertical centre bars, in metric (mm/cm). Value must be between 1 mm and 1.6 mm.
  - **Full bar height**: Height of the tallest vertical bars, in metric (mm/cm). Value must be between 4.2 mm and 5.8 mm.

The ascender and descender bar heights are automatically calculated based upon the *Full* and *Track* bar heights.

- **Module width**: Thickness of each bar in the Barcode, in metric (mm/cm). Value must be between 0.4 mm and 0.6 mm.
- **Module spacing**: The spacing between each bar, in metric (mm/cm). Value must be between 0.4 mm and 0.7 mm

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Codabar Settings

Codabar barcodes support the following data: 0-9 - $: / . + plus the optional specification of start/stop characters.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.

This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Codabar Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.

- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
- **Bar width ratio**: Set the Barcode bar width.
- **Default start symbol**: Use the drop-down to select the optional Barcode start character, which defines the encoding mode.
- **Default stop symbol**: Use the drop-down to select the Barcode stop character, which defines the encoding mode.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.
    - **Font size**: Enter a font size for the human readable text.
    - **Display start/stop symbols** checkbox: Adds the stop/start symbols to the Barcode text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
    - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
      This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
    - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Code 39 Settings

Code 39 is a discrete, self-checking barcode that is also known as "Alpha39", "Code 3 of 9" (often abbreviated to "3 of 9"), "Code 3/9", "Type 39", "USS Code 39" and "USD-3".

Code 39 data should contain no more than 20 digits from within the following range: Numeric digits: (0-9), upper-case letters (A-Z), seven special characters (- . space $ / + %) and the start/stop asterisk (*) character.

If the Extended character set is chosen, then lower-case letters (a-z) and other special ASCII characters can also be included.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Settings

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait **Landscape** **Inverse Landscape** **Inverse Portrait**

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Code 39 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Use extended character set**: Check to use the Code 39 Extended character set. This extends the range of supported data to include the full ASCII character set. This adds support for lower case letters (a-z) and the full range of ASCII punctuation and special characters.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger. The smallest Module Width is 0.19mm (high density).
  - **Bar width ratio**: Set the Barcode bar width.
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:

- **Ignore**: Ignore checksum calculations.
- **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
- **Check**: Verify the Barcode has a valid checksum.
- **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  > **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Code 128 Settings

Code 128 is a high-density barcode, used for alphanumeric or numeric-only barcodes. It supports all 128 ASCII characters.

> **Note:** To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
> Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Code 128 Properties** group:
- **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Print human readable text:** Check to add a textual version of the Barcode data.
  - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
  - **Font name**: Use the drop-down to select the font with which to display the human readable text.

    > **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

  - **Font size**: Enter a font size for the human readable text.
- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Data Matrix settings

A Data Matrix barcode is a high-density, two-dimensional (2D) matrix barcode which supports encoded text, numbers, files and digital data.

Note: In order to create a **GS1** compliant Data Matrix barcode, the barcode data must meet the following requirements:

- The barcode data must start with a leading FNC1 character. Either ~1 or ~d232.
- The GS1 Application Identifiers (AI) must be used for all data.
  The function code ~1 must be used as field separator for variable length AI elements.
- Only ASCII characters should be used.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Data Matrix Properties** group:
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Encoding**: The data represented in the symbol can be compressed using one of the following algorithms:

- **Auto**: Automatically detect the data content and encodes using the most appropriate method. This is the default option.
- **ASCII**: is used to encode data that mainly contains ASCII alphanumeric characters (ASCII 0-127). Use where Barcode size is a concern and where the data is alphanumeric.
- **Base 256**: used to encode 8-bit values.
- **C40**: used for data that mainly consists UPPER-case alphabetic letters and numbers.
- **Text**: used for data that mainly consists of lower-case alphabetic letters and numbers.
- **None**: Does not use any encoding.

Note: For general information on encoding, see https://en.wikipedia.org/wiki/Data_Matrix#Encoding.

Please check with the manufacture of the barcode reader if any specific encoding is required

- **Format**: select the Barcode size format from the drop-down list .

- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

  Note: The following tilde codes are supported:

  - ~1 = FNC1 character (for GS1 DataMatrix barcodes)
  - ~2 = Structure Append
  - ~3 = Reader programming
  - ~5 = Macro5
  - ~6 = Macro6
  - ~7 = ECI expressions

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

EAN-8 Settings

An EAN-8 barcodes are composed entirely of numerical data. It is comprised of 7 data digits containing the country/economic area code and an item reference code, with 1 following checksum digit.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait **Landscape** **Inverse Landscape** **Inverse Portrait**

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **EAN-8 Properties**:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger. The EAN-8 barcode employs a module width between 0.27mm and 0.66mm.
  -
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
    - **Ignore**: Ignore checksum calculations.
    - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
    - **Check**: Verify the Barcode has a valid checksum.
    - **Add**: Calculate and add a checksum character to Barcode, regardless of current value.

- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

EAN-13 Settings

EAN-13 barcodes are composed entirely of numerical data. The first 12 digits representing country/economic area, manufacturer and product codes + 1 following checksum digit.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **EAN 13 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger. The EAN-13 barcode employs a module width between 0.27mm and 0.66mm.
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
    - **Ignore**: Ignore checksum calculations.
    - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
    - **Check**: Verify the Barcode has a valid checksum.
    - **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.
    - **Font size**: Enter a font size for the human readable text.
- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.

- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
    - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

GS1-128 Settings

GS1-128 is also known as "EAN 128", "EAN/UCC 128" and "UCC 128". This barcode type not only encodes data, but also provides a mechanism for defining the meaning (or format) of that data. It supports alphanumeric data and some predefined Function Codes. See the Wikipedia GS1-128 entry for more information.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
    - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
      It is checked by default for all new entries.
      This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
    - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
    - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

      The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait	Landscape	Inverse Landscape	Inverse Portrait

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **GS1-128 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Check Digit marker**: This character is used as a placeholder for the check digit, which we be calculated at runtime. The character must be expressed in Hex.
  - **Group separator**: This character is used to define group separation points. The character must be expressed in Hex.
  - **Template**: Specify an optional Barcode "template".
    Examples:
    - `n13` defines a numeric field with exactly 13 digits.
    - `n13+cd` defines a numeric field with exactly 13 digits plus a check digit.
    - `an1-9` defines an alpha-numeric field with 1 to 9 characters.

    Elements can be combined using the '+' symbol.

- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Interleaved 2 of 5 Settings

Interleaved 2 of 5 barcodes are also known as "ITF" and "2/5 Interleaved". It is a numeric only barcode whose data must contain an even number of digits, as the barcode uses sequences of two digits interleaved with each other to create a single symbol. If the numeric data contains an odd number of digits, then a leading zero must be added to the beginning of the data.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Interleaved 2 of 5 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Bar width ratio**: Set the Barcode bar width.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.
    - **Font size**: Enter a font size for the human readable text.
- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Japan Post Settings

Japan Post 4 State Barcodes (also known as Kasutama barcodes, or Japanese Postal Barcode) are used by Japan Post for automated mail sorting. These barcodes contain the

address of the receiver encoded as a seven-digit postal code plus optional address data of up to 13 alphanumeric characters.

> **Note:** To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
> Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

> **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Japan Post Properties** group:
  - **Track height**: The height of the small vertical centre bars, in metric (mm/cm). Value must be between 0.6 mm and 1.8 mm.
  - **Full bar height**: Height of the tallest vertical bars, in metric (mm/cm). Value must be between 1.8 mm and 5.4 mm.
  - **Module width**: Thickness of each bar in the Barcode, in metric (mm/cm). Value must be between 0.3 mm and 0.9 mm.
  - **Module spacing**: The spacing between each bar, in metric (mm/cm). Value must be between 0.3 mm and 0.9 mm

- **Process Tilde**:
  This selection is not applicable to this barcode type.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.

- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

KIX Code (Dutch Post) Settings

KIX code barcodes (also known as Klantenindex, or Dutch KIX 4-State), are used by Royal Dutch TPG Post (Netherlands) for Postal code and automatic mail sorting. They contain the

address of the receiver, encoded in alphanumeric format (0-9, A-Z).

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Tip: A KIX code can also be part of an address block created by the Address Block wizard. See "Address Block" on page 955.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **KIX Code Properties** group:
  - **Track height**: The height of the small vertical centre bars, in metric (mm/cm). Value must be between 1 mm and 1.6 mm.
  - **Full bar height**: Height of the tallest vertical bars, in metric (mm/cm). Value must be between 4.2 mm and 5.8 mm.

    The ascender and descender bar heights are automatically calculated based upon the *Full* and *Track* bar heights.
  - **Module width**: Thickness of each bar in the Barcode, in metric (mm/cm). Value must be between 0.4 mm and 0.6 mm.
  - **Module spacing**: The spacing between each bar, in metric (mm/cm). Value must be between 0.4 mm and 0.7 mm

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.

- **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

PDF417 Settings

PDF417 is a two-dimensional, multi-row Barcode. It is used for encoding large amounts of data, with hundreds or even thousands of characters. It encodes alphabetic text, numbers, binary files and actual data bytes.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait **Landscape** **Inverse Landscape** **Inverse Portrait**

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **PDF417 Properties** group:
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Row height**: Defines the height of the bars for a single row, measured in pixels, points or metric.
  - **Width to height ratio**: Select the ratio of column width to row height.
  - **Mode**: Use the drop-down to set the compaction mode.
    - **Binary**: allows any byte value to be encoded.
    - **Text**: allows all printable ASCII characters to be end coded (ASCII values 32 to 126 and some additional control characters).
    - **Numeric**: more efficient mode for encoding numeric data.
    - **Auto**: Automatically detect the data content and encodes using the most appropriate method. This is the default option.

- **Error Correction Level**: Enter the error correction level for the built-in error correction method based on Reed-Solomon algorithms. The error correction level is adjustable between level 0 (just error detection, without correction) and level 8 (maximum error correction). Recommended error correction levels are between level 2 and 5, but the optimal value depends on the amount of data, printing quality of the PDF417 symbol and decoding capabilities.
- **Rows**: A PDF417 bar code can have anywhere from 3 to 90 rows.
- **Columns**: The number of data columns can vary from 1 to 30.

- ## Tilde processing

  Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.) The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
  Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

QR Code Settings

QR Code (Quick Response Code) is a 2D Barcode format that supports alphanumeric, numeric, byte/binary, and Kanji (Japanese-Chinese character) data.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).

Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

QRCode Properties group:

- **Size by**: Select size from the two options available:
  - **By area**: Connect will try to size the Barcode to fit the specified area by dynamically changing the module width to the **Size** selection. The lower module width limit is governed by the **Minimum module width** selection.
    Enter the sizes in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **By module width**: Connect will try to size the Barcode to the module width of the characters. Large Barcode values will result in larger Barcode and vice versa.
    Enter the **Module width**in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Encoding**: Define the encoding of the Barcode:
  - **Auto**: Automatically detect the data content and encodes using the most appropriate method. This is the default option.
  - **Numeric**: 7089 numerical characters.
  - **Alphanumeric**: 4296 alphanumerical characters.
  - **Byte**: 2953 characters.
  - **Kanji**: 1817 Japanese/Chinese characters.
- **Version**: Select the preferred QR code version (which sets the data length field) from the 40 available.

Note: The Encoding and Version fields work together to determine how many characters are encoded within a *length field*. The following table shows the number of bits in a length field, based upon the selections made:

Encoding	Ver. 1-9	Ver. 10-23	Ver. 27-40
Numeric	10	12	14
Alphanumeric	9	11	13
Byte	8	16	16

Kanji	8	10	12

- **Error Correction Level**: Part of the robustness of QR codes is their ability to sustain “damage” and continue to function even when a part of the QR code image is obscured, defaced or removed. A higher correction level duplicates data within the QR Code to allow for damaged areas. The higher the Error Correction Level, the larger the Barcode will be. The choices are (in order from lowest to highest): **L**ow, **M**edium, **Q**uartile and **H**igh.
- **Use ECI for encoding messages as bytes**: Selecting Extended Channel Interpretations (ECI) allows encoding multiple character sets (e.g. Arabic, Cyrillic, Greek, Hebrew) and other data interpretations, into one QR Code symbol.
- **Multi-part QR Code (structured append)**: Select to append a QR Code symbol in a structured format.
    - **Part**: indicates the position of the QR Code symbol within the group of Structured Append symbols.
    - **of**: indicates how many Structured Append symbols exist.

    Note: The Structured Append symbols Part number can never exceed the sum total of Structured Append symbols available (the "of" value). Thus selecting a Part number beyond the existing sum total will increase the sum total to the same value.
- **Use FNC1**: Check to enable Application Identifiers. These are often used to encode links to websites, or to encode production/batch details.
    - **Position**: Select between the two methods for encoding FNC1 characters within QR Codes:
        - **First** Position - uses the GS1 QR Code standard.
        - **Second** Position - uses the AIM QR Code standard. If this option is chosen then the appropriate Application Indicator will also need to be set.
            - **Application ID**: Enter the appropriate QR-Code Application Indicator in accordance with the specific industry or application specifications (as provided by AIM International).
- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Royal Mail 2D Mailmark Settings

Royal Mail 2D Mailmark Barcodes are a variant of 2 dimensional Datamatrix barcodes. They are used by the British Royal Mail postal service. They contain the address of the receiver, encoded in alpha-numeric format (0-9, A-Z).

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:

- **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

  The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Royal Mail 2D Properties** group:
  - **Module Width**: The size of each column and row within the barcode. Value must be between 0.5 mm and 0.7 mm.
  - **Preferred Version**: Select the preferred size of the barcode in terms of rows by columns.

    > **Note:** Depending on the information being embedded, this entry may be overridden if the specified size is not big enough to contain the barcode information.

- ### Tilde processing

  Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.) The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
  Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Royal Mail 4 State (CBC) Settings

Royal Mail 4-State Customer Bar Code (CBC) is also known as the Royal Mail 4-State Customer Code (RM4SCC or RoyalMail4SCC). It is used by the British Royal Mail postal service. The barcode contains an address, encoded in alpha-numeric format (0-9, A-Z).

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Royal Mail Properties** group:
  - **Track height**: The height of the small vertical centre bars, in metric (mm/cm). Value must be between1.02 mm and 1.52 mm.
  - **Full bar height**: Height of the tallest vertical bars, in metric (mm/cm). Value must be between4.22 mm and 5.84 mm.
    The ascender and descender bar heights are automatically calculated based upon the *Full* and *Track* bar heights.

- **Module width**: Thickness of each bar in the Barcode, in metric (mm/cm). Value must be between0.38 mm and 0.63 mm.
- **Module spacing**: The spacing between each bar, in metric (mm/cm). Value must be between0.38 mm and 0.63 mm.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Royal Mail 4 State Mailmark C Settings

Royal Mail 4-State Mailmark C is used by the British Royal Mail postal service. The barcode contains an address, encoded in alpha-numeric format (0-9, A-Z).

The content (or message) of the barcode consists of the following 6 parts (or fields), which are detailed here:

Field	Length	Allowed Characters
Format	1	[01234]
Version ID	1	[1234]
Class	1	[0123456789ABCDE]
Supply Chain ID	2	[0123456789]
Item ID	8	[0123456789]

Destination Post Code plus DPS	9	`[ 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ]` The value "XY11 " is used for this field to specify an *International Designation*.

As an example, the message "`1129966448822CC00AA0A `" has *Format* = 1, *Version ID* = 1, *Class* = 2, *Supply Chain ID* = 99, *Item ID* = 66448822 and *Destination Post Code plus DPS* = "CC00AA0A ".

Note: The quotes are not part of the message, but are used here to indicate that the space is part of the message.

There will be a runtime error (i.e. an error occurring during output creation), if the supplied message does not adhere to the format specified above.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait **Landscape** **Inverse Landscape** **Inverse Portrait**

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Royal Mail 4-state Mailmark C Properties** group:
  - **Track height**: The height of the small vertical centre bars, in metric (mm/cm). Value must be between1.02 mm and 1.52 mm.
  - **Full bar height**: Height of the tallest vertical bars, in metric (mm/cm). Value must be between4.22 mm and 5.84 mm.

    The ascender and descender bar heights are automatically calculated based upon the *Full* and *Track* bar heights.
  - **Module width**: Thickness of each bar in the Barcode, in metric (mm/cm). Value must be between0.38 mm and 0.63 mm.
  - **Module spacing**: The spacing between each bar, in metric (mm/cm). Value must be between0.3 mm and 0.7 mm.
- **Process Tilde**:
  This selection is not applicable to this barcode type.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Royal Mail 4 State Mailmark L Settings

Royal Mail 4-State Mailmark L is used by the British Royal Mail postal service. The barcode contains an address, encoded in alpha-numeric format (0-9, A-Z).

The content (or message) of the barcode consists of the following 6 parts (or fields), which are detailed here:

Field	Length	Allowed Characters
Format	1	`[01234]`
Version ID	1	`[1234]`
Class	1	`[0123456789ABCDE]`
Supply Chain ID	6	`[0123456789]`
Item ID	8	`[0123456789]`
Destination Post Code plus DPS	9	`[ 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ]` The value "XY11 " is used for this field to specify an *International Designation*.

As an example, the message "`1129999999664488 22CC00AA0A `" has *Format* = 1, *Version ID* = 1, *Class* = 2, *Supply Chain ID* = 999999, *Item ID* = 66448822 and *Destination Post Code plus DPS* = "CC00AA0A ".
Note: the quotes are not part of the message, but are used here to indicate that the space is part of the message.

There will be a runtime error (i.e. an error occurring during output creation), if the supplied message does not adhere to the format specified above.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Royal Mail 4-state Mailmark L Properties** group:
  - **Track height**: The height of the small vertical centre bars, in metric (mm/cm). Value must be between1.02 mm and 1.52 mm.
  - **Full bar height**: Height of the tallest vertical bars, in metric (mm/cm). Value must be between4.22 mm and 5.84 mm.

    The ascender and descender bar heights are automatically calculated based upon the *Full* and *Track* bar heights.
  - **Module width**: Thickness of each bar in the Barcode, in metric (mm/cm). Value must be between0.38 mm and 0.63 mm.
  - **Module spacing**: The spacing between each bar, in metric (mm/cm). Value must be between0.3 mm and 0.7 mm.
- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

UPC-A Settings

The Universal Product Code (UPC-A) Barcode is widely used for tracking trade items in stores and at the point-of-sale. It consists of 12 numerical digits which are uniquely assigned to each trade item.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

> **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

UPC A Properties group:

- **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
    - **Ignore**: Ignore checksum calculations.
    - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
    - **Check**: Verify the Barcode has a valid checksum.
    - **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      > **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

    - **Font size**: Enter a font size for the human readable text.
- **Process Tilde**:
  This selection is not applicable to this barcode type.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

UPC-E Settings

The Universal Product Code (UPC-E) Barcode is widely used for tracking trade items in stores and at the point-of-sale. It consists of 6 numerical digits which are uniquely assigned to each trade item.

> **Note:** To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
> Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:

- **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

  The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

UPC A Properties group:

- **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
  - **Ignore**: Ignore checksum calculations.
  - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.

- **Check**: Verify the Barcode has a valid checksum.
- **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

US Postal Service IMb Settings

US Postal Service Intelligent Mail Barcodes (IMb) are used by the United States postal service. They contain the address of the receiver, encoded in numeric format (0-9).

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **USPS IMB Properties** group:
  - **Track height**: The height of the small vertical centre tracking bars, in inches (in). Value must be between 0.039 and 0.057 inches.
  - **Ascender height**: The height of the ascender bar.
    This also determines the height of the descender bar and full height bars, in inches (in). Value must be between 0.082 and 0.111 inches.
  - **Module width**: Thickness of each bar in the Barcode, in inches (in). Value must be between 0.015 and 0.025 inches.
  - **Interchar gap size**: The gap between the bars, calculated as a ratio of the module width, where a value of 1.0 equates to a gap width the same as the module width.
    Value must be between 0.012 and 0.04 inches.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

    - **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.

- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
    - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

US Postal Service IMpb Settings

US Postal Service Intelligent Mail Package Barcodes (IMpb) are used by the United States postal service for parcel deliveries. They contain parcel destination information along with other information about the delivery.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Note: Application Identifiers in the barcode value need to be surrounded by round brackets, otherwise the barcode will not be generated correctly.
For example, this value contains both a Postal Code Application Identifier (*420*) and a Channel Application Identifier *93*: *42012345930223456789012345678*8. It should be written as follows: ***(420)****12345****(93)****0223456789012345678*8.
More information about Application Identifiers can be found in the barcode specification here: https://postalpro.usps.com/shipping/impb/BarcodePackageIMSpec (8.0 Appendix D).

Use the following options to configure the output Barcode settings:

- **General** group:
    - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
      It is checked by default for all new entries.
      This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
    - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:

- **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

  The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **USPS IMPB Properties** group:
  - **Barcode height**: The height of the bars in the barcode, in inches (in). Value must be between 0.075 and 1.0 inches.
  - **Module width**: The thickness of thin bars in the barcode, in inches (in). Thicker bars are a multiple of this value. Value must be between 0.013 and 0.021 inches.
  - **Text to barcode clearance**: The space between the barcode bars and the top and bottom text, in inches (in). Value must be between 0.125 and 0.25 inches.
  - **Text height**: The height of the text, in inches (in). Value must be between 0.09 and 0.125 inches.

- **Line to text clearance**: The space between the guard lines and text, in inches (in). Value must be between 0.031 and 0.04 inches.
- **Guard bar height**: The thickness of the guard lines, in inches (in). Value must be between 0.031 and 0.063 inches.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Additional OMR Mark Settings

The **Add OMR** dialog displays the properties of an OMR Mark that was added in the "Additional Content" on page 1409 page.

These OMR marks differ from High Capacity Feeder (HCF) generated inserter marks. Those marks are specific to the inserter machine they were created for, whereas these additional OMR marks are completely independent and customizable. These custom OMR marks can be used to cater for inserter machines not currently supported by a HCF, or they can be used for any non-inserter related post-processing driven by OMR marks.

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.

- **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.

- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the OMR Mark added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Page Side**: Select whether the OMR Mark will print on the front or back of page.
  - **Output once per sheet**: Option relates to Imposition (also known as N-Up) printing. Select this box to have the OMR Mark printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the OMR Mark, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **Bottom**: Enter the distance between the bottom margin of the page and the OMR Mark, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.

- **Options Tab**:
  - **Collation Level**: Choices are:
    - **Document**: Treats each document as a group and the group and match marks will be set based upon the start and end of a document.

    - **Document Set**: Treats each document set as a group and the group and match marks will be set based upon the start and end of a document set.
- **Draw Hot Spots**: This adds a red rectangle around the location of each individual mark in the output, allowing easier checking of the OMR mark logic.
- **Line Options** group:
    - **Line Thickness**: Sets the thickness of each OMR mark line.
    - **Line Length**: Sets the lenght of each OMR mark line.
    - **Line Spacing**: Determines how the spacing between each OMR mark line will be set. The associated control beneath the combination box will be enabled, based upon this selection.
        - **Line Per Inch**: If **Line Spacing** is set to *Lines Per Inch* this option will be enabled. It defines how many lines will print per inch.
        - **Gap Distance**: If **Line Spacing** is set to *Gap Distance* this option will be enabled. It defines the size of the gap between lines; i.e. the distance from the bottom of one OMR mark line to the top of the next.
        - **Line Distance**: If **Line Spacing** is set to *Line Distance* this option will be enabled. It defines the distance from the top of one line to the top of the next.
- **Sequence Number Range** group: Allows selection of Start and Stop points for the wrapping page sequence number in a group.
  For example, a range of 2-10 would cause the sequence numbers to iterate as follows: 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4 ...

  Note: The Sequence Number iterates per page within a group and is used to identify missing pages in a group.

    - **Start**: The starting point for the range
    - **Stop**: The end point of the range
    - **Start number**: The number to start from (from within the selected range).
- **Match Number Range** group: Allows selection of Start and Stop points for the wrapping match number for a group.
  For example, a range of 1-6, with a Start number of 2 would cause the matched numbers to be as follows: 2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 2, ...

> **Note:** The Match Number iterates per group and is used to identify missing groups

  - **Start**: Start number
  - **Stop**: Stop number
- **Condition**: Enter the condition which determines whether or not the OMR Mark will be added to the document at print time.
  For details on how to create a conditional, see the "How to set up conditions" on page 1453 page.

- **OMR Marks Tab**:
  - **#**: OMR Mark number (display only).
  - **Type**: Type of OMR Mark (display only).
  - **Value**: OMR Mark Value. These can be selected and altered for Sequence, Match and Parity marks, as described below.
  - **Add**: Add an OMR Mark entry to the table.
    Choices are between:
    - **On**: This represents a mark that is always printed.
    - **Off**: This represents a mark that is never printed; i.e. it pads the marks out with an empty position.
    - **Group Start**: This represents a mark that is printed on the first page of a group.
    - **Group End**: This represents a mark that is printed on the last page of a group.

      > **Note:** In a single page group both Group Start and End marks will print if defined since the page is both the start and end of the group.

    - **Sequence**: This represents a mark that is printed when the specified bit is set in the sequence number of the page.
      For example, if the bit for the mark is set to 2 and the sequence number for the page is 5 then it will not print since the value 5 consists of the bits 1 and 4.
      Use the drop down box to select the entry.

- **Match**: This represents a mark that is printed when the specified bit is set in the match number of the group.
  For example, if the bit for the mark is set to 2 and the match number for the group is 3 then it will print since 3 consists of the bits 1 and 2.
  Use the drop down box to select the entry.

  > **Note:** The match number is the same for all pages in a group.

- **Parity**: This mark prints in order to maintain the parity of the number of lines printed on the page. If set to *Even* then it will print if the total count of the other printed marks in the printed is odd.
  For example, by printing the parity mark it will create an even number of marks on the page. And vice versa with *Odd* parity - the parity mark will print if the total number of other printed marks on the page is even in order to keep the overall count odd.
  Use the drop down box to select the entry.

- **Conditional**: Enter the condition which determines whether or not this OMR Mark will be added to the document at print time.
  For details on how to create a conditional, see the "How to set up conditions" on page 1453 page

- **Delete**: Delete an entry from the table
- **Move up**: Move a entry up the table
- **Move down**: Move a entry down the table
- **Edit**: Edit a **Conditional** entry within the table.

  > **Tip:** You can also double click a **Conditional** entry within the table to edit it.

How to set up conditions

- **Condition**: Enter the condition which determines whether or not this element will be added to the document at print time.

  Use JavaScript conditional expressions to construct your condition.
  You may use JavaScript logical operators (`==`, `!=`, `<`, `>`, `&&`, `||`, etc.), literal values (`"text'`, `3.14`, etc.), mathematical operators (`/`, `*`, `+`, `-`, etc.) and data fields.

  - **Add**: Click to add a field to the to the conditional expression.
    Select from a list of metadata fields (as added in Metadata Options), document

information fields (see Variables Available in Output), or common expressions.

For example, selecting the **Expressions > First Page** would add `"page.nr == 1"` to the conditional entry.
Example: `if (page.nr == 1) { ... }`

Examples

Example: `"page.nr == 1"`

This condition means the additional content can only be added to the first page.

Example: `"page.banner"`

With this condition, the additional content will only be added to banner pages.

Example: page.banner && page.front

This condition determines that the page must be both a banner page and a front page. In case of duplex banner pages, this prevents content from appearing at the back of the banner page.

Example: `page.content`

This condition prevents additional content from appearing on banner pages. It is true for all pages that are part of the content, and false for banner pages.

Job Preset

A Job Preset (or Job Creation Preset) is a settings file by which the Connect Server can filter, sort, and group print content items, add meta data and store finishing settings.

The Presets are all stored as individual files, using the selected Preset name and a "OL-jobpreset" file extension.

Once saved, Print Presets can be loaded in the **Print Wizard** or be used in the "Send to Server", or "Package" options, to re-use the selected Output values.

You can also send the Preset to Workflow to use it in Workflow configurations. See "Sending files to Workflow" on page 593.

For more information about Job Presets, see "Print Presets" on page 1644.

Note: The **Job Creation** Wizard requires an active Template and associated data.

Dialog Interface

- **Name**: Enter the name for Job Preset. This will become the preset file name, when the preset is saved.
- **Description**: Optional description for the Job Preset.

  > **Tip:** It is recommended that you enter a meaningful description here, so what the Preset does and how it is to be used can be more easily understood in the future.

- **Options**: The following options can be used with Job Preset.
  - **Runtime Parameters**: select "Runtime Parameter Options" below to store information which can be used at runtime for comparisons against conditions within Job Creation, or for use with external sorting programs.
  - **Apply filtering and sorting to record selection**: Check to activate the "Data filtering options" on page 1363 (to filter out certain records) and "Sorting options" on page 1368 (to sort the remaining records) pages of the wizard.
  - **Use Grouping**: Check to configure grouping of output into jobs, job segments or document sets. See "Grouping options" on page 1372.
  - **Override Finishing Options**: Check to configure custom "Finishing options" on page 1366, such as binding.
  - **Include Metadata** : Check to add meta data to the output. This can be done at Job, Job Segment, Document, Document Set and Page level. See "Meta Data options" on page 1374.
- **Data Mapping Configuration**: Use the drop-down to select which data mapping configuration this Job Preset will be based on. The data mapping configuration's model is used for field names in sorting, etc.

Click Next in this dialog to continue entering Job Preset data. This takes you to the first page of the Wizard, which is dependent upon the option selections made.

Runtime Parameter Options

This page appears as part of the **Advanced Print Wizard** and the "Job Preset" on the previous page wizard.

The **Runtime Parameter Options** page allows you to add runtime parameters to the Job. These runtime parameters could be used in a number of ways. Such as in the following scenarios:

Filtering rules scenarios

A filter condition could be used to compare a date from the record against a date passed in as a runtime parameter.

For instance, the record could have a due date, with only documents due in two weeks time to be selected. Calculating the date "two weeks from today" could be done in Workflow, with this date then having to be passed as a parameter back to Connect when Job Creation is started.

Another example would be a list of document types for a job. The document could have a property with its document type (e.g., reminder, collection letter, etc.), and the runtime parameter could be a comma separated list of document types for the job. So a rule would then compare if document type occurs in the list of document types.

Finishing rules scenarios

Similar to filtering, a finishing condition could use runtime parameters to compare against.

For example, consider if a copy of a print job has to be produced, in which the copies have to be stapled, while the original print run are without staples. A runtime parameter could be used during job creation to indicate if the Job Creation is working on the original or the copy.

Without runtime parameters, the same functionality would require two job presets. So this use case becomes important for relatively complex presets where duplicating the preset creates a maintenance issue.

External sorting scenarios

External sorting benefits considerably from runtime parameters.

Paths for storing temporary files can be controlled from Workflow, allowing them to be unique per Job Creation. This facilitates running multiple instances of the sorting command at the same time. For instance, Workflow's temporary folder for a process could be supplied.

Especially in case of postal sorting, there can be a need for parameters that vary per job, such as intended delivery date, tracking id, and more.

Metadata properties

The value for a metadata property could be supplied through a runtime parameter. These could be, for example, the production date, or the name of the current operator.

Page breakdown

Runtime Parameters can be added, edited, removed, duplicated or moved within the Runtime Parameter table via the buttons to the right of the table.

The **Runtime Parameter options** are:

- **Add**: Click to open a new Runtime Parameter.
  This launches the Add/Edit Runtime Parameter dialog.
- **Edit**: Click to edit the currently selected entry. This launches the Add/Edit Runtime Parameter dialog.
  Double clicking on an entry in the table has the same effect as this button.
- **Delete**: Click to delete the currently selected entry or entries.
- **Duplicate**: Click to create a duplicated copy of the entry. This creates a new entry that is exactly the same as the original, with a numeric increment added to the end of the name.
- / **Move Up / Move Down**: Click to move the selected entry (or entries) up or down within the table.

Runtime Parameter Table: The table lists all the currently available runtime parameters. It is split up into the following columns

- **Parameter**: The name of the parameter.
- **Type**: The parameter type.
- **Default Value**: Displays the default parameter value.

Advanced Printing Wizard and Job Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Job Preset*** Wizard only): Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.

- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Data filtering options

This page appears as part of the **Advanced Print Wizard** and the "Job Preset" on page 1359 wizard.

The **Data Filtering Options** page is used to select the records to be printed. Conditions are evaluated on each record, and can be based upon such diverse data as the data fields used in records or the physical printing options for a record.

The rule(s) apply at a Group level. There may be one rule or many rules at the same level, and there may be groups within groups, providing the ability to create quite complex nested logical structures.

Group

A group consists of one or more rules with a logic operator. Four logic choices are available at the Group level. These are:

- **All of the following**.
  This equates to the logical operator (... AND ...).
  If *all* of the associated criteria are met, then this group resolves to TRUE.
- **Any of the following**.
  This equates to the logical operator (... OR ...).
  If *any* of the associated criteria are met, then this group resolves to TRUE.
- **Not all of the following**.
  This equates to the logical operator (NOT(... AND ...)).
  If *any (but not all)* of the associated criteria are met, then this group resolves to TRUE.
- **Not any of the following**.
  This equates to the logical operator (NOT (... OR ...)).
  If *none* of the associated criteria are met, then this group resolves to TRUE.

The top level is always a group. A group can contain one or more Rules and/or Groups.

Rule

A Rule is a logic expression using a single operator or function, and their associated operand(s). The number and type of operands is dependent upon the operator or function chosen.

Rules can also have different types. The available rule types depend upon the context where the Rule Editor is used. For instance, within Job Creation, there can be data rules that are based on record data belonging to a content item, but there will also be "location rules" that deal with the position of a document within the job.

The Rule types are as follows:

- **Data Rule**: This rule operates upon Data entries. Select the Data Field, then select the logical test and the associated test value(s). The options available will depend upon the Data Field type (they are "type aware"), but all have at least "*is equal to*" and "*is not equal to*" to another entry as an option. The "*is set*" and "*is not set*" selections allow to determine whether the data field has a value or not (i.e. do a null comparison).
  The data type specific options are as follows:
  - **Alphanumeric** data field specific selections: Whether the alphanumeric string "*contains*", "*does not contain*", "*starts with*" or "*ends with*" another string value. The "*is like*" and the "*is not like*" selections allow the use of "*" wildcards.
    The comparison string could be a manually entered string, or it could be another textual data field. The data field could be from the data model, or it could be a runtime parameter entered as part of Job Creation. Runtime parameters are expressed as "params.xxxx", where the xxxx is the name of the runtime parameter.

    In the Data Filtering Options rules editor the comparisons are always case sensitive. In other rule editors there is an option to turn off case sensitivity.
  - **Date** data field specific selections: Whether the date field is "*before*", "*after*", "*not before*" or "*not after*" another date entry, or "*is between*" two other date entries. The comparison dates can be a manually entered dates, or date data fields (either from the data model or runtime parameters).
    The date value should be in ISO 8601 format. When entering dates manually it is best to select the date using the date selection option.
  - **Boolean** data field specific selections. Check whether the selected Boolean data field "*is true*" or "*is false*", or whether it "*is equal to*", or "*is not equal to*" a Boolean runtime parameter.
  - **Numeric** data field specific selections: Whether the numeric fields is "*less than*", "*less than or equal to*", "*greater than*" or "*greater than or equal to*" another number entry. The comparison number could be a specified number, or it could

be another numeric data field (either from the data model or runtime parameters).

- **Media Rule**: This rule operates on either the Media *Name* or coating (either *Front Coating* or *Back Coating*) .
  For the *Name* value, the choice is a straight binary option based upon the selected Media Name.
  For the *Coating* values, the choice is about what type of coating applies (with an "Unspecified" option as a fall back).
- **Binding Rule**: This rule operates upon the binding that applies to the record.
  The options are based on the *Style*, *Side*, *Location* or *Angle* of the binding. The logical operators that apply are dependent upon the type of Binding the test is being applied to.
- **Duplex Rule**: This rule operates upon the record Duplex settings. This is a binary choice between Duplex or non-Duplex.
- **Size Rule**: This rule operates upon the size of the selected group. This could be the amount of sub-groups contained within, or the amount of physical *Sheets*, *Pages* or *Sections*. The logical operators that apply are dependent upon the group that the size test is being applied to.
- **Property Rule**: This rule compares a job *Property* value (such as "Control Scripts" on page 1098 set within Designer) against a selected string. The comparison operators that are available for this rule are the same as those for Alphanumeric **Data Rules**.
  The only difference is that the "*is set*" and "*is not set*" selections not only check for null property values but also for properties that don't exist at all.

Text box

This box displays a textual representation of the logical conditions set within the data filtering table.

Advanced Printing Wizard and Job Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.

- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Job Preset*** Wizard only): Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Finishing options

This page appears as part of the **Advanced Print Wizard** and the "Job Preset" on page 1359 wizard.

Use the **Finishing Options** page to force the use of specific printer finishing options, rather than using any finishing options that might have been set in the template's Print Context and Print Section options.

These settings are only applied when producing print output. They do not modify the original finishing options in either the Section or the Template.

Finishing settings can be set on various levels of Job Creation, such as Document Sets and Job Segments. This allows you, for example, to staple Document Sets whilst punching holes in Job Segments.
You can also have multiple finishing settings at the same level.

> **Note:** Whether certain features will work for your printer will be dependent upon the printer model selected (done in the "Print options" on page 1380 page). Finishing options do not really apply to continuous form printers.

- **Finishing Options Table**: This table, on the left of the Wizard page, allows you to add or remove sections to apply the Finishing options to.
  It contains the **Section** for which the settings are to be applied (*Section*, *Document*, *Document Set* or *Job Segment*), plus the **Finishing** options that are to be applied, and any **Rule** determining the conditions under which the Finishing settings are to be applied.
  To set the details, use the following options:
  - **Import Finishing:** Imports the existing settings from a Connect Template. The Template could be the current template (***Import from current template***), or some other Template (***Import from template ...***).

- **Add:** Adds a new section to apply Finishing options to. The default is to add a new *Section*, but these can be changed to *Document*, *Document Set* or *Job Segment* types, thereafter.
  You can have multiple selections of each type, all with different criteria determining when the Finishing options are to be applied.
- **Delete:** Removes the selected Section(s) from the table.
- **Move Up / Move Down:** Moves the selected Section(s) up or down within the table.
- **Edit ... :** Brings up the Rule Editor Dialog, which can be used to construct the rule(s) that determines whether this conditional Finishing option is to be applied or not.

- **Binding** group:
  - **Style**: What type of Binding to request on the printer. This includes *Stapled, Glued, Stitched, Ring, Comb, Coil*, amongst other options..
  - **Side**: Sets the side of the paper that the Binding is to occur. This includes both the top and the bottom of the paper,
  - **Location**: Sets where the binding is to occur, if applicable.
    The selections available here will be dependent upon the selection made in the Binding **Style**. Only Stapled and Stitched bindings have a **Location** option available to them.
  - **Angle**: Set Stapling or Stitching binding either horizontally, vertically, or at an angle (as supported by printer).
  - **Item count**: Select the amount of Staples or Stitches to use. The choice is between the default amount or selecting a specific number using the Count option.

    Tip: The options available to you in reality at print time will be printer dependent, so you will need to know the capabilities of your printer, or leave the value set to Default.

  - **Area**: The area where the binding can be applied.
- **Hole making** group:
  Hole making options are available only to *Ring, Comb* (wire and plastic) and *Coil* Binding **Styles**. The selections will need to be made at run-time based upon the types of binding options available that the printer supports.

- **Number of holes**: The number of holes to punch for the selected Binding option.
- **Pattern Catalog ID**: The Catalog ID of the selected Binding option.

Advanced Print Wizard navigation options

- **Preview** button: Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button: Click to produce print output according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout.

Sorting options

This page appears as part of the **Advanced Print Wizard** and the "Job Preset" on page 1359 wizard.

The **Sorting Options** page is used to sort the records in the output. Sorting is done from the top to the bottom, one after the other.

Sorting Settings

- **Use standard sort**: Sort using the fields below:
  - **Field Name**: Use the drop-down to select which field to sort on.
  - **Order**: Use the drop-down to choose Ascending or Descending.
  - **Add**: Click to add a new row to the sort list. The list that appears contains all the fields in the Data Model, as well as a special <Document Length> option which is used to sort by the number of pages in each document.
  - **Delete**: Click to delete the currently selected row in the list.
  - / **Move up / down**: Click to move the currently selected row up or down within the list.
- **Use external sort**: Sort the records using some external sorting software. A CSV file is exported, then sorted by the external application and the resultant sorted CSV file is returned and integrated, with all the records now being sorted according to the new

order within the CSV file.

> **Note:** ***External Sort commands must return a non-zero error code if an error occurs.***
>
> An external sort command could easily fail part way through processing and generate only a partial output file. Without receiving a return code from the external sort process, OL Connect cannot know if the sort has successfully completed or not. Thus the sort program must generate a return code, with a code of zero ('0') indicating success, and all non-zero results indicating failure.

- **External Sort Settings** group:
  This section of the dialog is only activated if the **Use external sort** option has been selected.
  - **Command**: Enter either the full path of the executable file that will sort the CSV file, or a valid Windows command line instruction to sort the records.

    Placeholders can be used within this command, and are available via the **Insert input/output file placeholder** options (expressed as either `${input}` or `${output}`) or via Job Creation runtime **Parameters** (expressed as `${params.xxxx}`, where the xxxx is the name of the runtime parameter).

    The Windows command line instruction should do something like the following:

    1. Do some processing of the input CSV file which OL Connect will pass through in the position of the `${input}` placeholder.
    2. Generate an output file that contains the sorted data and must be named according the file name OL Connect will pass through in the position of the `${output}` placeholder

    For example: `cmd /C sort /R ${input} ${output}`
    This would reverse the order of the `${input}` file, and sent the output to the `${output}` file.
  - **Timeout**: Enter the number of seconds to wait for an external sort command to complete before abandoning the external sort.
  - **Separator**: Enter the field separator used in the CSV file, such as a comma (,), pipe (|), semicolon (;), etc.
  - **Quote Character**: Enter the quoting character that wraps around any field that contains the separator.

- **Escape Character**: Enter the character use to escape the Quote character if it appears in the field value.
- **Line Ending**: Use the drop-down to select which line ending to use. The selections are: Windows *Carriage Return/Line Feed* combination (CRLF), Linux *Line Feed* (LF) or Apple Macintosh *Carriage Return* (CR).
- **Character Set**: Use the drop-down to select which character set to use when encoding the CSV file. This always defaults to UTF-8, as this caters for all possible characters, is relatively compact (in terms of Unicode character sets) and is compatible with standard ASCII.
- **Exported sort data** group:
    - **First row of sort data has field names** checkbox: select to have field names placed on the first line of the exported CSV file.
    - **Fields to export**: Lists the fields to export in the CSV file. The buttons to the right of the table provide the following functionality:
        - Click to open the Field Selection dialog, which allows selection of one or several from the available fields plus the document sheet count.

          Tip: For an example of how to apply sorting by document size, see this how-to: Sorting by document size.
        - Click to remove a field from the list.
        - Click to move fields up or down in the order of output.
    - **Record ID Field**: The Record ID field is a database *Primary Key* field, which is automatically added to the exported data file. The Record ID field name defaults to *RecID*, but can be changed here as desired.
- **Processing returns data** group: This allows the external sort application to introduce new data for each record. This data can be embedded in the metadata and used as the source for additional content within OL Connect.
  An example usage would be generating a postcode or postal barcode data from address details, making it available for use in OL Connect.
    - **Processing returns data** checkbox: Select this if the sort processing will be returning data. This activates the whole optional **Processing returns data** subsection.

- **First row of return data has field names** checkbox: select to have field names placed on the first line of the returning datafile.
- **Return Fields**: Lists the fields available in the selected data mapping configuration that can be used to sort the records. Fields can be added or removed by use of the Add datafield and Remove datafield buttons, or re-arranged with the / arrow buttons .
  Field names can be altered by selecting the field in the table, and editing the name.
  Fields can be made available to OL Connect via the "Include in meta data" checkbox. Click the checkbox beside the field name to make that datafield available as meta data.
- **Record ID Field** selection box: Select which return field is to be the Record ID field.
- **Sorting by** selection box: Select whether the sorting will be by the returned sort order or whether it is to be sorted on a selected datafield.
  - **Sequence Field** selection box: Select the datafield to be sorted on, if such was chosen in the **Sorting by** entry.

    Note: The selected datafield must be numeric.

Advanced Printing Wizard and Job Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Job Preset*** Wizard only): Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Grouping options

This page appears as part of the **Advanced Print Wizard** and the "Job Preset" on page 1359 wizard.

The **Grouping options** page separates the job output into multiple blocks that can then be physically separated using split sheets in the printer, or split into several print output files.

A typical usage of this feature is to create groups of mail pieces by size, so they can easily be inserted (into envelopes). For instance, one might want to group together all the "single page" mail pieces that fit within a C5 envelope and then put all larger mail pieces into an "oversized" category, that will go into larger envelopes.

- **Grouping Tabs**: Jobs can be grouped at three different levels. The three groups each have their own tab, and are as follows:
  - **Job Grouping Fields**
  - **Job Segment Grouping Fields**
  - **Document Set Grouping Fields**

  The Fields available to be used for any Grouping are contained within the **Available Fields** box.
  Any Fields that you want to use for Grouping need to be added to the **Selected Fields** box via the arrows found between the two boxes.
  Select the Field(s) you want to move and then click the appropriate arrow.
  Any fields that you decide don't need to be used in Grouping can be returned to the Available Fields box in the same fashion.
  Once a field is added to the **Available Fields** box, its *Sorting Option* can be selected by clicking in the "**Sorting Option**" column, and selecting the appropriate option.
  The options for sorting are either Ascending or Descending order.

- **Size Grouping** section: Check the checkbox to enable *size grouping*, which separates Documents into different groups, based upon their relative sizes.

  For example, selecting *Document Set Grouping* for the **Grouping Level**, *Documents* for the **Item to group**, **Group** by *Page Count* and then creating a page range from 1-5 and another page range 6 to Largest, will create two Document Set groups. The first will contain all Documents of 1 to 5 pages in length, and the second will contain any document of 6 or more pages.

  The options within Size Grouping are:

- **Grouping Level**: Use the drop-down to select which grouping level to use, between **Job**, **Job Segment** or **Document Set**. Only one grouping level can be selected.
- **Item to Group**: Use the drop-down to select which item to group. The item choices are **Job Segments**, **Document Sets** or **Documents**.

  > **Note:** A selection made here can over-ride a previously selected **Grouping Level** option.
  >
  > For example, if the **Grouping Level** was previously set to *Document Set*, then selecting *Job Segments* as the **Item to Group** will not make sense for that Grouping Level. Thus the **Grouping Level** would automatically be changed to the more appropriate *Job* selection.

- **Group by**:Use the drop-down to select what criteria the Items will be grouped by. The choices are **Page Count** or **Sheet Count**.
- **Size groups in reverse order** checkbox: Reverses the order of the groups created.
  By default, grouping is done from smallest to largest. Checking this option instead creates groups from largest to smallest.
- **Size grouping after normal grouping** checkbox: Check this option to firstly group using the selections made above and then secondly group upon page break grouping. This creates two different levels of grouping, applied in order.
- **Meta data property** edit box: Select a name for the *meta data* field that is created. This meta data field can then be used in other Print Wizard/Preset pages, such as in the "Job Output Mask Dialog " on page 1385.
- **Size Ranges**: Add (or remove) entries to this list to create new groups based upon the number of pages in the level selected above. All groups must be contiguous from 1-to-Largest and they must not contain any gaps.
  - **Range Name**: Enter a name identifying the range. It must be unique, but otherwise bears no impact on the range feature.
  - **From**: Enter the starting page number of the range. The first range must start with 1, all other ranges must be contiguous (the "From" range must be one higher than the previous "To" value).

- **To**: Enter the last page number for the range. The last range must end with a selection of "Largest".

Advanced Printing Wizard and Job Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Job Preset*** Wizard only): Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Meta Data options

This page appears as part of the **Advanced Print Wizard** and the "Job Preset" on page 1359 wizard.

The **Meta Data Options** page defines meta data entries that will be added to PDF output files and optionally added to AFP output (Enterprise edition only) where they are added as TLEs by default, but can be removed in the "AFP Options" on page 1388 page.

Meta data entries are ignored in all other output types, except when they are associated with "Additional Content" on page 1409 or used as conditionals or in output file names, in the Advanced Print Wizard or in "Output Presets" on page 1648.

> **Note:** Meta data properties have a length limitation of 255 characters.

The meta data entries can be added to any of these levels, as indicated by the tabs on top: **Job**, **Job Segment**, **Document**, **Document Set**, and **Page** Tags.

> **Note:** It is important to note that if Impositioning is used then the only meta data that will be available to the job thereafter will be the meta data at Job Segment level.

When using Impositioning everything within the Segment is Impositioned, so all meta data below the Job Segment level is lost.

In each of these levels, a list of entries is available in a table. The Table options are as follows:

- **Always create meta data for this level even when fields are selected**: Select this check box to create a blank meta data entry if no fields are selected. Done to ensure that a meta data store is always available, if required.
- **Tag Name** column: Name of the meta data tag added to this level. Once a tag has been added, its name can be edited by double-clicking on the Tag Name.
- **Source Type** column: Displays the type of field being used - either Text, Data Field or runtime Parameter (from Workflow).
- **Source** column: For Data Fields and runtime parameters only. Either the runtime parameter name or the Field name from the data mapping configuration whose value will be used for this tag.
- **Add meta data**: Click to add a new tag to the current level. Select from one of **Add field meta data**, **Add runtime parameter meta data** or **Add text meta data** from the Field Selection context menu. Those options do the following:
  - The **Add field meta data** option launches the Field Selection dialog.
  - The **Add runtime parameter meta data** option launches the Runtime Parameter Selection dialog.
  - The **Add text meta data** option inserts a new text meta data entry straight into the current meta data table.

  Once loaded in the table, the meta data fields *Tag Names* can be modified as desired. As can the source they point to, in the case of runtime parameters and data fields.
- **Delete**: Click to delete the currently selected meta data entry.
- / **Move Up / Move Down**: Click to move the currently selected tag(s) one position up or down.

Output Presets

The **Output Presets** (or print output presets) dialog displays a list of available presets and a summary of their settings. This dialog can be used to create new Presets or to edit and update existing Presets.

The Presets are all stored as individual files, using the Preset name and a "OL-outputpreset" file extension.

Once saved, Output Presets can be loaded in the **Print Wizard** or be used in the "Send to Server", or "Package" options, to re-use the selected Output values.

You can also send the preset to Workflow to use it in Workflow configurations. See "Sending files to Workflow" on page 593.

Dialog Interface

Configuration Selection section:

- **Name**: Enter the name for print output preset. This will become the print output preset filename, when the preset is saved.
- **Description**: Optional description for the print output preset.

  Tip: It is recommended that you enter a meaningful description here, so how the preset is to be used can be more easily understood in the future.

- **Production Options**: Select which of the following optional settings are going to .
  - **Booklet Imposition**: Select to add optional Booklet information. This opens the "Booklet Options" on page 1454 page of the wizard.
  - **Imposition**: Select to add optional Imposition information. This opens the "Imposition options" on page 1456 page of the wizard.
    This option is mandatory, if *Booklet Imposition* is selected.
  - **Separation**: Select to add optional separation. This opens the "Separation options" on page 1477 page of the wizard.
  - **Add additional content**: Select to add additional content to the output. This opens the "Additional Content" on page 1409 page of the wizard.
  - **Add Inserter marks**: Select to add optional print inserter marks. This opens the "Inserter options" on page 1464 page of the wizard.
  - **Print virtual stationery**: Select to add previously set virtual stationery.
    The virtual stationery in a Print template (also called "Media" on page 652) is in fact an image that is normally only visible in the Designer in the background of the template to make designing a template easier.
    If this option is selected, that image will also be printed. The output can then be printed on plain paper instead of pre-printed paper. Printing virtual stationery is also very useful when producing PDF output for digital viewing.

Chose a job preset selection:

- Select the job preset that is to be used for the metadata, if any is to be used in the output preset.

Click Next in this dialog to continue entering Output Preset data. This takes you to the first page of the Wizard, the "Print options" on page 1380 page.

Printer settings

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Printer Settings** page provides options for cut-sheet printers. It maps media types to printer trays.
It is available for AFP, IPDS, PCL and PostScript printers that are configured for cut-sheet printing.

Note: AFP and IPDS are only available in OL Connect Enterprise edition.

Note: Not all the settings will be available for all printers. Only those printers whose Printer Definition supports the extra information will have the **Position**, **Weight**, **Type** or **Color** options available.

- **PostScript Options** group (only available in PostScript) output:
  - **Title** selection: Enter the Title to add to the PostScript output.
    The default is `${template}`, which uses the Template name.
- **Tray Mapping** group:
  - **Map media by** options: Select from following choices:
    This entry will only be available for those printers that support the extra information.
    - **Media Attribute** displays all Media details, except the Tray selection.
    - **Tray** displays just the Media name and Tray selections.
    - **Both** displays all Media details.
  - **Media** columns:
    - **Media**: Lists the Media name, as defined in the template.
    - **Tray**: Use the drop-down to select which Tray to send any page using the media.
      Only available if **Tray** or **Both** is selected as **Map Media by** selection.

- **Position**: Enter a MediaPosition option on the printer to define the media to use.
  This entry will only be available for those printers that support the extra information.
- **Weight**: Enter a weight for the paper.
  This entry will only be available for those printers that support the extra information.
- **Type**: Use the drop-down to select which type of stock to use on the printer.
  This entry will only be available for those printers that support the extra information.
- **Color**: Use the drop-down to select which color the paper should be on the printer.
  This entry will only be available for those printers that support the extra information.

- **Media/Tray Table** buttons:
  - **Add**: Adds a new Media/Tray entry in the table.
  - **Delete**: Deletes the current Media/Tray selections from the table.
  - / **Move Up / Move Down**: Move the selected Media/Tray entries up or down within the table.
  - **Import Tray Settings**: Import the Media/Tray settings from a Connect Template.
    This entry will only be available for those printers that support the extra information.
    The options are to:
    - **Import from current template**, which will import any Media/Tray settings from the current Connect Template into the table.
    - **Import from template file**, which allows you to browse for a Connect Template to import the Media/Trays settings from.

    There is no restriction on how many Templates you may import settings from.

- **Output Destination** group: These settings allow for the selection of output bins.
  It is available for AFP, IPDS and PCL printers whose Printer Definition caters for multiple output bins. If the selected Printer Definition does not support bins, this group will not be visible on the page.

Note: AFP and IPDS are only available in OL Connect Enterprise edition.

The table has the following choices:

- **Trigger**: Select what the trigger will be for changing bins. The options are *>Page*, *>Sheet*, *>Document*, *>Document Set* or *>Job Segment*.
- **Destination**: Select which bin should be used when this trigger is met. The options will be the bins available to the selected Printer Definition.
- **Rules**: Defaults to true, but can be altered at will. Double clicking in the field brings up the Rule Editor, which can be used to construct the rule(s) that determines whether this bin is to be used or not.

The options (both button and right mouse click context menu) available to the table are:

- **Add**: Adds a new bin selection choice to the table.
- **Delete**: Deletes the current selection(s) from the table.
- / **Move Up / Move Down**: Move the selected entries up or down within the table.
- **Edit ...**: Edit the current selection.

Default output destination selection: Select the default output bin from those bins available to this Printer Definition.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on the facing page page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on the facing page page have been completed or not.

- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Print options

The **Print Options** page is the first page of the **Advanced Print Wizard** (used for both Production and Proof printing in the Designer) and the second page of the "Output Presets" on page 1375 wizard.

This page is the most important of both the Advanced Print Wizard and the ***Output Preset*** Wizard. All other pages that appear in the Wizard are determined by the selections made on this page.

The options that don't appear in the ***Output Preset*** Wizard (as noted below), do appear in the ***Job Preset*** Wizard. Only the Advanced Print Wizard contains all the options, but it cannot save your settings to file like the preset wizards allow.
For more information about print preset files, see "Print Presets" on page 1644.

The choices can be broken down as follows.

Printer

- **Model**: Use the drop-down to select the printer language that will be generated.

  OL Connect comes with several bundled print output Models which cover a large range of industry standard print output types. Depending on the license, these include AFP, IPDS, PCL (if licensed), PDF, PPML and PostScript (including the optimized VIPP and VPS variants), with a range of quality settings available for each.

  > **Note:** AFP and IPDS are only available in OL Connect Enterprise edition.

  The bundled print output Models do not cater for *all* types of printers, however. There are simply too many different printers with their own individual (often quirky) settings and capabilities to cater for all of them with just a few standardized print Models.

  To use OL Connect with a particular printer Model you must obtain a tailored OL Connect Printer Definition that defines that printer and its capabilities for OL Connect. For example, to cater for IPDS or AFP printers that support Encapsulated Postscript resources included through an Include Object (IOB) structured field. Such a print output Model would need to be customized for you. These customized Printer Definitions will be created by Upland OL Support, in conjunction with information you provide about both your printer and your specific needs. Support will then provide you with a customized Printer Definition that supports the desired functionality with your specific printer. By default, OL Connect initially displays just a single generic *PDF*

output option, to keep the interface clean. Other print output types can be added to the Printer Model drop-down list at any time via the **Import Definitions** button.

The **Import Definitions** options are:

- **Import Definition**: Used to add any customized or tailored Printer Definition that Upland Software, Inc. has prepared for you, specific to your printer.
- **Create Definition from PPD**: Used to create your own customized PostScript Printer Model through the printer manufacturer provided PostScript Printer Definition (PPD) file. This option launches the "Dynamic PPD Options" on page 1468 page later in the Printer Wizard, which allows you to setup the printer.
- **Edit available printers**: Add extra Printer Definitions from the selection that come installed with OL Connect. Adding a printer from this list is probably all you will ever need.
- **Export printer definition from output preset**: Use to extract a Printer Definition from a new or modified Printer Model. This option is only available when the Printer Model currently loaded differs from those previously saved or installed.

For more information on how to add Printer Definitions to OL Connect, see "Adding print output Models to the Print Wizard" on page 1652.

Output options

- **Output Local**: Select to have the output created using the local Print Server, instead of the OL Connect Server (see "The OL Connect server" on page 18 for an explanation).
- **Output Type** choices:
  - **Prompt for file name**: Select to output to a local file on the hard drive. When this option is selected, no other configuration is necessary. A Save As dialog will appear to allow selection of the folder and file name.
  - **Directory**: Select to output to a local folder on the machine.
    Selecting this will open the **Directory Options** sub-group, which has these options:
    - **Job Output Mask**: The name of the file that will output.
      You could write the Job Output Mask directly into this edit box (for a list of available variables see "Print output variables" on page 1659), or you could create a Mask via the Options button. This opens the custom dialog:

"Job Output Mask Dialog " on page 1385.
The Job Output Mask may contain (dynamic) folder names, for example: `${document.metadata['Country']}\${template}`.
The evaluated value of the Job Output Mask is taken as a path **relative** to the folder specified by the Job Output Folder (the next option in this dialog). The Job Output Folder must exist, but folders specified in the Job Output Mask will get created if they don't exist.
Spaces inside the Job Output Mask (i.e., not at the beginning or end) are allowed, but any leading and trailing spaces are trimmed from the Job Output Mask to prevent errors.

- **Job Output Folder**: The path on the disk where the file is produced. Please note that the folder must exist, or output will fail when produced through the server.

- **LPR Queue**: Select to send the print job to an LPR queue. It is assumed that the print technology is supported by the system receiving the LPR job.
  - **Local Printer**: The IP or host name of the printer or machine where the LPD is installed and will receive
  - **Queue Name**: The queue name that will accept the job on the LPD. Default is generally "auto".
  - **Job Owner Name**: Optional entry for adding the name of the job owner.
  - **Job Name**: The name of the output file. You can use ${template} as a variable for the name of the Designer Template used to generate the output. (See also: "Print output variables" on page 1659.)
  - **Timout (seconds)**: Set the timeout duration, in seconds. Default is 60 seconds.
  - **Retry Attempts**: Set the amount of job retry attempts. Default is twice.
- **Windows Printer**: Select to send the Print Job to a printer queue (note that this is an actual printer queue, not a Workflow Printer Queue).
  For non-PDF output, the job will be first rendered as a PDF before being printed through the Windows driver.

  For PDF output the Adobe Library is used.
  - **Windows Printer**: Use the drop-down to select the Windows printer queue where the job will be sent.
  - **Job Owner Name**: Optional entry for adding the name of the job owner.

- **Job Name**: The name of the output file. You can use ${template} as a variable for the name of the Designer Template used to generate the output.
- **PDF Rendering Options** sub-group (only available for *PDF output*):
  - **Auto-rotate and center**: Check to automatically select the page orientation that best matches the content and paper.
  - **Choose paper source by page size**: Check to use the PDF page size to determine the output tray rather than the page setup option. This option is useful for printing PDFs that contain multiple page sizes on printers that have different-sized output trays.
  - **Scale**: Chose from one of the following scaling options:
    - **None**: Select to not scale any page, whether it fits or not.
    - **Expand to printable area**: Select to expand any page to fit the page area. Pages larger than the paper size are not re-sized.
    - **Shrink to printable area**: Select to shrink any page to fit the page area. Pages smaller than the paper size are not re-sized.
  - **Prompt for Printer Properties**: Select to run the Windows Driver preferences dialog as part of the printing process, to allow selection of printer functionality.

Production Options

- **Runtime Parameters**: select "Runtime Parameter Options" on page 1360 to store information which can be used at runtime for comparisons against conditions within Job Creation, or for use with external sorting programs.
- **Override Finishing options** (not available in ***Output Preset*** Wizard): Check to configure custom "Finishing options" on page 1366, such as binding.
- **Booklet Imposition**: Check to tell the printer to generate a booklet for the print output. Booklet options are set in the "Booklet Options" on page 1454 page.
  This option is unselected by default unless selected in the Designer "Print section properties" on page 1200.
- **Separation**: Check to activate the "Separation options" on page 1477 page of the wizard.
- **Add Inserter marks**: Check to enable Inserter mark functionality, which is set in the "Inserter options" on page 1464 page.

- **Use grouping** (not available in ***Output Preset*** Wizard): Check to configure grouping of output into jobs, job segments or document sets. See "Grouping options" on page 1372.
- **Include meta data** (not available in ***Output Preset Creation*** Wizard): Check to add meta data to the output. This can be done at Job, Job Segment, Document, Document Set and Page level. See "Meta Data options" on page 1374.
- **Imposition**: Check to enable Cut & Stack Imposition, which is set in the "Imposition options" on page 1456 page.
- **Add additional content**: Check to activate the "Additional Content" on page 1409 page of the wizard.
- **Print virtual stationery**: Check to enable virtual stationery in the output.

Records

(Not available in ***Output Preset*** Wizard.)

- **Record Range**: Allows selection of a range of records or a custom selection.
  You can specify individual records separated by semi-colons (;) or ranges using dashes.
  For example: 2;4;6-10 would print pages 2, 4, 6, 7, 8, 9 and 10.
- **Apply filtering and sorting to record selection**: Check to activate the "Data filtering options" on page 1363 (to filter out certain records) and "Sorting options" on page 1368 (to sort the remaining records) pages of the wizard.

Copies

(Not available in ***Output Preset*** Wizard.)

- **Copies**: Enter the number of copies to print, of each record.
- **Collate**: When printing multiple copies you can check this checkbox to have the record copies printed together.
  For example, in a three record job the records would print out as 1-1-2-2-3-3 rather than 1-2-3-1-2-3.

Pure Color Thresholds

This section is valid for AFP, IPDS (Enterprise edition only)and PCL only. It applies to elements within the record that are shades of grey, rather than black or white.

- **Black Threshold Percentage**: The percentage of shading at which the element will appear as full black, rather than dark grey.

- **White Threshold Percentage**: The percentage at which the element will appear as full white, rather than light grey.

Advanced Print Wizard navigation options

- **Load** button (not available in ***Output Preset*** Wizard): Click to select a previously created Output Preset.
  This will change the Advanced Print Options to match the entries contained within the Preset.
- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Job Output Mask Dialog

This dialog allows you to select a Job Output Mask using dynamic Metadata.

These are the options available in this dialog:

- **File Separation**: Select from options that allow various print output separations. The choices are not to separate (None), or to separate upon **Job** (Job/Job Segment) or **Document** (Document/Document Set) criteria.
  What Metadata options that will subsequently be made available will be determined by the choice made here.
  The choices are:
  - **None / Job**: Neither of these options allow access to much in the way of metadata.
    Options available thereafter are:

- **Meta data**. With this selection only Page Break Grouping fields will be available. You need to select a different grouping level to access other metatdata fields.
    - **Use Page Break Grouping Metadata**: Allows you to create some Page Range metadata options.
      The choices are:
        - Add a Page Range.
        - Delete a Page Range
        - **Page break range metadata field name**. Edit the choice of name for the Page Break Range metadata field. This name can be manually altered in the edit box, before adding to the Job Output Mask, via the Insert button.

          The entry will be added at wherever the cursor is currently positioned within the Job Output Mask edit box. This defaults to the beginning of the mask, if not otherwise specified.
- **Job Segment / Document / Document Set**: All of these selections provide the same metadata options.
    - **Meta data**: Allows you to add datafile fields, or text strings to Job/Document metadata.
        - **Tags** table: This table contains any and all extra metadata entries that will be made available for this
            - Add datafile field or text to metadata.
              To add datafile field, select *Add field meta data* option, and select from the available fields in the datafile.
              To add text entry, select *Add text meta data* option, and then edit the text within the "Source" column.
            - Delete the selected entry from the Tag table.
            - / Move the selected entry up or down the Tag table order.
            - Add the selected entry to the Job Output Mask. The entry will be added at wherever the cursor is currently positioned within

the Job Output Mask edit box. This defaults to the beginning of the mask, if not otherwise specified.

- **Use Page Break Grouping Metadata**: Allows you to create some Page Range metadata options.
  The choices are:
  - Add a Page Range.
  - Delete a Page Range
  - **Page break range metadata field name**. Edit the choice of name for the Page Break Range metadata field. This name can be manually altered in the edit box, before adding to the Job Output Mask, via the Insert button.

    The entry will be added at wherever the cursor is currently positioned within the Job Output Mask edit box. This defaults to the beginning of the mask, if not otherwise specified.

- **Job Output Mask**: This contains the current Job Output Mask, as constructed from within this dialog.

  You could manually write the Job Output Mask directly into this edit box, or use the General options button to add standard metadata variable options to the Job Output Mask.

  For a list of the standard metadata variable options available see "Print output variables" on page 1659.

  Note: Anything inside a `${…}` placeholder is evaluated as a JavaScript expression, with the eventual result replacing the placeholder. Everything else is treated as literal text.

  For example: The Job Output Mask `${template}_${system.time,'yyyy-mm-dd'}.${template.ext}` applied to a Template called "Testing", that was run on January 1 2017 and which outputted a PDF file would resolve to: *Testing_2017-01-01.pdf*.

  The characters : `* ? " < > |` (and `/` at the beginning and end) are not allowed as literal characters in the Job Output Mask, but may be used in a `${…}` placeholder.

AFP Options

> **Note:** This page is only available in the **Enterprise** edition of OL Connect.

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **AFP Options** page is shown only when a AFP Print output type is selected in the Print Options dialog. It is used to select AFP specific options.

- **Page Groups**
  Select (or deselect) what Page Grouping options are to be created for AFP output.
  - *Job Segments*.
  - *Document Sets*.
  - *Documents*.
- **Options** group:
  AFP specific output options, relating to AFP **T**agged **L**ogical **E**lements (TLEs)
  - **Create TLEs for all metadata properties**: Used to control whether automatic TLEs are to be created from job metadata properties and included in the AFP output, or not.

    Prior to Connect 2018.2 TLEs were always created for job metadata properties. This checkbox provides an option for switching off that automatic TLE creation.

    > **Note:** If you want to create your own TLEs it can be useful to disable automatic TLE creation. This reduces the chance of unexpected clashes.
- **Rules** selection:
  TLEs and NOPs can be controlled by configurable rules. These rules are organized in groups. Each rule is associated with 1 or more custom AFP records. Those records get inserted into the output when their rule is true.
  - **Add**. Add a Rule or rule Group. All rules need to reside within rule Groups.

    Groups:
    All rule Groups have a name and trigger.
    The name has no functional purpose other than in identifying the Group within the Table.
    The trigger determines when the rules are evaluated, and they can be set to

either *Page*, *Document*, *Document Set* or *Job Segment* triggers.

Rules:
Rules can be based upon other groups of rules, or based upon Positioning, Size or Metadata (if Metadata is available).
Rules within a group act as alternatives, with the first rule equating to "true" being chosen.

- **Edit**. Edit a Rule or Group.
  The editing options are the same as those that applied when the Group or Rule was initially created.
- / **Move**. The order that groups appear in, controls the order in which the AFP records they control get inserted. Use these controls to sort the order as desired.
- **Delete**. Delete the selected Rule(s) or Group(s) from the table.

- **Condition** display box: This displays the contents of the currently selected Rule or rule Group. The contents cannot be edited.
- **Records**:
  Add a TLE or NOP to be applied when the current condition is met. This adds embedded textual metadata in the output, that does not impact upon the appearance of the output in any way.

  TLE (**T**agged **L**ogical **E**lements) records:
  A TLE structured field assigns an attribute name and an attribute value to a page or page group. The TLE structured field may be embedded directly in the page or page group, or it may reference the page or page group from a document index.
  When a TLE structured field is associated with a page group, the parameters of the TLE structured field are inherited by all the pages in the page group and by all other page groups that are nested within that page group.
  The scope of a TLE is determined by its position with respect to other TLEs that reference, or are embedded in, the same page or page group. The TLE structured field does not provide any presentation specifications and therefore has no effect on the appearance of a document when it is created.

  NOP (**N**o **Op**eration) records:
  In some cases, instead of using TLEs, a choice might be made to use NOP records (a field that is bypassed when processing output, and produces no output) with custom data to come up with an entirely proprietary way of adding metadata.

- **Add**. Add either a TLE or NOP record to be associated with the current Condition.
  The TLE requires both a name and some textual content.
  The NOP requires just the textual content.
- **Edit**. Edit the TLE or NOP entry.
- / **Move**. Use these controls to move the selected entry(s) up or down within the Records table.
- **Delete**. Delete the selected entries from the table.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

PDF Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **PDF Options** page is shown only when a PDF Print output type is selected in the Print Options dialog. It is used to select PDF specific options.

> **Note:** Ideally, PDF should not be used for generating Mixplex output. You should instead use a print format that supports Duplex/Simplex switching. PDF is not suitable for this as it has no device control.

- **PDF Creation Group**
  - **Type**: Use the drop-down list to specify what PDF format will be generated. The options are:
    - Standard PDF (PDF)
    - Archive format PDF (PDF/A-1b)

      Note: PDF/A output created from a template that uses CMYK colors will be much bigger than PDF/A output created from a template that uses RGB colors, because PDF/A needs to contain a color profile and the CMYK color profile is rather big compared to a RGB color profile. If output size matters it is recommended to avoid using CMYK colors.
    - E-invoice format PDF (PDF/A-3b)
    - Graphics format PDF (PDF/X-4)
    - Variable data printing format PDF (PDF/VT).
  - **Font Creation** selection: The choices are:
    - **Simple fonts when possible (smallest size)**. This is the default option.
      This option uses Simple fonts in cases where the fonts do not use too many characters. It automatically switches to using composite fonts when the number of characters becomes too large to fit a Simple font. This allows omitting fonts that are considered "standard fonts" in the PDF reference, which helps reduce file size.
      The drawback to this option is that some PDF viewers do not support it properly, which can cause viewing issues with jobs using non-Latin1 encoded characters.

      Note: A Simple font can be a Type1 font, a TrueType font or a Type3 font.
      One of the main characteristics of a Simple font is that its glyphs are selected using a single byte character code, limiting the number of glyphs to a maximum of 256.

      For more information on Simple fonts, see Abode's documentation regarding Portable Document Format files.

- **Always create CID fonts (best compatibility)**. This option always uses composite fonts (CID fonts), regardless of the number of glyphs in the font. CID fonts use 2 byte ids to select glyphs, so this option always leads to larger file sizes.
  The benefit of this option is that all the fonts get embedded in the output, making for the most portable of PDF outputs.
  Customers using Asian fonts should generally use the CID font option.

Note: Any pre Connect 2018.2 Output Presets will use the **Simple Fonts** default, unless updated in Connect 2018.2 or later.

- **Embed standard fonts**: Click to embed the 14 standard system fonts within the PDF output. This increases the output filesize but makes the PDF output truly portable. Such PDFs print as displayed on screen, regardless of whether the 14 standard fonts are present on the target printing system or not.

Note: This box is automatically selected for all but standard PDF outputs, as fonts are always embedded in such output types.

- **Pass-through PDF resources**: Click to have PDF resources used *as-is*, without any additional processing. This guarantees the fidelity of any PDF graphics used within the template will be retained in the output.

  This option is only available for standard PDF output.

Note: Connect tries to write content in the best way possible, depending on the chosen output format and optimization settings. Selecting *PDF pass-through* means the output will be less optimized, which typically produces somewhat larger files.

Note: Any non-content features in the original PDF will get stripped out of the original PDF. These include document navigation (outline), local links, hyperlinks, annotations (markup annotations, comments, etc.), digital signatures, multimedia features, interactive forms, and tagged PDF features (marks in the page content will stay, but the logical structure is gone).

Note: Encrypted PDF files are **not supported** in *PDF pass-through* mode.

- **Keep attachments**: Click to keep any PDF attachments, without any modification. This allows processing of PDF input files with attachments, producing either standard or digitally signed PDF files with attachments.

  This option is not available for PDF/A-1b output.

  > **Note:** This function will only work when a PDF with attachments is loaded into DataMapper as an input data and "bypass content creation" is selected.

  > **Note:** Generating a PDF and retaining it's attachment directly from the Designer may not be possible. This can be achieved by using an automation tool, though, such as OL Connect Workflow or Automate.

- **Keep Tagged PDF (experimental)**: Select to retain any Tagged PDF information that was present in the input PDF file.

  > **Note:** This option is experimental only, at this time.
  >
  > Please feel free to use it and to let Upland Software, Inc. know if you encounter any difficulties. We will endeavor to improve this feature based upon customer feedback.
  >
  > For known limitations see "Tagged PDF Limitations" on page 1402.

Metadata

Use this Tab to add optional Metadata to the PDF output.

The fields that can be configured are the Title, Author, Description and Keywords. These metadata entries are those that are displayed in the Adobe Reader *Document Properties* dialog.

The values for all the fields can be either static or dynamic (such as values obtained from data mapping) or a combination of both.

Metadata

- **Title**: Add the document name(s). If no entry is made, this defaults to the existing document name (*${template.base}*).
- **Author**: Add the document author, if desired. This can be either a static or a dynamic value, or a combination thereof.
- **Description**: Add an optional description about the document. A static value is most likely for this field, but it can contain dynamic value(s) as well.
- **Keywords**: Optionally add any Keywords that might be beneficial to include. These can be either a static or dynamic values, or a combination of static and dynamic values.
- **Keep PDF/A-3 extension schema** checkbox (option only available for PDF/A-3b output): Click to retain any PDF/A-3 extension schema metadata, if the original PDF input file contained such.

Initial View

Use this Tab to select how the PDF output file is to be displayed upon initial viewing.

The options provided are largely the same as those seen in Adobe Acrobat.

- **Magnification and Layout** selection:
    - **Navigation Tab**: Select what panels should open in the Reader application when viewing the PDF.
    - **Page Layout**: Select how many pages should be displayed upon PDF opening. The choices are:
        - *Default* = Single Page.
        - *Single Page* = Starts with a display sized to fit one entire page, and scrolling down thereafter displays only entire page(s).
        - *Single Page Continuous* = starts with a display of one page. Scrolling down thereafter scrolls onto the next page, with both the first and second page in display.
        - *Display Pages in Two Columns, Left* = the pages appear in two columns, with the odd-numbered pages on the left.
        - *Display Pages in Two Columns, Right* = the pages appear in two columns, with the odd-numbered pages on the right.
        - *Two Pages, Odd Pages Left* = display the pages side by side, with odd-numbered pages on the left .
        - *Two Pages, Odd Pages Right* = display the pages side by side, with odd-numbered pages on the right.
    - **Page Magnification**: Select the level of magnification. The choices are the same as those presented in Acrobat Viewer.
    - **Open to Page**: Select which page is to be the first of the pages to be displayed.
- **Window Options**: Select which of the following application window options to apply:
    - **Resize window to initial page**: This has the application open to the same size as the initial page.
    - **Centre Windows on screen**: Centre the application on screen.
    - **Open in full screen mode**: Open the application in full screen mode.
- **User Interface Options**: Chose which application interface options to hide, if any.
  Select from the following: **Menu bar**; **Tool bars**; **Windows controls**

Digital Signature

A digital signature identifies the person signing a document, similarly to a conventional handwritten signature.

A digital signature is more difficult to forge than a handwritten signature as it contains encrypted information which is unique to the signer and which can be password protected and verifiable.

> **Note:** OL Connect currently supports only a single Digital Signature per PDF output file.

If the PDF has attachments, signing it requires a change in the print process via Workflow. The Workflow process is as follows:

1. Use the PDF as an input in the Execute Data Mapping task and enable **Bypass content creation**.
2. Feed the result of the Execute Data Mapping task to the Create Job task. Omit the Create Print Content task.
3. Next up is the Create Output task which will then sign the PDF while preserving any attachments.

- **Digital signature**: Check the box to enable the integration of a digital signature within the PDF

  The Digital signature table contains the Digital signature Keystore and associated Signature.
  The table has the following columns:

  - **Name**: The user-defined name of the Keystore\Signature.
  - **Alias**: The user-defined alias for the signature.

  The table allows the following options:

  - **Add Keystore**: opens the "Keystore" on page 1398 dialog for adding a Digital signature Keystore. Digital signatures *require* a Keystore, so initially only the **Add Keystore** option will be available.
  - **Add Signature** button only becomes active once a Keystore has been added. Selecting this option opens the PDF Signature dialog for adding a new PDF signature to the Keystore.
  - **Edit**: Click to edit the currently selected item. Either the Keystore or PDF Signature.
  - **Delete**: Click to delete the currently selected item. Deleting the Keystore will also delete the associated Signature.

Optimize

Use the options in this tab to optimize the PDF output.

Optimize

- **Optimize Images** selection: Select to enable optimization of images, with different settings available for Color Images, Grey Images and Monochrome Images.
  Each of the image types share the same basic options, with individual options tailored to the selected Image type also available.
  The options are:
    - **Compression**: Select what compression to use, if any.
      The default option is to *Retain Existing* settings, which disallows any further alteration of compression for this Image type.
    - **Downsample Filter**: This choice becomes available only when a Compression type other than *Retain Existing* is selected.
      Select which Downsample Filter to use, if one is to be used. The default is to not include one (*Off*).
      If a Downsample Filter has been selected, then set the desired display resolution in Pixels Per Inch (PPI).
      Select the maximum resolution (the **when Images PPI is above** value), and what resolution is to be used if the selected maximum is exceeded (the **set PPI to** value).
- **Optimize for fast web view**: Check this box to allow linearizing the PDF output. This allows readers of the PDF to start reading the PDF online before the entire PDF downloads.

Security

Use the options in this tab to set the PDF output security. The options are similar to those available in Adobe Acrobat.

When PDF output files of OL Connect are encrypted, the **encryption method** used, is always AES256. This encryption method was introduced with PDF version 1.7. This means that older readers may not support the encryption method and will therefore not be able to decrypt PDFs that were encrypted with OL Connect. For instance, the Adobe Acrobat readers support AES 256 as of version 9 or later.

Note that even though AES 256 encryption is used, PDF output from OL Connect may still have PDF 1.5 in its headers.

Security

- **Open Document** checkbox: Select this option to add password protection to the output PDF, which allows the document be opened only after entering the same **Password** as that selected here.

  To view the password entry in plain text, or to toggle back to masked view, click the Password button.

  - **Password is dynamic** checkbox. Select this if you wish to use a variable (such as a metadata field or a data file field) for the password.
    The datafile selection format is ${datafield}. For example, *${document.metadata.password}*
- **Permissions**: checkbox: Select this option to set optional Permissions. These include a choice of the following:
  - **Allow Printing**: select whether printing is to be disallowed (*None*), or whether to restrict to low quality printing only (*Low Quality*), or whether to allow full printing (*High Quality*).
  - **Changing the document**: whether the document itself is editable, with changes allowed.
  - **Content copying**: whether the contents of the document are allowed to be copy and pasted or not.
  - **Annotations and comments**: whether the document can be commented upon.
  - **Document assembly**: whether the Document assembly is available.
  - **Form filling**: whether PDF forms elements are allowed to be filled.
  - **Screen readers (Accessibility)**:whether the PDF will be WCAG capable and allow screen readers to convert the text into spoken word.
  - **Permission password**: Select a **Password** that allows the Permissions to be edited in the output document.

    To view the password entry in plain text, or to toggle back to masked view, click the Password button.
    - **Password is dynamic** checkbox. Select this if you wish to use a variable (such as a metadata field or a data file field) for the password.
      The datafile selection format is ${datafield}. For example, *${document.metadata.password}*

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Keystore

The security certificate **Keystore** dialog appears when adding or editing a keystore from the "PDF Options" on page 1390 page. This dialog allows you to select a keystore with a private key.

The keystores currently supported by Connect are:

- JKS (Java Key Store) format.
- JCEKS (Java Cryptography Extension Key Store) format.
- PKCS#12
- PKCS#11

Note: *JKS* and *JCEKS* represent file based keystores in a Java proprietary format. The Java keytool command line application can be used to create and manage keystores in this format.

Type *PKCS#12* also represents a file based keystore, but in the PKCS#12 format.

Type *PKCS#11* represents a keystore on a hardware device. These hardware devices must first be setup in the "Hardware for Digital Signing preferences" on page 1053 **Preferences** page, before they become available as an option in this dialog.

These are the options available in this dialog:

- **Name**: Enter a name for the keystore to describe it within Connect.
- **File**: Enter the path to the keystore file, or use the Browse button to locate the file. This option is not relevant for *PKCS#11* hardware devices.
- **Keystore** properties group:
  - **Type**: Use the drop-down to select the appropriate type of keystore format.
  - **Password**: If the keystore is password protected, type in the password that secures the keystore.
  - **Repeat Password**: Re-type in the password that secures the keystore. Once this is done the two Password entry boxes will no longer have the red cross icon (indicating incomplete or unselected) flag beside them.
- **PKCS#11** properties group:

- **Module Name**: Select which secure hardware device (USB tokens, smart cards, and Hardware Security Modules) to connect to through PKCS#11. These hardware devices must first be setup in the "Hardware for Digital Signing preferences" on page 1053 **Preferences** page, before they become available as an option in this dialog.
- **Slot list index**: Select which hardware slot to use.

  > **Note:** Slots are logical partitions in the hardware device. In case of Hardware Security Modules, there could be hundreds or more slots are available while in the case of smart cards, there would likely only be one slot available.
  >
  > Specifying both a *Slot list index* and a *Slot ID* is not allowed. Enter one or the other, or neither.

- **Slot ID**: Select which hardware Slot ID to use.

  > **Note:** Slots are logical partitions in the hardware device. In case of Hardware Security Modules, there could be hundreds or more slots are available while in the case of smart cards, there would likely only be one slot available.
  >
  > Specifying both a *Slot list index* and a *Slot ID* is not allowed. Enter one or the other, or neither.

- **Properties file group**:
  - **File**: Load optional keystore properties file. Properties files could be used for storing the password, or similar.

PDF Signature

The **PDF Signature** dialog appears when either adding or editing a Signature on the "PDF Options" on page 1390 page.

The PDF digital signature contains information about the signer, the location and reason used for signing and the time of creation.

The digital signature can be made visible on the output, with the dimensions of the signature box being configurable.

Note: OL Connect currently supports only a single Digital Signature per PDF output file.

- **Name**: Enter a name that describes the signature entry.
- **Keystore name**: Use the drop-down to select which keystore the signature is pulled from.
  These keystores are set in the "Keystore" on page 1398 dialog, called from the "PDF Options" on page 1390 page.
- **Signature Properties group**: These are optional Metadata fields associated with the signature, which can be omitted.
  - **Location**: The CPU host name or physical location of the signing.
  - **Reason**: Records the reason for the signing.
  - **Contact**: Information to enable a recipient to contact the signer to verify the signature. For example: a phone number.
- **Key group**: Refers to a key from the keystore.
  - **Alias**: The user-friendly name of the key.
    To select the alias directly from those present in the specified keystore, press the **Chose Alias from keystore** button ( ) to the right of the edit box. This launches the "Chose Alias from Keystore" on page 1402 dialog.
  - **Password**: Enter the password for the key (the same password as was entered in Key Store).
  - **Repeat Password**: Re-enter the password for the key (same as previous).
- **Timestamp**: Check to enable time stamping authentication.

  Note: Not available for signatures set to use Adobe.PPKLite Handler.

  - **Timestamp Server URL**: Select the Time Stamp Authority (TSA) URL address.
    One example of a TSA is https://freetsa.org.
  - **Account**: Account name specific to the TSA server chosen.
  - **Password**: Password specific to the TSA server chosen.
  - **Repeat Password**: Repeat of password.
- **Visible Signature**: Check to add a visible signature to the PDF file.

  The signature is added to the specified page of each output file.

The signature will be created using Arial 10pt black font. If this font does not exist on the production machine, a substitute font will be used.

If the digital signature contents exceed the dimensions of the configured box, the text will be scaled down.

> **Note:** When creating PDF/VT-1 or PDF/X-4 output, the visible signature should be located outside the visible page area (trimbox/bleedbox).
>
> The Weaver engine logs a warning message if this is not the case.

The default measuring option is points (pt) but metric options (mm / cm) can be manually selected.

- **X**: Enter the horizontal distance between the left side of the page and the left side of the signature.
- **Y**: Enter the vertical distance between the top of the page and the top of the signature.
- **Width**: Enter the desired width of the signature.
- **Height**: Enter the desired height of the signature.

Select which **Page** the signature is to be embedded in:

- **First Page**
- **Last Page**
- **Specific Page**. Enter which page in the **Page number** edit field, which becomes enabled when this option is selected.
  If the output file does not contain the specified page, then the signature will be added as an invisible signature to the PDF output file.

Advanced Print Wizard navigation options

- **Preview** button: Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.

- **Print** button: Click to produce print output according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout.

Chose Alias from Keystore

Select from which alias to use, from the list of aliases for keys that can be found in the keystore. If any such aliases exist and can be read.

If the keystore cannot be accessed to retrieve those aliases, a message will be shown stating that the keystore cannot be accessed.

If the keystore does not contain any aliases or if they cannot be read, then a message will be shown stating that aliases could not be obtained.

Tagged PDF Limitations

This is and experimental feature at this stage. Not every Tagged PDF file is expected to work.

The following is a list of **Known Issues** at time of release:

- While all files of the PDF/UA Reference Suite 1.1 can be processed and most will keep all of the Tagged PDF data (the output may no longer by PDF/UA compliant), a few files will loose part of the logical structure.
  In this case the problems are with files that contain re-usable content, technically referred to as “Form XObjects”, that also have tagged content.
- Keep Tagged PDF does not work when the output preset also has Separations enabled. Such as:
  - Splitting at a fixed number of sheets will fail with an error.
  - Splitting at document set, job segment, or job boundaries level will produce PDF with problems in the tag structure. While they might look okay, they are not correctly Tagged PDF.
  - Splitting at document boundaries may produce correct Tagged PDF, but this can also depend on how document boundaries were done on the input file(s).
- Impositioning will produce correct PDF output, but the output will not be Tagged PDF.

- Adding content during output creation will still produce Tagged PDF output, but the additional content will not be tagged.

These issues will be addressed in future releases, as the Tagged PDF feature is improved.

PPML Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **PPML Options** page is shown only when a PPML Print output type is selected in the Print Options dialog. It is used for tray selection.

PPML Tray Options

Use the drop-down list to specify what output Tray Options to use, if any. The options are as follows:

- **None**: No Tray Options specified. This is the default selection.
- **SetPageDevice**: This provides a table where one can allocate job Media to different types of print paper stock and assign them to the output via the SetPageDevice entry. The choices are a selection of common print stock Types and Colors.
  - **Media** column: The available job Media . These will initially be populated with the Job Media entries, but new Media can also be added via the Add button.
  - **Type** column: Select the Media Type from the range available. The selections are standard print output media types, except for the Unspecified option, which resolves to the printer default, or to that which matches the PageSize setting.
  - **Color** column: Select the Media color from the range available, if needed.

  The SetPageDevice table options are:
  - **Add**: Add new Media entries.
  - **Delete**: Delete the selected Media from the table.
  - / **Move Up / Down**: Move the selected Media up or down within the table.
  - **Import**: Import additional Media entries from existing templates.

    Choose from:
    - **Import from current template**: As the Media selections from the current template are automatically added to the table, selecting this option would either duplicate the existing entries (if they had not been deleted) or would

repopulate the table with those entries (had they been deleted).
If the existing entries had not been deleted, then the newly imported Media names will be identical to the entries already in the table, and thus their names will need to be changed.

- **Import from template file**: Import Media settings from some other Connect Template file.

- **Mapped PPD Entries**: This allows selection of a printer manufacturer's printer specific PostScript Printer Definition (PPD) file. These files contain what printing options are available for this specific printer.
  - **PPD Info** box: This is a read only box that contains the name of the selected PPD file, once such a selection is made.
  - **PPD file name**: Select a PostScript Printer Definition (PPD) file, using the Browse button.
    Once the PPD is loaded, you will be asked to save this new Printer Definition type. When the new Printer Definition is saved, the PPD Info box will be populated with information regarding it, and the following new dialog options become available:
    - **Media**: This box will initially be populated with any existing Media found in the current Template. You can add or remove Media, as you wish. The options available to the Media table are:
      - **Add**: Add new Media entries. This launches the Dynamic Tray Mapping Editor Dialog to allow you to set the Media properties.
      - **Delete**: Delete the selected Media from the table.
      - / **Move Up / Down**: Move the selected Media up or down within the table.
      - **Import tray settings**: Import additional Media entries from existing templates. The choices are:
        - **Import from current template**: As the Media selections from the current template are automatically added to the table, selecting this option would either duplicate the existing entries (if they had not been deleted) or would repopulate the table with those entries (had they been deleted).
          If the existing entries had not been deleted, then the newly

imported Media names will be identical to the entries already in the table, and thus their names will need to be changed.

- **Import from template file**: Import Media settings from some other Connect Template file.

Double clicking the left mouse button on a Media table entry will launch the Dynamic Tray Mapping Editor Dialog to allow you to modify that Media's properties.

- **Details**: This is a read only table that displays the Details of whatever Media has been selected in the Media box. Double clicking the left mouse button on a Media table entry will launch the Dynamic Tray Mapping Editor Dialog to allow you to modify that Media's properties.
- **Printer Options**:

  The **Printer Options** table displays :

  - **Name** column: This column contains the names of the Printer Options, as taken from the PPD file.
  - **Option** column: This column shows the selected Printer Option preference.
  - **Order** column: This column displays any Dependency Order criteria that applies to the selected Printer Option.

  The choices available to the Printer Options are as follows:

  - **Add**: This launches the Add Printer Options Dialog which allows you to add one or more Printer Options.
  - **Edit**: This allows you to edit an existing Rule Printer Option. It launches the Add Printer Options Dialog.
  - / **Move Up / Down**: Move the selected Printer Option(s) up or down within the table. This allows sorting of options for easier legibility, but manually moving options up and down is done without any check on Order Dependency.
  - **Delete**: Delete the selected Printer Option(s).
  - **Sort by Order dependency**: This sorts the Printer Options by the order of their dependencies.
    Not all Printer Options have dependencies, but for those that do, the

order is important.
For example, consider these following PPD file Printer Option extracts:

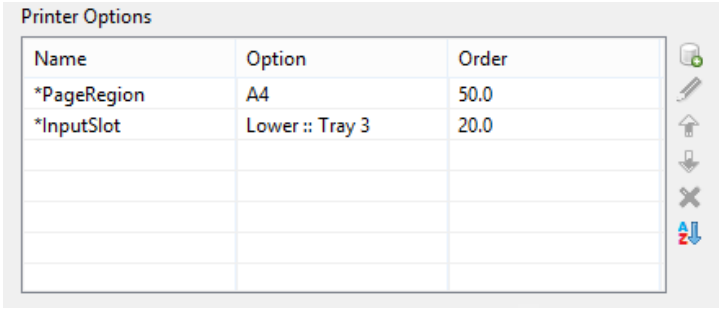
- ***PageRegion:***

```
*OpenUI *PageRegion: PickOne
*OrderDependency: 50 AnySetup *PageRegion
...
*CloseUI: *PageRegion
```

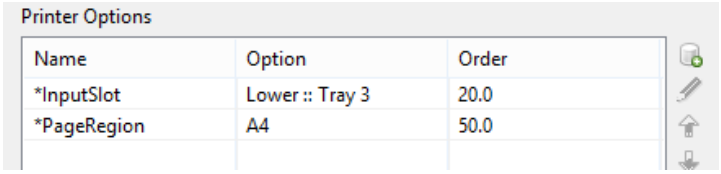
- ***InputSlot:***

```
*OpenUI *InputSlot: PickOne
*OrderDependency: 20 AnySetup *InputSlot
...
*CloseUI: *InputSlot
```

We can see that one has an Order Dependency of 50 and the other 20.
If *PageRegion* was selected prior to *InputSlot* in the Add Printer Options Dialog, then they would appear like this in the Print Wizard:

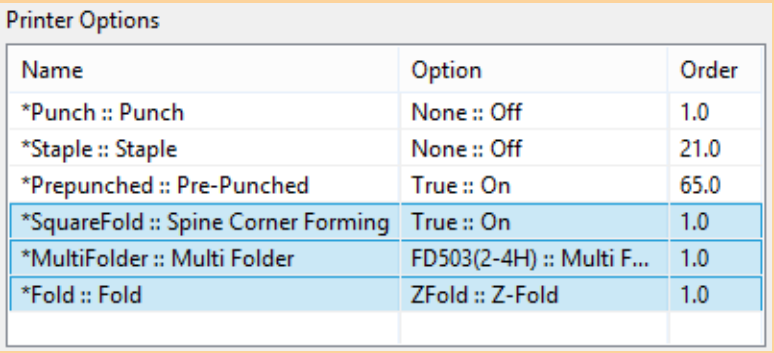


To sort them into the correct dependance order, use the **Sort by Order dependency** button. This will sort them into the proper dependency sequence. As seen below, where the lower dependency now appears first.



Caution: Nothing prevents the selection of conflicting and contradictory Print Options.

This can be seen in the following example, which shows multiple conflicting Folding Options have been chosen:



Printer Options

Name	Option	Order
*Punch :: Punch	None :: Off	1.0
*Staple :: Staple	None :: Off	21.0
*Prepunched :: Pre-Punched	True :: On	65.0
*SquareFold :: Spine Corner Forming	True :: On	1.0
*MultiFolder :: Multi Folder	FD503(2-4H) :: Multi F...	1.0
*Fold :: Fold	ZFold :: Z-Fold	1.0

The breadth and diversity of PPD files as well as individual printer capabilities mean that the Print Wizard cannot meaningfully validate the selections made, and thus does not attempt to.

So you must be careful when making selections. We recommend that you always test each individual selection made, to confirm that they are valid and that they do what you expected of them.

This could entail considerable trial and error, but should lead to less anguish in the long run.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings.

This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.

- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Spot Color Settings

> **Note:** This page is only available in the **Enterprise** edition of OL Connect.

The **Spot Color Settings** page is used to match Spot Colors contained within a job to the color settings available on the printer.
This option is only available for AFP and IPDS print outputs, and requires a Printer Definition File (see "Print preferences" on page 1057) that supports Spot Colors.

- **SpotColor Mapping** table:
  - **SpotColor Name** column: This contains the names of the Spot Colors used within the job.
    If the Spot Color were added to the job within Connect Designer, then those Spot Colors will be automatically included in the table.
  - **SpotColor Number** column: This lists which Spot Color available on the printer is to be associated with the named Spot Color.
- **Add**: Add a new association between a Spot Color in the job, and the Spot Colors options available on the printer.

  > **Note:** If the Spot Color was not added to the job within Connect Designer itself but are instead included within a external file (such as a PDF) added into the job, then care must be taken to ensure the Name added here is identical to the name reference used within that included file.
- **Delete**: Delete a Spot Color association.
- **Edit**: Edit an existing Spot Color association.

Additional Content

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Additional Content** page allows you to add content at the time of printing. There are four different types of Additional Content that can be added at print time: **Text**; **Images**; **Barcodes** and **OMR Marks**.
They can be used to add either static content or variable content.

The Additional Content option is particularly useful when you might need to drive custom processes on production machines using either Barcodes or OMR Marks, or if you need to add some last minute additions to the print job via text and/or images.
Additional content can also be used on banner pages.

Additional Text and Barcodes can contain `template` and `file` variables (see "File name variables" on page 1660), making it possible, for example, to display the output file name on banner pages.

> **Tip:** Having a barcode added at the time of printing is also a time saver in an automated process where only a barcode needs to be added to an existing PDF and a template is actually unnecessary. In this case, content creation can be skipped. (See "Print processes with OL Connect tasks" on page 306.)

Page breakdown

The Additional Content table displays any Additional Content that has been set.
Additional Content can be added, edited, removed, duplicated or moved within the table via the buttons to the right of the table.

The **Additional Content options** are:

- **Add**: Click to open some Additional Content.
  The choices are as follows:
  - "Additional Text Settings" on page 1411 entry.
  - "Additional Barcode Options" on page 1415 entry, from the selection presented.
  - "Additional Image Settings" on page 1413 entry.
  - "Additional OMR Mark Settings" on page 1449 entry.
- **Edit**: Click to edit the currently selected entry. This will launch the appropriate edit box, for the selected entry type.
  Double clicking on an entry in the table has the same effect as this button.

- **Delete**: Click to delete the currently selected entry or entries.
- **Duplicate**: Click to create a duplicated copy of the entry. This creates a new entry that is exactly the same as the original.
- / **Move Up / Move Down**: Click to move the selected entry (or entries) up or down within the table.

Additional Content Table: The table is split up into the following columns

- **Include in Output**: Indicates whether the entry is to be included in the output or not. The settings made in the Additional Content dialog box can be over-ridden here. This allows you to create a virtual "library" of Additional Content options in an Output Preset, from which you can then pick and choose from at time of printing.
- **Type**: Displays an icon showing what type the Additional Content entry is.
- **Description**: Displays the Description" that was entered in the Additional Content dialog.

  > **Note:** The entered text Description might stretch over a couple of lines, so it is possible that not all of the Description will be displayed here.

- **Left**: Displays the Additional Content Left Positional value.
- **Bottom**: Displays the Additional Content Bottom Positional value.
- **Content**: Displays the actual Additional Content that is to be inserted.
  For *Text*, this would be the actual text (or data fields) that are to be inserted.
  For *Image*, this would be the name of the image file.
  For *Barcode*, this would be the data fields used for creating the barcode.
  For *OMR*, this would be the number of marks set for this entry.
- **Condition**: Displays the condition that is used to determine whether this Additional Content instance is to be added to the output or not (see "How to set up conditions" on page 1453). Document information variables that you can use in conditions are listed in the topic: Variables Available in Output.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the

way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.

- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Additional Text Settings

The **Additional Text Settings** dialog displays the property of Text added in the "Additional Content" on page 1409 page.

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Text added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: Option relates to Imposition (also known as N-Up) printing. Select this box to have the Text printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Text, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Text, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.

- **Font** group:
  - **Font Name**: Use the drop-down to select which font type to apply to the Text. The drop-down displays all the fonts installed on the system.

    > **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

  - **Style**: Choose between "Regular", "*Italic*", "**Bold**" or "***Bold Italic***".
  - **Font Size**: Enter the font size in points (pt).
  - **Color**: Select what color the Text will be.
- **Text**: Enter the actual Text to appear on the page in the selected location. The Text can be spread over multiple lines, but no additional formatting can be added within this edit box. The entire Text will be printed using the formatting options selected in the **Font group**.
  Use the **Add** button ( ) to display a list of variable data that can be added to the

Text. This includes metadata fields added in the Metadata Options, as well as some document information fields (see "Print output variables" on page 1659 for information on those).

- **Condition**: Enter the condition which determines whether or not the Text will be added to the document at print time.
  Use the **Add** button ( ) for selection options.

  For details on how to create a conditional, see the "How to set up conditions" on page 1453 page.

  > **Note:** To add the content to the banner page, type the condition sheet.banner, or page.banner.

Additional Image Settings

The **Additional Image** dialog displays the properties of the image added in the "Additional Content" on page 1409 page.

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the image.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait	Landscape	Inverse Landscape	Inverse Portrait

- **Layer**: Whether this image will appear behind the text (the text will print over the image) or in front of the text (the text behind will be blanked out by the image, as transparent images are not supported).
    - **Ouput once per sheet**: Option relates to Imposition (also known as N-Up) printing. Select this box to have the Image printed once per sheet rather than once per document page.

      Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.
- **Left**: Enter the distance between the left margin of the page and the image, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the image, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.

- **Filename**: Use the browse button to launch a Browse box to select an image file. This is a mandatory field.

  Note: Transparent images are not supported.
- **Preview** group: This displays a preview of the selected Image.
- **Scaling** group:
  Scaling the image expands the image but keeps the aspect ratio. The amount of scale and specific limitations can be applied used a combination of the following options:
    - **Max Width**: Enter the absolute maximum width the image can be scaled to, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.

- **Max Height**: Enter the absolute maximum height the image can be scaled to, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Scale**: What scale to apply to the image. The maximum scale is 10.0 to 1. Decimal values are allowed for this field.

- **Condition**: Enter the condition which determines whether or not the image will be added to the document at print time. Use the Add button for selection options.

  For details on how to create a conditional, see "How to set up conditions" on page 1453 page.

Additional Barcode Options

When adding Barcodes in the "Additional Content" on page 1409 page you can select from a series of predetermined Barcode types.
To create dynamic barcodes, "Meta Data options" on page 1374 must first have been set. Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

The options for each of these types are described on the following pages:

- "Australia Post Settings" on page 1302
- "Codabar Settings" on page 1417
- "Settings" on page 1422
- "Code 128 Settings" on page 1419
- "Data Matrix settings" on page 1425
- "EAN-8 Settings" on page 1433
- "EAN-13 Settings" on page 1431
- "GS1-128 Settings" on page 1428
- "Interleaved 2 of 5 Settings" on page 1435
- "Japan Post Settings" on page 1324
- "KIX Code (Dutch Post) Settings" on page 1326
- "PDF417 Settings" on page 1438
- "QR Code Settings" on page 1440
- "Royal Mail 2D Mailmark Settings" on page 1335
- "Royal Mail 4 State (CBC) Settings" on page 1337
- "Royal Mail 4 State Mailmark C Settings" on page 1339

- "Royal Mail 4 State Mailmark L Settings" on page 1342
- "UPC-A Settings" on page 1444
- "UPC-E Settings" on page 1446
- "US Postal Service IMb Settings" on page 1349
- "US Postal Service IMpb SettingsUS Postal Service Intelligent Mail Package Barcodes (IMpb) are used by the United States postal service for parcel deliveries. They contain parcel destination information along with other information about the delivery.To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1). Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.Application Identifiers in the barcode value need to be surrounded by round brackets, otherwise the barcode will not be generated correctly. For example, this value contains both a Postal Code Application Identifier (420) and a Channel Application Identifier 93: 420123459302234567890123456788. It should be written as follows: (420)12345 (93)02234567890123456788.More information about Application Identifiers can be found in the barcode specification here: https://postalpro.usps.com/shipping/impb/BarcodePackageIMSpec (8.0 Appendix D).Use the following options to configure the output Barcode settings:General group:Include in output checkbox: This determines whether of not this Additional Content entry should be included in the output or not. It is checked by default for all new entries.This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.Description edit box: An edit box for adding an optional Description to the Additional Content entry.Position group:Orientation: Use the drop-down to select the orientation of the Barcode added to the page.The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document: PortraitLandscapeInverse LandscapeInverse PortraitOutput once per sheet: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.Left: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.Bottom: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.USPS IMPB Properties group:Barcode height: The height of the bars in the barcode, in inches (in). Value must be between 0.075 and 1.0 inches.Module width: The thickness of thin bars in the barcode, in inches (in).

Thicker bars are a multiple of this value. Value must be between 0.013 and 0.021 inches.Text to barcode clearance: The space between the barcode bars and the top and bottom text, in inches (in). Value must be between 0.125 and 0.25 inches.Text height: The height of the text, in inches (in). Value must be between 0.09 and 0.125 inches.Line to text clearance: The space between the guard lines and text, in inches (in). Value must be between 0.031 and 0.04 inches.Guard bar height: The thickness of the guard lines, in inches (in). Value must be between 0.031 and 0.063 inches.Process Tilde:This selection is not applicable to this barcode type.Text: Enter the text used to generate the Barcode. Add button: Click to display a list of variable data that could be used for generating the Barcode. This includes metadata fields which must previously have been added in the Metadata options, likely at the Document Tags level (see Metadata Options), as well as some information fields.Condition: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1. Add button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition." on page 1

Codabar Settings

Codabar barcodes support the following data: 0-9 - $: / . + plus the optional specification of start/stop characters.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:

- **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

  The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Codabar Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Bar width ratio**: Set the Barcode bar width.
  - **Default start symbol**: Use the drop-down to select the optional Barcode start character, which defines the encoding mode.
  - **Default stop symbol**: Use the drop-down to select the Barcode stop character, which defines the encoding mode.

- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.
- **Display start/stop symbols** checkbox: Adds the stop/start symbols to the Barcode text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Code 128 Settings

Code 128 is a high-density barcode, used for alphanumeric or numeric-only barcodes. It supports all 128 ASCII characters.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

  - **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

    - **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Code 128 Properties** group:
- **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.
    - **Font size**: Enter a font size for the human readable text.
- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.
- **Text**: Enter the text used to generate the Barcode.
    - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
      This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
    - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Code 39 Settings

Code 39 is a discrete, self-checking barcode that is also known as "Alpha39", "Code 3 of 9" (often abbreviated to "3 of 9"), "Code 3/9", "Type 39", "USS Code 39" and "USD-3".

Code 39 data should contain no more than 20 digits from within the following range: Numeric digits: (0-9), upper-case letters (A-Z), seven special characters (- . space $ / + %) and the start/stop asterisk (*) character.

If the Extended character set is chosen, then lower-case letters (a-z) and other special ASCII characters can also be included.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Settings

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait	Landscape	Inverse Landscape	Inverse Portrait

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Code 39 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Use extended character set**: Check to use the Code 39 Extended character set. This extends the range of supported data to include the full ASCII character set. This adds support for lower case letters (a-z) and the full range of ASCII punctuation and special characters.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger. The smallest Module Width is 0.19mm (high density).
  - **Bar width ratio**: Set the Barcode bar width.
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:

- **Ignore**: Ignore checksum calculations.
- **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
- **Check**: Verify the Barcode has a valid checksum.
- **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  > **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Data Matrix settings

A Data Matrix barcode is a high-density, two-dimensional (2D) matrix barcode which supports encoded text, numbers, files and digital data.

Note: In order to create a **GS1** compliant Data Matrix barcode, the barcode data must meet the following requirements:

- The barcode data must start with a leading FNC1 character. Either ~1 or ~d232.
- The GS1 Application Identifiers (AI) must be used for all data. The function code ~1 must be used as field separator for variable length AI elements.
- Only ASCII characters should be used.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Data Matrix Properties** group:
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Encoding**: The data represented in the symbol can be compressed using one of the following algorithms:
    - **Auto**: Automatically detect the data content and encodes using the most appropriate method. This is the default option.
    - **ASCII**: is used to encode data that mainly contains ASCII alphanumeric characters (ASCII 0-127). Use where Barcode size is a concern and where the data is alphanumeric.
    - **Base 256**: used to encode 8-bit values.
    - **C40**: used for data that mainly consists UPPER-case alphabetic letters and numbers.

- **Text**: used for data that mainly consists of lower-case alphabetic letters and numbers.
- **None**: Does not use any encoding.

> **Note:** For general information on encoding, see https://en.wikipedia.org/wiki/Data_Matrix#Encoding.
>
> Please check with the manufacture of the barcode reader if any specific encoding is required

- **Format**: select the Barcode size format from the drop-down list .

- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

  > **Note:** The following tilde codes are supported:
  >
  > - ~1 = FNC1 character (for GS1 DataMatrix barcodes)
  > - ~2 = Structure Append
  > - ~3 = Reader programming
  > - ~5 = Macro5
  > - ~6 = Macro6
  > - ~7 = ECI expressions

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.

- **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

GS1-128 Settings

GS1-128 is also known as "EAN 128", "EAN/UCC 128" and "UCC 128". This barcode type not only encodes data, but also provides a mechanism for defining the meaning (or format) of that data. It supports alphanumeric data and some predefined Function Codes. See the Wikipedia GS1-128 entry for more information.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait **Landscape** **Inverse Landscape** **Inverse Portrait**

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **GS1-128 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Check Digit marker**: This character is used as a placeholder for the check digit, which we be calculated at runtime. The character must be expressed in Hex.
  - **Group separator**: This character is used to define group separation points. The character must be expressed in Hex.
  - **Template**: Specify an optional Barcode "template".
    Examples:
    - `n13` defines a numeric field with exactly 13 digits.
    - `n13+cd` defines a numeric field with exactly 13 digits plus a check digit.
    - `an1-9` defines an alpha-numeric field with 1 to 9 characters.

    Elements can be combined using the '+' symbol.

- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

  With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

EAN-13 Settings

EAN-13 barcodes are composed entirely of numerical data. The first 12 digits representing country/economic area, manufacturer and product codes + 1 following checksum digit.

> **Note:** To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
> Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **EAN 13 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger. The EAN-13 barcode employs a module width between 0.27mm and 0.66mm.
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
    - **Ignore**: Ignore checksum calculations.
    - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
    - **Check**: Verify the Barcode has a valid checksum.
    - **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

> **Note:** Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

    - **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
    - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
      This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
    - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

EAN-8 Settings

An EAN-8 barcodes are composed entirely of numerical data. It is comprised of 7 data digits containing the country/economic area code and an item reference code, with 1 following checksum digit.

Use the following options to configure the output Barcode settings:

- **General** group:
    - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
      It is checked by default for all new entries.
      This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
    - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:

- **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

  The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **EAN-8 Properties**:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger. The EAN-8 barcode employs a module width between 0.27mm and 0.66mm.
  -
    - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:

- **Ignore**: Ignore checksum calculations.
- **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
- **Check**: Verify the Barcode has a valid checksum.
- **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Interleaved 2 of 5 Settings

Interleaved 2 of 5 barcodes are also known as "ITF" and "2/5 Interleaved". It is a numeric only barcode whose data must contain an even number of digits, as the barcode uses sequences of two digits interleaved with each other to create a single symbol. If the numeric data

contains an odd number of digits, then a leading zero must be added to the beginning of the data.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **Interleaved 2 of 5 Properties** group:
  - **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Bar width ratio**: Set the Barcode bar width.
    - **Print human readable text:** Check to add a textual version of the Barcode data.
    - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
    - **Font name**: Use the drop-down to select the font with which to display the human readable text.

      Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.
    - **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.

- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.

- **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

PDF417 Settings

PDF417 is a two-dimensional, multi-row Barcode. It is used for encoding large amounts of data, with hundreds or even thousands of characters. It encodes alphabetic text, numbers, binary files and actual data bytes.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait **Landscape** **Inverse Landscape** **Inverse Portrait**

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

- **PDF417 Properties** group:
  - **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Row height**: Defines the height of the bars for a single row, measured in pixels, points or metric.
  - **Width to height ratio**: Select the ratio of column width to row height.
  - **Mode**: Use the drop-down to set the compaction mode.
    - **Binary**: allows any byte value to be encoded.
    - **Text**: allows all printable ASCII characters to be end coded (ASCII values 32 to 126 and some additional control characters).
    - **Numeric**: more efficient mode for encoding numeric data.
    - **Auto**: Automatically detect the data content and encodes using the most appropriate method. This is the default option.

- **Error Correction Level**: Enter the error correction level for the built-in error correction method based on Reed-Solomon algorithms. The error correction level is adjustable between level 0 (just error detection, without correction) and level 8 (maximum error correction). Recommended error correction levels are between level 2 and 5, but the optimal value depends on the amount of data, printing quality of the PDF417 symbol and decoding capabilities.
- **Rows**: A PDF417 bar code can have anywhere from 3 to 90 rows.
- **Columns**: The number of data columns can vary from 1 to 30.

- ## Tilde processing

  Check this option to process tilde (~) characters in the data as special characters. (See the Java4less Barcodes Guide to learn what the tilde character can be used for.) The tilde is expected to be followed by the 'd' character and 3 digits representing an ASCII character: **~dNNN** . For example, ~d013 represents a carriage return.
  Note that with this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

QR Code Settings

QR Code (Quick Response Code) is a 2D Barcode format that supports alphanumeric, numeric, byte/binary, and Kanji (Japanese-Chinese character) data.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).

Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

QRCode Properties group:

- **Size by**: Select size from the two options available:
  - **By area**: Connect will try to size the Barcode to fit the specified area by dynamically changing the module width to the **Size** selection. The lower module width limit is governed by the **Minimum module width** selection.
    Enter the sizes in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
  - **By module width**: Connect will try to size the Barcode to the module width of the characters. Large Barcode values will result in larger Barcode and vice versa.
    Enter the **Module width**in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Encoding**: Define the encoding of the Barcode:
  - **Auto**: Automatically detect the data content and encodes using the most appropriate method. This is the default option.
  - **Numeric**: 7089 numerical characters.
  - **Alphanumeric**: 4296 alphanumerical characters.
  - **Byte**: 2953 characters.
  - **Kanji**: 1817 Japanese/Chinese characters.
- **Version**: Select the preferred QR code version (which sets the data length field) from the 40 available.

Note: The Encoding and Version fields work together to determine how many characters are encoded within a *length field*. The following table shows the number of bits in a length field, based upon the selections made:

Encoding	Ver. 1-9	Ver. 10-23	Ver. 27-40
Numeric	10	12	14
Alphanumeric	9	11	13
Byte	8	16	16

Kanji	8	10	12

- **Error Correction Level**: Part of the robustness of QR codes is their ability to sustain “damage” and continue to function even when a part of the QR code image is obscured, defaced or removed. A higher correction level duplicates data within the QR Code to allow for damaged areas. The higher the Error Correction Level, the larger the Barcode will be. The choices are (in order from lowest to highest): **L**ow, **M**edium, **Q**uartile and **H**igh.
- **Use ECI for encoding messages as bytes**: Selecting Extended Channel Interpretations (ECI) allows encoding multiple character sets (e.g. Arabic, Cyrillic, Greek, Hebrew) and other data interpretations, into one QR Code symbol.
- **Multi-part QR Code (structured append)**: Select to append a QR Code symbol in a structured format.
    - **Part**: indicates the position of the QR Code symbol within the group of Structured Append symbols.
    - **of**: indicates how many Structured Append symbols exist.

    Note: The Structured Append symbols Part number can never exceed the sum total of Structured Append symbols available (the "of" value). Thus selecting a Part number beyond the existing sum total will increase the sum total to the same value.
- **Use FNC1**: Check to enable Application Identifiers. These are often used to encode links to websites, or to encode production/batch details.
    - **Position**: Select between the two methods for encoding FNC1 characters within QR Codes:
        - **First** Position - uses the GS1 QR Code standard.
        - **Second** Position - uses the AIM QR Code standard. If this option is chosen then the appropriate Application Indicator will also need to be set.
            - **Application ID**: Enter the appropriate QR-Code Application Indicator in accordance with the specific industry or application specifications (as provided by AIM International).
- **Process Tilde**:
  Check this option to process tilde characters in the data as special characters.

With this option checked, any tilde that needs to be included in the output must be escaped by adding another tilde: ~~.

- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

UPC-A Settings

The Universal Product Code (UPC-A) Barcode is widely used for tracking trade items in stores and at the point-of-sale. It consists of 12 numerical digits which are uniquely assigned to each trade item.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).
Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:

- **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

  The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

- **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

UPC A Properties group:

- **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
  - **Ignore**: Ignore checksum calculations.
  - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.

- **Check**: Verify the Barcode has a valid checksum.
- **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
- **Print human readable text:** Check to add a textual version of the Barcode data.
- **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
- **Font name**: Use the drop-down to select the font with which to display the human readable text.

  Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.

- **Font size**: Enter a font size for the human readable text.

- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.
- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

UPC-E Settings

The Universal Product Code (UPC-E) Barcode is widely used for tracking trade items in stores and at the point-of-sale. It consists of 6 numerical digits which are uniquely assigned to each trade item.

Note: To create dynamic barcodes, Metadata options must first have been set (see "Meta Data options" on page 1374).

Metadata fields are required to create the association between the dynamic data used in the print run and the barcode.

Use the following options to configure the output Barcode settings:

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the Barcode added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

  - **Output once per sheet**: This option relates to Imposition printing (see Imposition), also known as N-Up printing. Select this box to have the Barcode printed once per sheet rather than once per document page.

    Note: If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the Barcode, in either metric (cm/mm), inch (in) or point (pt) values.

UPC A Properties group:

- **Height**: Enter the Barcode height in either metric (cm/mm), inch (in) or point (pt) values.
- **Module Width**: Specifies the width of the narrow bars. Changing this value to a higher value will generally make the Barcode bigger.
  - **Checksum**: Use the drop-down to select how to deal with the Barcode checksum:
  - **Ignore**: Ignore checksum calculations.
  - **Auto**: Add a checksum character to the Barcode if the initial value does not validate. This is the default value.
  - **Check**: Verify the Barcode has a valid checksum.
  - **Add**: Calculate and add a checksum character to Barcode, regardless of current value.
  - **Print human readable text:** Check to add a textual version of the Barcode data.
  - **Placement**: Use the drop-down to select whether to place the human readable text above or below the Barcode.
  - **Font name**: Use the drop-down to select the font with which to display the human readable text.

    Note: Vertically-oriented fonts (fonts whose typeface name begin with the at (@) character) are not supported in Connect.
  - **Font size**: Enter a font size for the human readable text.
- **Process Tilde**:
  This selection is not applicable to this barcode type.
- **Text**: Enter the text used to generate the Barcode.
  - **Add** button: Click to display a list of variable data that could be used for generating the Barcode.
    This includes metadata fields which must previously have been added in the

Metadata options, likely at the *Document Tags* level (see Metadata Options), as well as some information fields.

- **Condition**: Enter the condition which determines whether or not the Barcode will be added to the document at print time. For details on how to create a conditional, see "How to set up conditions" on page 1453.
  - **Add** button: Click to display a list of metadata fields, information fields to add, or common expressions to the condition.

Additional OMR Mark Settings

The **Add OMR** dialog displays the properties of an OMR Mark that was added in the "Additional Content" on page 1409 page.

These OMR marks differ from High Capacity Feeder (HCF) generated inserter marks. Those marks are specific to the inserter machine they were created for, whereas these additional OMR marks are completely independent and customizable. These custom OMR marks can be used to cater for inserter machines not currently supported by a HCF, or they can be used for any non-inserter related post-processing driven by OMR marks.

- **General** group:
  - **Include in output** checkbox: This determines whether of not this Additional Content entry should be included in the output or not.
    It is checked by default for all new entries.
    This option allow for "libraries" of Additional Content to be created in Presets, libraries from which you can pick and choose what entries you wish to have included, at time of printing.
  - **Description** edit box: An edit box for adding an optional Description to the Additional Content entry.
- **Position** group:
  - **Orientation**: Use the drop-down to select the orientation of the OMR Mark added to the page.

    The orientations would look like this (using text and a QR barcode as the examples), in a Portrait document:

Portrait	Landscape	Inverse Landscape	Inverse Portrait

- **Page Side**: Select whether the OMR Mark will print on the front or back of page.
- **Output once per sheet**: Option relates to Imposition (also known as N-Up) printing. Select this box to have the OMR Mark printed once per sheet rather than once per document page.

  > **Note:** If Imposition options such as auto-positioning and scaling were selected, these options won't apply to the Additional Content added to the physical N-Up sheet.

- **Left**: Enter the distance between the left margin of the page and the OMR Mark, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.
- **Bottom**: Enter the distance between the bottom margin of the page and the OMR Mark, in either metric (cm/mm), inch (in), pixel (px) or point (pt) values.

- **Options Tab**:
  - **Collation Level**: Choices are:
    - **Document**: Treats each document as a group and the group and match marks will be set based upon the start and end of a document.
    - **Document Set**: Treats each document set as a group and the group and match marks will be set based upon the start and end of a document set.
  - **Draw Hot Spots**: This adds a red rectangle around the location of each individual mark in the output, allowing easier checking of the OMR mark logic.
  - **Line Options** group:
    - **Line Thickness**: Sets the thickness of each OMR mark line.
    - **Line Length**: Sets the lenght of each OMR mark line.

- **Line Spacing**: Determines how the spacing between each OMR mark line will be set. The associated control beneath the combination box will be enabled, based upon this selection.
    - **Line Per Inch**: If **Line Spacing** is set to *Lines Per Inch* this option will be enabled. It defines how many lines will print per inch.
    - **Gap Distance**: If **Line Spacing** is set to *Gap Distance* this option will be enabled. It defines the size of the gap between lines; i.e. the distance from the bottom of one OMR mark line to the top of the next.
    - **Line Distance**: If **Line Spacing** is set to *Line Distance* this option will be enabled. It defines the distance from the top of one line to the top of the next.

- **Sequence Number Range** group: Allows selection of Start and Stop points for the wrapping page sequence number in a group.
  For example, a range of 2-10 would cause the sequence numbers to iterate as follows: 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4 ...

  > **Note:** The Sequence Number iterates per page within a group and is used to identify missing pages in a group.

    - **Start**: The starting point for the range
    - **Stop**: The end point of the range
    - **Start number**: The number to start from (from within the selected range).
- **Match Number Range** group: Allows selection of Start and Stop points for the wrapping match number for a group.
  For example, a range of 1-6, with a Start number of 2 would cause the matched numbers to be as follows: 2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 2, ...

  > **Note:** The Match Number iterates per group and is used to identify missing groups

    - **Start**: Start number
    - **Stop**: Stop number
- **Condition**: Enter the condition which determines whether or not the OMR Mark will be added to the document at print time.
  For details on how to create a conditional, see the "How to set up conditions" on

page 1453 page.

- **OMR Marks Tab**:
  - **#**: OMR Mark number (display only).
  - **Type**: Type of OMR Mark (display only).
  - **Value**: OMR Mark Value. These can be selected and altered for Sequence, Match and Parity marks, as described below.
  - **Add**: Add an OMR Mark entry to the table.
    Choices are between:
    - **On**: This represents a mark that is always printed.
    - **Off**: This represents a mark that is never printed; i.e. it pads the marks out with an empty position.
    - **Group Start**: This represents a mark that is printed on the first page of a group.
    - **Group End**: This represents a mark that is printed on the last page of a group.

      Note: In a single page group both Group Start and End marks will print if defined since the page is both the start and end of the group.
    - **Sequence**: This represents a mark that is printed when the specified bit is set in the sequence number of the page.
      For example, if the bit for the mark is set to 2 and the sequence number for the page is 5 then it will not print since the value 5 consists of the bits 1 and 4.
      Use the drop down box to select the entry.
    - **Match**: This represents a mark that is printed when the specified bit is set in the match number of the group.
      For example, if the bit for the mark is set to 2 and the match number for the group is 3 then it will print since 3 consists of the bits 1 and 2.
      Use the drop down box to select the entry.

      Note: The match number is the same for all pages in a group.
    - **Parity**: This mark prints in order to maintain the parity of the number of lines printed on the page. If set to *Even* then it will print if the total count of

the other printed marks in the printed is odd.
For example, by printing the parity mark it will create an even number of marks on the page. And vice versa with *Odd* parity - the parity mark will print if the total number of other printed marks on the page is even in order to keep the overall count odd.
Use the drop down box to select the entry.

- **Conditional**: Enter the condition which determines whether or not this OMR Mark will be added to the document at print time.
  For details on how to create a conditional, see the "How to set up conditions" below page

- ❌ **Delete**: Delete an entry from the table
- ⬆️ **Move up**: Move a entry up the table
- ⬇️ **Move down**: Move a entry down the table
- ✏️ **Edit**: Edit a **Conditional** entry within the table.

> **Tip:** You can also double click a **Conditional** entry within the table to edit it.

How to set up conditions

- **Condition**: Enter the condition which determines whether or not this element will be added to the document at print time.

  Use JavaScript conditional expressions to construct your condition.
  You may use JavaScript logical operators (`==`, `!=`, `<`, `>`, `&&`, `||`, etc.), literal values (`"text'`, `3.14`, etc.), mathematical operators (`/`, `*`, `+`, `-`, etc.) and data fields.

  - **Add**: Click to add a field to the to the conditional expression.
    Select from a list of metadata fields (as added in Metadata Options), document information fields (see Variables Available in Output), or common expressions.

    For example, selecting the **Expressions > First Page** would add `"page.nr == 1"` to the conditional entry.
    Example: `if (page.nr == 1) { ... }`

Examples

> **Example:** `"page.nr == 1"`

This condition means the additional content can only be added to the first page.

Example: `"page.banner"`

With this condition, the additional content will only be added to banner pages.

Example: page.banner && page.front

This condition determines that the page must be both a banner page and a front page. In case of duplex banner pages, this prevents content from appearing at the back of the banner page.

Example: `page.content`

This condition prevents additional content from appearing on banner pages. It is true for all pages that are part of the content, and false for banner pages.

Booklet Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Booklet Options** page defines how to generate booklets in the output. It is used in conjunction with Imposition settings, which will appear after the Booklet entries have been made.

This page includes a handy illustration that displays how the final binding would look, based upon the current selections.

Options:

- **Configuration**: Use the drop-down to select the type of binding to use:
  - **Saddle Binding**: This binding places all the pages in a stack, binds the middle and folds the stack as one.
  - **Perfect Binding**: This binding type is often used for books. Pages are folded in the middle and then set side by side. The pages are then bound along the folded "spine".
  - **1 up Perfect Binding**: This binding does not contain any folding. The pages are lined up side by side and bound along one edge.
- **Booklet Binding Edge**: Use the drop-down to select the side on which to bind the booklet.

Optional **Cover Page** selections are available to Saddle Binding only.

- **Cover Page** checkbox: Check to enable cover pages to be created with the options below:
  - **Media** selections:
    - **Cover Media Size**: Use the drop-down to select the media size for the cover page, or use a Custom size and select **Width** and **Height** values.
  - **Front Cover** selections:
    - **Blank**: Select to add no data to the front cover.
    - **First page on outside and second page on inside**: Select to use the first 2 pages as the inside and outside of the front cover.
  - **Back Cover** selections:
    - **Blank**: Select to add no data to the back cover.
    - **Last two pages on inside and outside**: Select to use the final 2 pages as the inside and outside of the back cover.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Imposition options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

Imposition refers to the printing of multiple pages on a single sheet. This is also known as N-Up printing.

The options on the **Imposition Options** page allow for the setting of imposition repetition, order, margins and markings.

As imposition selections are very specific and can be quite confusing, we have provided a handy dynamic diagram that displays a representation of the current imposition selections in real time. Whenever selections are changed, this display changes to reflect the selections made.

You can even select how many pages are to be represented, whilst page numbers appear on the pages in the diagram and change dynamically, like everything else.

It is important to note that if Impositioning is used then the only meta data that will be available to the job thereafter will be the meta data at Job Segment level.

When using Impositioning everything within the Segment is Impositioned, so all meta data below the Job Segment level is lost. If *Booklet Binding* were selected, some of the imposition settings will be determined by the options made within the "Booklet Options" on page 1454 Page and those imposition settings will thus be disabled on this page.

> **Note:** During Impositioning, all hyperlinks - both external and internal - are removed from the document (see "Hyperlink and mailto link" on page 850).

The Imposition selections that can be made are as follows:

- **Sample Page** group: Use to choose a paper size for the logical page(s) which would be printed upon the actual physical Imposition sheet. This helps in decision making regarding what paper sizes and orientations to use, with the Sample Page shown in blue on the white Imposition Sheet in the output preview (and validation) diagram.

  > **Note:** The logical page size can be set independently of any template open in the Designer, allowing for output presets to be created independently from the active template.

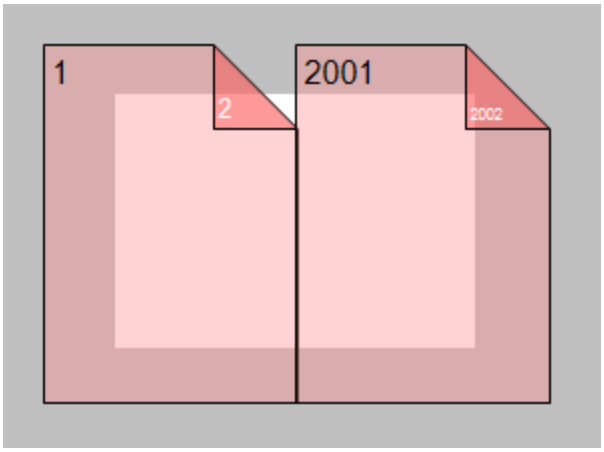
If a template is currently loaded in Connect Designer, the size and portrait attribute of the first section of that template are used as defaults for the Sample Page.

As a general rule, it makes little sense to modify the Sample Page size within the **Print Wizard** (as the template media size will already have been selected), but it does make sense when creating Print Output Presets for other templates.

If no template is loaded or if the template does not have a Print section, then the Sample Page defaults to A4 portrait.

  - **Size**: Use the drop-down to select the media size of the logical template page.
  - **Orientation**: Select orientation (aspect ratio) of the logical template page (*Landscape* or *Portrait*).
  - The **Width** and **Height** values are real-only and appear for display and design purposes only.
- **Imposition Sheet** group: Use to select the physical media size that the logical pages will be printed upon. This appears as the white background in the output preview (and validation) diagram.
  - **Size**: Use the drop-down to select the size of the physical media the output is to be printed upon. The default is SRA3 (oversized A3).
  - **Orientation**: Select orientation (aspect ratio) of physical media sheet (*Landscape* or *Portrait*).
  - The **Width** and **Height** values are real-only and appear for display and design purposes only.
- Output diagram and validation. Use the settings here to control how the output diagram is to look and behave. It contains the following options:
  - **Display size** entry: Chose between a pre-set viewing percentage, or manually enter whatever percentage you would like, or select *Fit Page* or *Fit Width* to have the diagram display the page(s) within the diagram window dimensions.
  - Use the zoom in/out buttons / to increase or reduce the viewing percentage by 25% increments.

- **Sample size** entry: Set the sample output size here. The output diagram will change dynamically to reflect the selection made here, if applicable.
  The maximum Sample size is 10,000 for booklet imposition and 10,000,000 for standard impositions.
- Output diagram: A real-time display that reflects the selections made in the Imposition Options page.
  The blue page(s) are the individual logical page(s), upon a white background that represents the physical Imposition Sheet.
  If the settings are invalid, the display will appear as red, such as in the following diagram:



  In this example the error is that the logical pages are too large to fit upon the Imposition Sheet that has been selected.

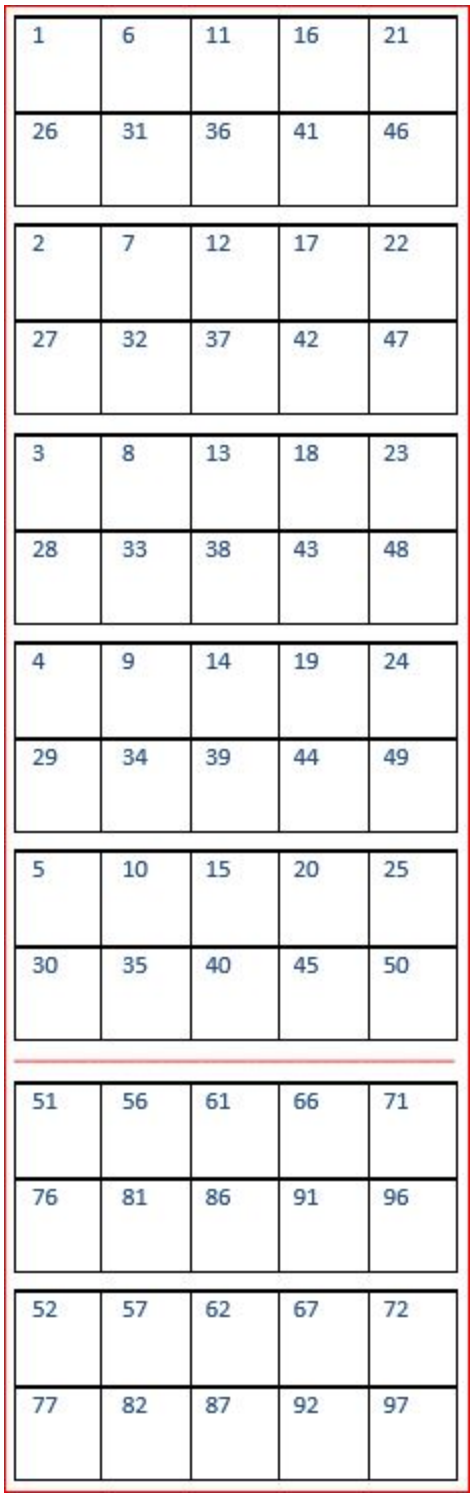
- **Position** group: These settings apply to the positioning of the logical page(s) within the Imposition Sheet.
  - **Method** selection: Set the position of the logical page(s) upon the Imposition Sheet. The choices are as follows:
    - *Center*: Have the logical page(s) centered within the Imposition Sheet. This is the default method.
    - *Scale to fit*: Scales the logical page(s) so that they fit the Imposition Sheet exactly.
    - *Offset*: Have the logical page(s) offset within the Imposition Sheet. The offset values can be set in the Left and Top entry boxes, which only become available for this positional method.

- **Left** and **Top** values: These entry boxes are only enabled and their values applied when the Offset method has been chosen. They are used to set the offset of the logical page(s) upon the Imposition sheet.
- **Rotate** selection: Select to rotate the logical page(s) within the Imposition Sheet.

- **Layout** group: Select how the logical page(s) are to be laid out on the Imposition Sheet.
  - **Horizontal** selection: Set how many logical pages are to be placed on one horizontal line within the Imposition Sheet.

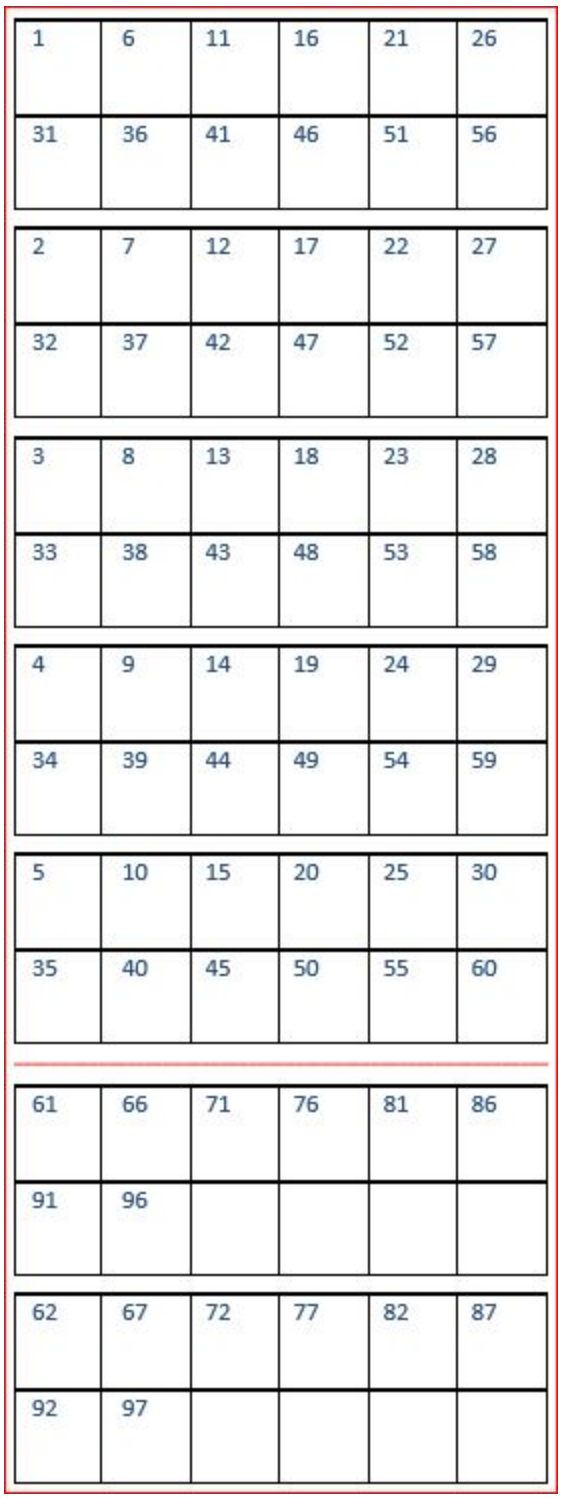
    If the logic page was set to **Rotate** in the Position selections, then the Horizontal selection might appear as vertical, and the Vertical selection might appear as horizontal, on the real time output diagram.
  - **Vertical** selection: Set how many logical pages are to be placed on one vertical line within the Imposition Sheet.
  - **Horizontal Gap** selection: Set the amount of blank space to add between each logical page on this axis.
  - **Vertical Gap** selection: Set the amount of blank space to add between each logical page on this axis.
  - **Stack Depth** selection: Stack Depth defines the number of Imposition Sheets that will be printed before drill sorting resets.
    Stack Depth works in conjunction with the Horizontal and Vertical page, Gap and Page Order selections in order to determine the printing order. This is best illustrated by way of example.

    The following "Cut and stack" example has a job with 100 single page documents and repetition settings of 5 Horizontal and 2 Vertical, plus a Stack Depth of 5. Each table represents a single Imposition Sheet, and the red line between the tables represents the Stack Depth:



Each set of 5 Imposition Sheets is a discrete drill sorted nUp block that can be guillotined and stacked. The first 5 sheets encompass records 1-50, whilst the next 5 sheets encompass records 51-100.

The following example is the same job but with a Horizontal repetition setting of 6:



1	6	11	16	21	26
31	36	41	46	51	56
2	7	12	17	22	27
32	37	42	47	52	57
3	8	13	18	23	28
33	38	43	48	53	58
4	9	14	19	24	29
34	39	44	49	54	59
5	10	15	20	25	30
35	40	45	50	55	60
61	66	71	76	81	86
91	96				
62	67	72	77	82	87
92	97				

As you can see, the outputs are *very* different.

> **Note:** If the Stack Depth is greater than the number of sheets in the entire job then you will get a single drill sorted block that can be guillotined and stacked to give the complete job in record order.

- **Sides** selection: Select how many sides of the Imposition Sheet are to be printed upon. Choice is between *Simplex* or *Duplex*.
- **Method** selection: In 2018.2 we introduced a new imposition page order for cut & stack impositioning that works with continuous feed devices: *Stack by Column*. This option is intended for impositions with more than 1 row. It places the next page in the next row of the same column on the current sheet, instead of in the same row and column of the next sheet like the pre-existing *Cut and Stack* does.
  The choices are between:
  - *Cut and Stack*: Any preset created in a version of Connect prior to Connect 2018.2 will default to this Layout Method. It allows total control of subsequent **Page Order** choices.
  - *Stack by Column*: For use with continuous feed printers. This selection limits subsequent **Page Order** choices to column based options only.
- **Page Order**: Select in which direction to go when adding sections to the output:
  - **Left to right, top to bottom**
  - **Right to left, top to bottom**
  - **Top to bottom, left to right**
  - **Top to bottom, right to left**
- **Reverse** selection: The reverse selections have been extended in Connect 2018.2, with the addition of the *Inverse page order* option. The options now available are:

- *Off*. Output is not reversed.
- *Stack upside down*: Essentially turns the stack upside down by reversing the output imposition page sides. The last sheet becomes the first, and in the case of duplex imposition, the back sides become the front sides.
  For presets created in a version prior to Connect 2018.2, a "Reverse Pages" selection will apply this *Stack upside down* setting.
- *Inverse page order*: This impositions the output in the ***exact opposite order*** from what has been selected.
  For example, if the selected order was **Top to bottom, left to right** (down the stack, left to right, top to bottom), then selecting **Inverse page order** would result in output that was up the stack, right to left, bottom to top.
  In the case of duplex impositioning, the front and back sides don't switch.

- **CropMarks** group: Select to add Crop Marks (also known as cut marks or trim marks) to the printed output.
  - **Type**: Use the drop-down to select whether to add Crop Marks to the page or not, and what type of Crop Marks to use. The Crop Mark types available are *Standard* or *Japanese*.
  - **Page side**: What side(s) of the page to put the Crop Marks. The choices are *Front*, *Back* or *Both*.
  - **Width**: Select the width of the crop mark lines.
  - **Offset**: How much separation (if any) to leave between the vertical and horizontal corner markings.
  - **Length**: Select the Length of the crop mark lines.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.

- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Inserter options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Inserter Options** page allows the selection of a High Capacity Feeder (HCF) model. These machines are also commonly referred to as Inserters or Folder-Inserters.

The options available on this page are dependent upon the model selected.

The options selected on this screen influence the position of the markings set on the next page: "Mark Position Options" on page 1466.

- **Model**: Use the drop-down to select from any previously loaded Inserter model, or use the Browse button to select a HCF file to load a new Inserter model.
  An image representing the chosen folder-inserter is displayed under the list, along with the HCF file details.
- **Options Group:**
  The options available here are all Inserter dependent, and thus will change based upon the Inserter model selection.
  To see how the selected Inserter markings would look on the printed page, click the Next button to move to the "Mark Position Options" on page 1466 page, which has a preview of the page. You can move back and forward between these two pages until you are entirely satisfied with the selections made.
  - **Mark Configuration**: Use the drop-down to select the type of markings to add. This selection basically equates to the amount of area the markings will take up on the printed page.
  - **Fold Type**: Use the drop-down to select the type of fold to apply to the paper. This will impact upon where on the page the markings will be placed.

- **Collation level**: Select whether the markings will be made at Document level, or Document Set level.
- **Print marks on back**: Check to place the Inserter Marks on the rear of the page.
- **Selective Inserts**: If selective inserts are supported by the chosen Mark Configuration you can select what markings to include and whether those markings are to included based upon some conditional setting. This can be done by highlighting the Mark Name entry and either pressing the **Edit** button, or using the right mouse click context menu, and selecting **Edit**.
  For information on how to edit the Selective Inserts settings, see "Selective Insert Dialog" on page 1467

- **Clear Background Area** Tab:
  Check the **Clear Background Area** checkbox to add a white background to the OMR, preventing background colors or elements interfering with the OMR Markings when they are read by the Inserter.
  - **Margins**:
    - **Same for all sides**: Check so that the Left margin selection is used to set all sides identically.
    - **Left, top, right, bottom**: Enter measurements for the margins on each side of the OMR Marks.
- **Custom OMR mark sizing** Tab:
  If supported by the currently chosen Mark Configuration you can select a Custom OMR size by checking the **Custom OMR mark sizing** checkbox.
  Select from any of the following, or leave the entries blank to use default values:
  - **Line length**: Enter a value between 10.16mm and 20mm.
  - **Line thickness**: Enter a value between 0.254mm and 0.63mm.
  - **Gap distance**: Enter a millimeter value 2.91mm and 4.2mm.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on

page 1380 page) and add or remove printing options from the print run.

- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Mark Position Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Mark Position Options** page displays a Preview of the output and the possible locations to place the inserter marks. The initial settings are determined by the selections made within the "Inserter options" on page 1464 page.
You can move back and forward between these two pages to perfect the settings, or you could move the inserter mark box to the desired location on the preview.

Preview box:

- The *pink area* displays the areas of the page where inserter marks can be positioned.
- The *small checkered box* displays the current location of the inserter marks. This box is selectable and can be dragged to the desired location within the printable (pink) areas.
  If the box is placed outside the printable areas the page will display an error and prevent attempts at leaving the page.

Below the Preview box are buttons which allow control of the Preview box. The selections that can be made are:

- **First Page**: Click to jump to the first page.
- **Previous Page**: Click to move to the previous page.
- **Next Page**: Click to move to the next page.
- **Last Page**: Click to jump to the last page.

- **Show Page**: Use the up and down arrows or type a page number to display a specific page within the document.
- **Zoom in/out**: Click to zoom in or out by 25%
- **Zoom Level**: Use the drop-down to select a predefined level or enter a zooming percentage.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Selective Insert Dialog

The Selective Insert Dialog provides control over individual inserts. In this dialog the selected can be set to be either selected ("Yes") or not ("No"), or selected based upon some conditional criteria ("Conditional").

For example, you could add a marking to the third page of a document by making the selection Conditional and then setting the Condition entry to "`page.nr == 3`".

Both Metadata and Informational fields can be used in the Conditionals settings.

For more details on how to create a conditional, see the "How to set up conditions" on page 1453 page.

Dynamic PPD Options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Dynamic PPD mode options** page allows you to add PostScript printer capabilities to Connect, through the printer's associated PostScript Printer Definition (PPD) file, and to assign rules to them to trigger those capabilities in conjunction with Connect Templates.

> **Note:** This Print Wizard page is intended for use only by those operators who are very familiar with their PostScript printer's capabilities and the options available to them in the printer's associated PPD file.
>
> A video example of how to set Dynamic PPD Options can be found here: Setting Dynamic PPD Options within Connect - An example.

> **Note:** As of OL Connect 2019.1 *JobPatchFile* sections in PPDs are automatically added to the output, if the PPD includes them.

The **Dynamic PPD mode Options** page can be broken up as follows:

PPD Info group

This shows information about the selected PostScript Printer Definition (PPD). The information box contains general printer details (such as the model name) whilst the **PPD file name** field displays the name of the actual PPD file that is loaded.

If the incorrect PPD was assigned to this Printer Model, the PPD file can be changed at this point, via the **Import PPD** button.

This is not considered good practice, however. It would be better to return to the Printer Model selection dialog and chose the correct Printer Model and PPD combination there.

PostScript Options

PostScript specific options that are not catered for by the PPD.

- **Rotate landscape pages** checkbox: For certain printers, printing in landscape orientation requires rotating the page content. Typically this cannot be derived from the PPD, nor does the PPD contain instructions for carrying out this rotation.

  In such a scenario, selecting this checkbox will have those printers print in Landscape

mode.

- **Title** selection: Enter the Title to add to the PostScript output. The default is `${template}`, which uses the Template name.

Rules

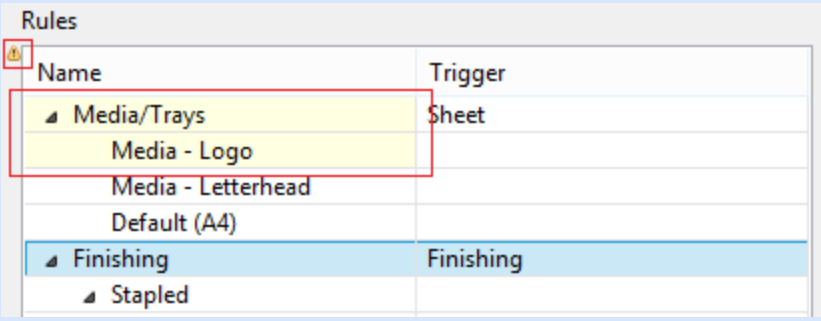
This group displays the print options for the template along with their associated triggers, as well at the more general Simplex/Duplex options.

If your Template already has **Finishing** and **Media/Trays** entries set, then those options will appear pre-populated in the Rules table along with their Triggers, to allow you to assign specific options as **Printer Option** entries.

Additional Rules (and Triggers) can be added (or deleted) and all Rules can be edited, and/or moved up or down within the table after being added.

Within a group of Rules, the Rules are processed sequentially, from top to bottom. Thus if two Rules apply to the same option (such as two different Rules applying for Tray 1), then the topmost Rule will be applied, once the Rule trigger is encountered. Thus it makes sense to have the most specific rules at the top of a group, with the more generic rules placed below them.

Note: Rules should have at least one **Printer Option** assigned to them. If they do not, then the page is considered incomplete and a warning symbol will be displayed and the offending Rules highlighted. As seen in the following screen-shot:



If a **Media/Trays** Rule has an associated **Printer Option** that matches a Media entry in the Template or Presets, then it is highlighted in light green.

It is possible to create a set of Rules in which none of the Rules ever evaluate to true, and thus no Rule will be applied. This could be done on purpose, to allow the printer's default settings to apply.

The types of Rules are as follows:

- **JCL** (Job Control Language) Rules: This option only appears if the PPD contains a "`*JCL keywords`" section.
  Certain printers require Printer Job Language (a type of JCL) commands to function properly, or to allow certain features.
  If such entries are included in the PPD, then rules can be added to use them via the **Add** button.

  Note: JCL is a generic way to refer to languages like PCL's Printer Job Language.

- **Prolog** Rules: This adds options to the Prolog section of a PostScript document.
  You can add Prolog settings via the **Add** button.

- **Setup** Rules: This supplies PostScript resources for document or page setup.
  You can add Setup settings via the **Add** button.

- **Media/Trays** Rules: This allows you to set Media/Trays options. The Media/Trays settings will be pre-populated from settings contained within the current Connect Template, if such exist.
  They can also be added from other Templates, selected using the **Import** button.

  You can import settings from as many Templates as you like.
  You can also add additional Media/Trays settings via the **Add** button.

  Media/Tray selections always apply to *Sheets*.
  Once the Media/Trays are entered you will need to assign specific Printer Options (from those contained in the PPD) to the individual Media/Tray types.

- **Finishing** Rules: This allows you to set Finishing options.
  These settings will be pre-populated from settings contained within the current Connect Template, if such exist.
  They can also be added from Job Presets and other Templates, selected using the **Import** button.
  You can import settings from as many Templates as you like.
  You can also add additional Finishing settings via the **Add** button.

  The Finishing Rules support all industry standard binding options.
  Once the Finishing Rules are entered you will need to assign specific Printer Options (from those contained in the PPD) to each of the individual Finishing Rules.

  Note: You should also choose an Option to switch off the Finishing selection, to cater for subset Finishing.

- **Plex** rules: This fixed group allows you to set rules for sheet plexing. The choices are Duplex, Simplex or Tumble Duplex.
  You should add Printer Options for each of these Plexing Rules.
- **Custom** rules: The Custom group does not appear by default, but can be added at any point via the **Add** button.

The **Rules** Options. These options can either be selected via the icons to the right of the Rules table, or from right click context sensitive menus within the Rules table.
The options are:

- **Import**: Allows importation of Rules from existing Job Presets and/or Connect Templates. You will be presented with options to either **Import Finishing Settings**, or **Import Media Settings**, based upon what is currently selected and/or available in the Rules table.
  Both options allow you to select an existing Connect Template, from which to import the settings from, whilst Finishing settings can also be imported from Job Presets.
  The settings will be added to those already in the Rules table.
- **Add**: This allows you to Add various options, dynamically.

  What options are available is dependant upon what rule is currently selected. For example, the *Plex* group does not have an option to **Add Rules** available to it.

  1. **Add Rule**: This will add a Rule that is specific to the type of option was selected when the Add Rule was pressed. The options are as follows:
     - **Prolog** : If the Rule is being added to a *Prolog* selection, then the Dynamic Custom Element Editor Dialog will be launched.You should add Printer Options for each of the Rules added. See the "Printer Options" on page 1474 section for how to do this.
     - **Setup** : If the Rule is being added to a *Setup* selection, then the Dynamic Custom Element Editor Dialog will be launched.
       You should add Printer Options for each of the Rules added.See the "Printer Options" on page 1474 section for how to do this.
     - **Media/Trays** : If the Rule is being added to a *Media/Trays* selection, then the Dynamic Tray Mapping Editor Dialog will be launched. This dialog allows the description of standard Media options, such as paper Weight and Color.
       You should add Printer Options for each of the Rules added.See the "Printer Options" on page 1474 section for how to do this.

- **Finishing** : If the Rule is being added to a *Finishing* selection, then the Dynamic Finishing Editor Dialog will be launched.
  You should add Printer Options for each of the Rules added.See the "Printer Options" on page 1474 section for how to do this.
- **Plex** : Add Rule is not available as an option for *Plex* settings.
  You should add Printer Options for each of the Rules added.See the "Printer Options" on page 1474 section for how to do this.
- **Custom** : If the Rule is being added to a *Custom Group* selection, then the Dynamic Custom Element Editor Dialog will be launched. The *Custom Group* selection allows for considerable customization and complexity. For example, they could be used to separate large documents (that might not fit into an Inserter machine), or to use a letterhead stock that is based upon what city the document recipient lives in.
  You should add Printer Options for each of the Rules added.See the "Printer Options" on page 1474 section for how to do this.

2. **Add Custom Group**: This launches the "Custom Group Editor Dialog" on page 1477 which adds a new Custom Group entry, and associated Trigger.
3. **Add PSHeader Group**: This adds a PostScript Header group (triggered on File). To add the actual Rule details you must select the new Header entry and then select the **Add Rule** option.
   Note that these rule are added in order the appear.
4. **Add Media/Trays Group**: This adds a new blank Media/Trays entry.
   To add the actual Rule details you must select the new Media/Tray entry and then select the **Add Rule** option.
5. **Add Finishing Group**: This adds a new blank Finishing entry.
   To add the actual Rule details you must select the new Finishing entry and then select the **Add Rule** option.

Each selection allows you to set the appropriate details for that type of rule.

- **Edit**: This allows you to edit the current Rule, dynamically. What options are available is dependant upon what rule is currently selected.
  The options are as follows:
  - **Media/Trays** : If the Rule is being edited is a *Media/Trays* selection, then the Dynamic Tray Mapping Editor Dialog will be launched.
  - **Finishing** : If the Rule is being edited is a *Finishing* selection, then the Dynamic Finishing Editor Dialog will be launched.

- **Plex** : *Plex* Rules cannot be edited.
- **Custom / Prolog / Setup** : If the Rule being edited is any of these, then the "Custom Group Editor Dialog" on page 1477 will be launched.

Rules can also be edited via double mouse clicking on the rule. This launches the appropriate Editor Dialog, as listed above.

- / **Move Up / Down**: Move the selected rule(s) up or down within the Rules table. This allows sorting of options, for easier legibility, and more importantly, *for Order Dependency*.
  Within a group of Rules, the Rules are processed sequentially, from top to bottom. Thus if two Rules apply to the same option (such as two different Rules applying for Tray 1), then the topmost Rule will be applied, once the Rule trigger is encountered. Thus it makes sense to have the most specific rules at the top of a group, with the more generic rules placed below them.
- **Delete**: Delete the selected rule(s) from the Rules table.

Rule Details

This table displays the details of the currently selected Rule.

For Media/Trays, Finishing and Plex selections only the **Properties** are displayed. For Custom selections, the **Condition** can also be displayed.

The tabs can be broken down as follows:

- **Properties** Tab: displaying all the relevant Rule Details. These are:
  - **Property**: The list of properties available to this Rule.
    What properties are displayed is dependent upon the type of Rule selected.
  - **Value**: The property's associated value, if set. If not set, the Value will remain blank.

  The table is read only and the details cannot be adjusted or edited from within the table. Rules need to be edited from within the Rules table directly, through either the Edit button or by double mouse clicking on the Rule.
- **Condition** Tab: displays the logical condition that needs to be fulfilled before the Custom Rule is to be applied.

Printer Options

This section allows for the selection and ordering of Printer Options to be associated with the active Rule. The Print Options are all those contained in the PPD.

If no Rule is active, the Printer Options section will not be available.

The **Printer Options** table displays :

- **Name** column: This column contains the names of the Printer Options, as taken from the PPD file.
- **Option** column: This column shows the selected Printer Option preference.
- **Order** column: This column displays any Dependency Order criteria that applies to the selected Printer Option.

The choices available to the Printer Options are as follows:

- **Add**: This launches the Add Printer Options Dialog which allows you to add one or more Printer Options. That dialog will be populated with options that relate to the Rule that was selected, when the Add Printer Options button was pressed.
- **Add custom PostScript**: This launches the Add Custom Printer Option dialog which allows you to add customized PostScript instructions.
  Note that this option requires an in depth understanding of both the PostScript language and the specific printer itself.
- **Edit**: This allows you to edit an existing Rule Printer Option. It launches the Add Printer Options Dialog.
- / **Move Up / Down**: Move the selected Printer Option(s) up or down within the table. This allows sorting of options for easier legibility, but manually moving options up and down is done without any check on Order Dependency.
- **Delete**: Delete the selected Printer Option(s).
- **Sort by Order dependency**: This sorts the Printer Options by the order of their dependencies.
  Not all Printer Options have dependencies, but for those that do, the order is important.
  For example, consider these following PPD file Printer Option extracts:
  - ***PageRegion:***

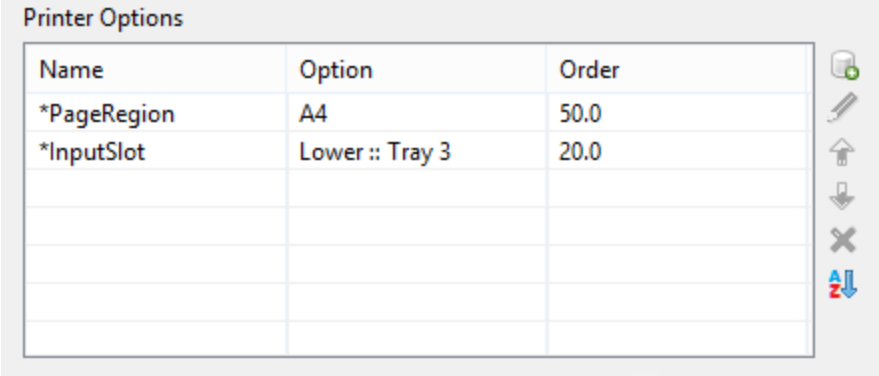
    ```
    *OpenUI *PageRegion: PickOne
    *OrderDependency: 50 AnySetup *PageRegion
    ```

```
...
*CloseUI: *PageRegion
```

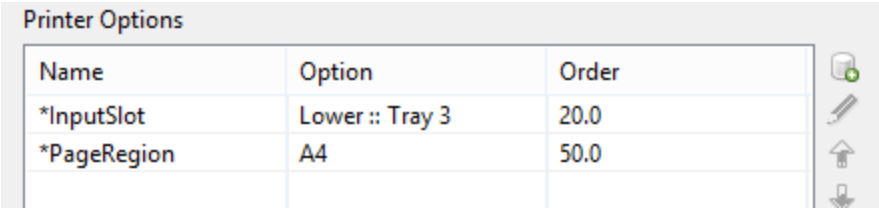
- ***InputSlot:***

```
*OpenUI *InputSlot: PickOne
*OrderDependency: 20 AnySetup *InputSlot
...
*CloseUI: *InputSlot
```

We can see that one has an Order Dependency of 50 and the other 20.
If *PageRegion* was selected prior to *InputSlot* in the Add Printer Options Dialog, then they would appear like this in the Print Wizard:



To sort them into the correct dependance order, use the **Sort by Order dependency** button. This will sort them into the proper dependency sequence. As seen below, where the lower dependency now appears first.



Caution: Nothing prevents the selection of conflicting and contradictory Print Options.

This can be seen in the following example, which shows multiple conflicting Folding Options have been chosen:

Printer Options

Name	Option	Order
*Punch :: Punch	None :: Off	1.0
*Staple :: Staple	None :: Off	21.0
*Prepunched :: Pre-Punched	True :: On	65.0
*SquareFold :: Spine Corner Forming	True :: On	1.0
*MultiFolder :: Multi Folder	FD503(2-4H) :: Multi F...	1.0
*Fold :: Fold	ZFold :: Z-Fold	1.0

The breadth and diversity of PPD files as well as individual printer capabilities mean that the Print Wizard cannot meaningfully validate the selections made, and thus does not attempt to.

So you must be careful when making selections. We recommend that you always test each individual selection made, to confirm that they are valid and that they do what you expected of them.

This could entail considerable trial and error, but should lead to less anguish in the long run.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Custom Group Editor Dialog

The **Custom Group Editor** dialog allows you associate a Custom Rule with a trigger. It is launched when either adding a new Custom Group rule, or editing an existing one in the "Dynamic PPD Options" on page 1468 page of the Printer Wizard.

The **Custom Group Editor** dialog can be broken up as follows:

- **Group name** edit box: What name to give the Custom Group Rule in the "Dynamic PPD Options" on page 1468 page.
- **Trigger on** drop down list box: Select from a series of available Connect Template resource related triggers.

Separation options

This page appears as part of the **Advanced Print Wizard** and the print Output Preset wizard.

The **Separation Options** page defines how to separate a job into discreet portions. In addition the printer can add banner pages, slip sheets, and/or jog the output.

Note: Jogging is not an option on Continuous Form printers.

- **Separate by Count** group:
  This group allows for the splitting of output based upon a specified number of *Sheets*, *Documents*, *Document Sets* or *Job Segments*.
  - **Split**: Use the drop-down to select whether or not to split the job based on number of *Sheets*, *Documents*, *Document Sets* or *Job Segments*.
    - **None**: Select this option to ignore count splitting.
    - **At Exactly**: Select to create a split based upon a predetermined number of *Sheets*, *Documents*, *Document Sets* or *Job Segments*.
  - **Every**: Enter the number of *Sheets*, *Documents*, *Document Sets* or *Job Segments* at which the output is to be split.
    This option is only available if the **Split** entry has been set to *"At Exactly"*.
  - Use the drop-down box to chose what the separation option is to be.
    The selection is either **Sheets**, **Documents**, **Document Sets** or **Job Segments**.
    The default separation option is *Sheets*, which was the only option available for splitting prior to Connect 2019.1.

> **Note:** If the "**Separation by Count**" option is enabled, you can use the `segment.metadata` variable to set a dynamic file name. See the Segment section in the "Print output variables" on page 1659 for more information on setting this variable.

- **Separation Settings** group:
  This setting is only available if no **Separate by Count** split has been specified.
  - **Separation**: Use the drop-down to select when a job separation occurs, which is either **None** (no separation) or at the **Job**, **Job Segment**, **Document** or **Document Set** level.
- **Banners** (banner page) group:
  This setting adds banner pages to print jobs, at the selected level.
  - **Every**: Use the drop-down to select where the banner page is to be inserted. Choices are is either **None** (no banner page, which is the default) or at the **Job**, **Job Segment**, **Document** or **Document Set** level.

> **Note:** To add content to the banner page, the content can be made conditional on `sheet.banner`, or `page.banner` in the "Additional Text Settings" on page 1411.

> **Note:** If the sheet following the banner is simplex, the banner will be simplex, but it the sheet following the banner is duplex, the banner will default to duplex. This ensures that a duplex print job doesn't accidentally become mixed-plex, which is not supported by all printers.

> **Tip:** When banners are enabled, the overview of output preset properties shows a property "Banners", with a value that reflects the setting.

- **Slip Sheets** group:
  - **Add slip sheet**: Use the drop-down to select whether to add a slip sheet before or after a specific separation, or whether to use none.
  - **Every**: Use the drop-down to select at which separation to add a slip sheet, at the **Job**, **Job Segment**, **Document** or **Document Set** level.
  - **Media Size**: Use the drop-down to select the media size of the slip sheet.
    If a custom Media Size was chosen:

- **Width**: enter slip sheet page width.
- **Height**: enter slip sheet page height.

Note: When both banners and slip sheets are enabled for the same level, the slip sheet will go in front of the banner.

- **Jog** group:

Note: Jogging is not an option on Continuous Form printers.

- **Jog after every**: Use the drop-down to select when to jog the printer, which is either **None** (no forced jogging) or at the **Job**, **Job Segment**, **Document** or **Document Set** level.

Advanced Printing Wizard and Output Preset navigation options

- **Preview** button (active in ***Advanced Print*** Wizard only): Click to launch a Proof Preview window, which displays how the printed output would look based upon the currently chosen selections.
- **Back** and **Next** buttons: Used to navigate back and forth through all the selected options within the Wizard. Up until the Print button is pressed, one can reverse all the way through the wizard to return to the main selection page (the "Print options" on page 1380 page) and add or remove printing options from the print run.
- **Print** button (***Advanced Print*** Wizard only) or **Finish** button (***Output Preset*** Wizard only):
  Click to produce print output/finalize the Preset according to the current settings. This can be done at any point within the Wizard, whether or not *all* the options selected in the "Print options" on page 1380 page have been completed or not.
- **Cancel** button: Cancels the Print Wizard, without creating any printout or altering Preset.

Designer Script API

In Designer templates, every bit of information can be tailor-made, using scripts. When Connect generates actual output – letters, web pages or emails -, it opens a record set and merges it with the template. It takes each record, one by one, and runs all scripts for it (in a specific order, see "The script flow: when scripts run" on page 1082).

Most scripts can be made using one of the Script Wizards (see "Personalizing content" on page 923).
However, when you want to do more than what you can do with a Wizard, you may write a

script yourself. If you are not familiar with writing scripts, please read "Writing your own scripts" on page 1064 first.

All scripts in the Designer have to be written in JavaScript.
If you don't know JavaScript, the many examples given in this API will help you get started. It is worth the effort, however, to familiarize yourself with the JavaScript syntax. For a simple script all you need to know can be found on the following web pages: https://www.w3schools.com/js/js_syntax.asp and https://www.w3schools.com/js/js_if_else.asp. A complete JavaScript guide for beginners can be found on MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

In the editor window, press **Ctrl** + **Space** to see the available features and their descriptions. Use the arrow keys to select a function or object and press Enter to insert it in the script. Type a **dot** after the name of the function or object and press Ctrl + space again to see which features are subsequently available.
For more keyboard shortcuts, see "Keyboard shortcuts" on page 1223.

Designer API

The "Standard Script API" on the next page describes the objects and functions that are available in **template scripts**, created inside the Scripts pane. Template scripts change the contents of sections in a template.

> **Note:** In a Print context, the scripts in the Scripts pane run once for each section and once for each Master Page (see "Master Pages" on page 648).

Control Script API

Control Scripts are a special kind of Designer Scripts. They don't touch the content of the sections themselves, but they change the way a template is outputted, for example by selecting or omitting sections from the output.
For more information about Control Scripts and their use, see "Control Scripts" on page 1098.
Features that are specific to Control Scripts are listed in the "Control Script API" on page 1591.

Post Pagination Script API

Post Pagination Scripts are run in a **Print** context **after** the content has been paginated. Because they can search through the output of all Print sections, and modify Print sections

(one at a time), they may be used to create a Table Of Contents (TOC), as explained in the topic: "Creating a Table Of Contents" on page 1116.

For more information about Post Pagination Scripts and their use, see "Post Pagination Scripts" on page 1114.
Features that are specific to Control Scripts are listed in the "Post Pagination Script API" on page 1619.

Standard Script API

This page lists the global objects and functions that are available in Standard Scripts, created inside the Scripts pane. Click through to an object or function to get a description and examples.

Most of these objects and functions are also available in Control Scripts ("Control Scripts" on page 1098) and Post Pagination Scripts ("Post Pagination Scripts" on page 1114).
For objects and functions restricted to Control Scripts see "Control Script API" on page 1591, and for the Post Pagination Script API see "Post Pagination Script API" on page 1619.

The basics of script-writing in the Designer are explained in the following topic: "Writing your own scripts" on page 1064.

For tips and tricks to use when writing template scripts., see "Writing scripts: tips and tricks" on page 1089.

Objects

Object	Description
"results" on page 1623	This object - of the type: QueryResults - is used to manipulate the content of the template. It contains the HTML element or set of HTML elements that match the selector of the script, specified in the script editor. This object is not available in Control Scripts, because that type of script doesn't have a selector (see "Control Scripts" on page 1098).
"this" on page 1554	A Standard script or Post Pagination script that has its scope set to "Each matched element" (see "Setting the scope of a script" on page 1071) will be called in a loop over the elements that match the selector of the script - the `results` (see "results" on page 1623). In such a script, `this` is an object of the type QueryResult. It represents the current element in the loop. This object is not available in Control Scripts, because that type of script doesn't have a selector (see "Control Scripts" on page 1098).
"record" on page 1515	The record in the main data set that is currently being merged. To get the value of a field in the record, use `record.fields['fieldname']` or `record.fields.fieldname`.
"logger" on page 1502	Global object that allows you to log messages.

Object	Description
"Locale" on page 921	Defines which locale to use. Note that the value is read-only.
"formatter" on page 1486	Global object that allows you to format values (such as a date or number).
"automation" on page 1592	This object encapsulates the properties of the Workflow process that triggered the current operation. Not available in PrintShopMail Connect.
"merge" on page 1635	The `merge` object is mainly used in Control Scripts. It gives access to the template with all of its contexts, sections and runtime parameters. It doesn't give access to the content of the sections. To change the content of a section, you would create a script with a selector and use the `results` object in the script (see "results" on page 1623).
"contentitem" on page 1620	(Print output only.) The `contentitem` object holds the Print Content Item that is written to the Connect database when generating Print output. It allows you to add custom properties to the Print Content Item, in the form of key-value pairs (a JSON string). **Note:** The `contentitem` object is not available in PrintShopMail Connect, due to the absence of a Connect database.

Global functions

Function	Description
"fatalError(message)" on page 1484	Triggers a fatal error that aborts content creation.
"loadhtml()" on page 1493	Loads HTML data from an HTML (snippet). The returned HTML can be placed into a variable or into a set of HTML elements.
"loadjson()" on page 1497	Loads JSON data from a URL. This is a simple way to retrieve content from external systems.
"query()" on page 1513	Performs a query in the template's contents and creates a new result set containing the HTML elements that match the given CSS selector.
"resource()" on page 1599	This function returns information about an image resource. It can also be used to check if a file exists.
"translate()" on page 1590	Gets the translation of a source text with an (optional) context. It returns the message as-is if no translation is found.

each()

A generic iterator function, to iterate over the elements in a result set.

Tip: Instead of using the `each()` function to loop over a result set, you could set the scope of a Standard script or Post Pagination script to "Each matched element" and access each matched element directly via the `this` object. See "Setting the scope of a script" on page 1071 and "this" on page 1554.

each(callback)

Iterates over the elements in a set, such as the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

callback

A function. The callback function is passed the iteration index and the current element. In the scope of the callback function, `this` refers to the current element.

Examples

The following scripts demonstrate a simple iteration over the elements in the `results` (the set of HTML elements that match the selector of the script).

The first script sets the background color of each of the elements to red. (This is just to demonstrate how this function works. It is easier to change the style of a set of HTML elements using the `css()` function; see "css()" on page 1571.)

```
results.each(function(index){
    results[index].css('background-color','red');
});
```

This script adds a random integer to each element in the result set.

```
results.each(function(index){
    var test = Math.floor((Math.random() * 10) + 1);
    this.html(test);
});
```

Selector	Matched element	Matched element after script execution
p	 	3 1 7

The next script gets the row index (of the current element in the set) and puts it in a paragraph.

```
results.each(function(index){
    this.text(index);
}
```

Selector	Matched element	Matched element after script execution
p	 	0 1 2

Using `each()` in a translation script

The following script first loads a snippet containing translation strings, depending on the value of a field. Then it inserts translations by iterating over elements in the `results` (the set of HTML elements that match the selector of the script) and setting the HTML of each element with a value from the array of translation strings.

```
var strings = loadjson('snippets/' + record.fields.locale + '.html');
results.each(function(index){
    if( strings[this.attr('data-translate')])
    this.html(strings[this.attr('data-translate')]);
});
```

Note: For documentation on the `data-*` attribute, see MDN Web Docs.

Selector	Matched element	Matched element after script execution
p	 	primero último dirección de correo electrónico

fatalError(message)

The `fatalError()` function triggers a fatal error that aborts content creation.

message

A string.

When a script calls this function in Preview mode, the script that triggers it is marked with an error icon in the Scripts pane, and the given message is displayed in a hint.

When generating output from the Designer, the Designer will log the error and display an error dialog with the given message. Content creation is aborted.

When generating output from Workflow, the job fails. Workflow will log the error and execute any follow-up actions that are defined in the On Error tab of the respective OL Connect Content Creation task (All in One, Create Email Content, Create Print Content, Create Preview PDF, and Create Web Content). For more information about how to set up follow-up actions, see Using the On Error tab in the Workflow Help.

> **Note:** Errors thrown in a `try` statement do not abort content creation or trigger any follow-up actions in Workflow. Content creation will continue with the next script, the same record.

for(... in ...)

Can be used to iterate over fields in a data set or rows in detail table. Also see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in.

for(`variable` **in** `object`**) { ... }**

Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

Examples

This script iterates over field names in the current record and adds them to a paragraph.

```
for(var i in record.fields){
    results.after("<p>" + i + "</p>");
}
```

Selector	Matched element	Matched element after script execution
#test	<h1 id="test">Fields</h1>	<h1 id="test">Fields</h1> first last email

This script iterates over fields in the current record, retrieving their values. Then it adds the values to a paragraph.

```
for(var i in record.fields){
    results.after("<p>" + record.fields[i] + "</p>");
}
```

Selector	Matched element	Matched element after script execution
#test	<h1 id="test">Fields</h1>	<h1 id="test">Fields</h1> Peter Parker pparker@localhost.com

This script iterates over rows in a detail table and adds the contents of the 'country' field to a paragraph.

```
for(var i in record.tables['countries']) {
    results.after("<p>" + record.tables['countries'][i].fields['country'] + "</p>");
}
```

Selector	Matched element	Matched element after script execution
#countries	<h1 id="countries">Countries</h1>	<h1 id="countries">Countries</h1> The Netherlands Canada Australia

This script iterates over rows in a detail table and adds the contents of the 'ItemID2' field to an option. The <option> tag defines an option in a select list in an HTML form.

```
for(var i in record.tables['detail']) {
    var str = record.tables['detail'][i].fields["ItemID2"];
    results.append("<option value='" + str + "'>" + str + "</option>");
}
```

formatter

The `formatter` is a global object that allows you to format values in a script.

The Helper Wizard and the Text Script Wizard also allow you to format variable data; see "Using the Helper Wizard" on page 1003, "Using the Text Script Wizard" on page 948 and "Formatting variable data" on page 952.

Note: The TextFormatter object is now deprecated and will eventually be removed.

Functions

Function	Description
• currency() • currencyNoSymbol() • grouped() • integer() • integerUngrouped() • number() • numberUngrouped()	The `currency()`, `grouped()`, `integer()` and `number()` functions allow you to format a number, possibly with a custom pattern. See "Number functions" on page 1510.
• date() • dateLong() • dateMedium() • dateShort() • dateTime() • dateTimeLong() • dateTimeMedium() • dateTimeShort() • timeLong() • timeMedium() • timeShort()	The `date()`, `dateTime()` and `time()` functions allow you to format a date and/or time in different ways. See "Date, date/time and time functions" below.
• lowerCase() • upperCase() • properCase() • sentenceCase()	The text formatting functions are used on Strings. `lowerCase()` transforms all characters to lowercase, `upperCase()` transforms all characters to uppercase, `properCase()` transforms the first character of each word to uppercase and all other characters to lowercase, and `sentenceCase()` transforms the first character of the first word to uppercase.

Date, date/time and time functions

- date()
- dateLong()
- dateMedium()
- dateShort()
- dateTime()
- dateTimeLong()
- dateTimeMedium()

- dateTimeShort()
- time()
- timeLong()
- timeMedium()
- timeShort()

Note: The locale also influences the output of the different Date functions; see "Locale" on page 921.

Tip: To format a date from a `date` field in the record set, you can enter a custom pattern via the Helper Wizard and the Text Script Wizard; see "Using the Helper Wizard" on page 1003, "Using the Text Script Wizard" on page 948, "Formatting variable data" on page 952 and "Date and time patterns" on page 1491).

date(value, pattern)

Formats a date object using a custom pattern.

value

A Date object. A Date can contain a date and time.

pattern

String. The custom pattern may consist of pattern letters, separating symbols and quoted text, for example: "MMMM dd, yyyy"; see "Date and time patterns" on page 1491. Note that the repetition of pattern letters determines the exact presentation.

dateLong(value)

Formats a date as long string representation, for example **April 1, 2016**.

value

A Date object. A Date can contain a date and time.

dateMedium(value)

Formats a date as medium string representation, for example **1-Apr-2016**.

value

A Date object. A Date can contain a date and time.

dateShort(value)

Formats a date as short string representation, for example **4/1/16**.

value

A Date object. A Date can contain a date and time.

dateTime(value, pattern)

Formats a date and time object using a custom pattern.

value

A Date object. A Date can contain a date and time.

pattern

String. The custom pattern may consist of pattern letters, separating symbols and quoted text, for example: "yyyy.MM.dd G 'at' HH:mm:ss z"; see "Date and time patterns" on page 1491. Note that the repetition of pattern letters determines the exact presentation.

dateTimeLong(value)

Formats a date and time as long string representation, for example **April 1, 2016 12:00:00 EDT AM**.

value

A Date object. A Date can contain a date and time.

dateTimeMedium(value)

Formats a date and time as medium string representation, for example **1-Apr-2016 12:00:00 AM**.

value

A Date object. A Date can contain a date and time.

dateTimeShort(value)

Formats a date and time as short string representation, for example **1/4/16 12:00 AM**.

value

A Date object. A Date can contain a date and time.

time(value, pattern)

Formats a time using a custom pattern.

value

A Date object. A Date can contain a date and time.

pattern

String. The custom pattern may consist of pattern letters, separating symbols and quoted text, for example: "'at' HH:mm:ss z"; see "Date and time patterns" on the next page. Note that the repetition of pattern letters determines the exact presentation.

timeLong(value)

Formats a time as long string representation, for example **12:00:00 EDT AM**.

value

A Date object. A Date can contain a date and time.

timeMedium(value)

Formats a time as medium string representation, for example **12:00:00 AM**.

value

A Date object. A Date can contain a date and time.

timeShort(value)

Formats a time as short string representation, for example **12:00 AM**.

value

A Date object. A Date can contain a date and time.

Examples

The following script passes the value of a field in the record set to the date() function. This will only work if the type of the field has been set to Date in the data mapping configuration and if the field contains a valid date.

```
var myDate = formatter.date(records.fields.DATE, "MM/dd/yyyy");
```

The custom pattern that the script provides, outputs the month and day in two digits each and the year in four digits: 05/21/2016. For more examples of formatting patterns, see "Date and time patterns" on the next page.

Creating a Date object from a string

In a data mapping configuration you can set the type of a field to Date, but when you open a data file or database in the Designer without a data mapping configuration, all fields are text fields (fields of the type `string`). The `formatter` cannot be used to format a string with a particular date format. The solution is to store the string in a variable as a Date object, and use the `formatter` with that variable.

The following sample script demonstrates this solution. It splits a string into parts and then creates a new Date object with the parts in the correct order. To construct a Date, the parts of the date must be put in the following order: year, month, day, and optionally hours, minutes, seconds, milliseconds (see MDN Web Docs - Date object.) When the time is omitted, it defaults to 12:00:00 AM.

```
/* Convert the string 21-12-1997 into a valid JavaScript date */
        var strDate = record.fields["date"];
        var dateParts = strDate.split("-");
        var date = new Date(dateParts[2], (dateParts[1] - 1), dateParts[0]);
```

Note: JavaScript counts months from 0 to 11. January is 0. December is 11.

Another way to put a string in a Date is to use the `Date.parse` function; see MDN Web Docs - Date parse.

The `date` variable can be used as the value in the date, dateTime or time functions of the `formatter`.

```
var myDate = formatter.date(date, "MM/dd/yyyy");
```

The custom pattern that the script provides, outputs the month and day in two digits each and the year in four digits: 05/21/2016. For more examples of formatting patterns, see "Date and time patterns" below.

Date and time patterns

Dates and times in a template originating from a `date` field in a record set can be displayed using a custom pattern. You can enter the pattern via the Helper Wizard or the Text Script Wizard, lest the field type is set to Date in a data mapping configuration and the field contains a valid date; see "Using the Helper Wizard" on page 1003, "Using the Text Script Wizard" on page 948 and "Formatting variable data" on page 952. In the Script Editor, the pattern can be passed to a date, dateTime or Time function of the `formatter`; see "formatter" on page 1486.

The custom pattern may consist of pattern letters (see below), for example: "MM/dd/yyyy". The components can be separated with a space or a symbol, e.g. `.`, `/`, `-`. Text must be put in quotes.

The repetition of pattern letters determines the exact presentation. For example, if the number of pattern letters for a month is less than 3 (M or MM), the month is displayed as a number. If the number of pattern letters is 3 (MMM), it will be displayed as text; if available, a short or abbreviated form of the month's name will be used. If the number of pattern letters is 4 or more (MMMM), the month's full name is displayed.

Note: The pattern letters and patterns on this page are only suitable for **displaying** dates and times in templates, not for extracting dates in the DataMapper module. For pattern letters and patterns available in the DataMapper, see "Date" on page 439.

Pattern characters

Letter	Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
Y	Week year	Year	2009; 09
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day name in week	Text	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	Number	1
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0

k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800
X	Time zone	ISO 8601 time zone	-08; -0800; -08:00

Note: These date and time pattern letters and patterns conform to standard Java notation. For more information, see http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html.

loadhtml()

Loads HTML content from the specified HTML file. The file may be located inside the template (see "Snippets" on page 867) or hosted on a Content Management System or on another location outside the template.

An optional `selector` allows you to retrieve only the content of matching elements.

Note:

- The specified HTML file is expected to be UTF-8 encoded.
- If the HTML contains a Dynamic Table, the table will be expanded (i.e. filled with data).
- Any Handlebars expressions in the retrieved HTML are evaluated, using the record as input, if the current section is set accordingly. (To enable evaluating Handlebars expressions, right-click the section in the Resources pane and choose Properties.)
- Loadhtml() is cached per batch run (based on the URL) in print/email.

Tip: To load a JavaScript file (.js) or a style sheet (.css) you can use `loadtext()`. See "loadtext()" on page 1499.

Tip: External content is not loaded while editing a script. To test a script that loads external content, you can do a preflight; see "Doing a Preflight" on page 1075.

Note:
In OL Connect 2025.1 stricter checking was introduced for the Uniform Resource Identifier (URI) passed into the `loadhtml`, `loadjson` and `loadtext` scripting functions. Special characters such as spaces in a path used to work but now need to be properly URI encoded.

Example:

```
 // Worked in 2024.2 but not in 2025.1
 var result = loadhtml("file:///C:/PPWorkDir/Hello
World/Data/Example.html");
 // 2025.1 onwards
 var result = loadhtml
("file:///C:/PPWorkDir/Hello%20World/Data/Example.html");
```

This applies to more than just the space character. Refer to RFC 2396 for the rules on URI encoding.

loadhtml(location)

Loads all HTML from the specified HTML file or Handlebars snippet (.hbs).

location

String containing a path that can be absolute or relative to the section/context.
Use `snippets/<snippet-name>` to retrieve the content from an HTML file residing in the Snippets folder on the Resources panel.
In order to retrieve files from outside the template the `file` protocol is supported as well as the `http/https` protocols.

The complete syntax of a fully qualified URL with the "file" protocol is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/somefolder/somecontent.html.
If the file is located on another server in your network, the path must contain five slashes after "file:".

When using the http/https protocol, remember that only absolute paths are supported inside remote snippets (see "Remote snippets" on page 868).

Tip: To quickly get the location of a snippet in the template resources, right-click the snippet in the Resources pane and select **Copy Resource Location**.
The path to a remote snippet can be copied from the snippet's properties: right-click the snippet in the Resources pane and select **Properties**.

Note: Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file:////myserver/myfolder/file.txt).

Examples

This script loads a local HTML snippet (from the Resources panel) directly into the matched elements

```
results.loadhtml("snippets/snippet.html");
```

The following script loads a local HTML snippet (Resources panel) into a variable. The replaceWith() command is used to replace the element(s) matched by the script's selector with the contents of the snippet.

```
var mysnippet = loadhtml('snippets/snippet.html'); results.replaceWith(mysnippet);
```

Same result as the previous script, but a different notation:

```
results.replaceWith(loadhtml('snippets/snippet.html'));
```

The following script loads a snippet into a variable and finds/replaces text in the variable before inserting the content into the page. The second `find` command also adds formatting to the replacing text.

```
var mysnippet = loadhtml('file:///C:/Users/PParker/Documents/Example.html');
mysnippet.find('@var1@').text('OL Connect 1');
mysnippet.find('@var2@').html('<i>OL Connect 2</i>').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

This last script loads a remote snippet into a variable and retrieves an element from the snippet using query().

```
var mysnippet = loadhtml('http://www.somewebsite.com/text-root-wrapped.html');
var subject = query("#subject", mysnippet).text();
results.append("<p style='font-weight: bold;'>" + subject + "</p>");
```

loadhtml(location, selector)

Retrieves specific content from the specified HTML file or Handlebars snippet (.hbs).

location

String containing a path that can be absolute or relative to the section/context.
Use `snippets/<snippet-name>` to retrieve the content from an HTML file residing in the Snippets folder on the Resources panel.
In order to retrieve files from outside the template the `file` protocol is supported as well as the `http/https` protocols.

The complete syntax of a fully qualified URL with the "file" protocol is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/somefolder/somecontent.html.
If the file is located on another server in your network, the path must contain five slashes after "file:".

When using the http/https protocol, remember that only absolute paths are supported inside remote snippets (see "Remote snippets" on page 868).

Tip: To quickly get the location of a snippet in the template resources, right-click the snippet in the Resources pane and select **Copy Resource Location**.
The path to a remote snippet can be copied from the snippet's properties: right-click the snippet in the Resources pane and select **Properties**.

Note: Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file://///myserver/myfolder/file.txt).

selector

String. The supplied selector should conform to CSS selector syntax and allows you to retrieve only the content of matching elements.
If the selected element is a Dynamic Table, the retrieved HTML will contain the expanded table.

Examples

This script loads a **specific element** from a snippet and uses that to replace the `results` (the HTML element or set of HTML elements matched by the selector of the script; see

"results" on page 1623).

```
var mysnippet = loadhtml('snippets/snippet-selectors.html','#item3');
results.replaceWith(mysnippet);
```

This script loads the **children** of the selected element.

```
var snippet = loadhtml('file:///C:/Users/PParker/Documents/Example.html','foobar').children();
results.replaceWith(snippet);
```

The next script loads a **remote** snippet (see "Remote snippets" on page 868), looks for an H1 heading and uses that text.

```
var post = loadhtml('snippets/post.rhtml');
var h1 = query('h1', post).text();
results.text(h1);
```

Tip: Another example is given in the following how-to: Using a selector to load part of a snippet.

loadjson()

Creates a JSON object based on the text retrieved from the supplied location. The function lets you retrieve content from a JSON enabled server using a standard HTTP request. Popular content management systems like WordPress (requires JSON API plugin) and Drupal provide a JSON service/API to retrieve content.

Note:

- The specified JSON file is expected to be UTF-8, UTF-16 or UTF-32 encoded. A byte order mark specified as the first character will be stripped. Saving any changes will change the encoding to UTF-8.
- Loadjson() is cached per batch run (based on the URL) in print/email.
- This online JSON viewer is handy to debug JSON data: http://jsonviewer.stack.hu

Tip: External content is not loaded while editing a script. To test a script that loads external content, you can do a preflight; see "Doing a Preflight" on page 1075.

Note:
In OL Connect 2025.1 stricter checking was introduced for the Uniform Resource

Identifier (URI) passed into the `loadhtml`, `loadjson` and `loadtext` scripting functions. Special characters such as spaces in a path used to work but now need to be properly URI encoded.

Example:

```
 // Worked in 2024.2 but not in 2025.1
 var result = loadhtml("file:///C:/PPWorkDir/Hello
World/Data/Example.html");
 // 2025.1 onwards
 var result = loadhtml
("file:///C:/PPWorkDir/Hello%20World/Data/Example.html");
```

This applies to more than just the space character. Refer to RFC 2396 for the rules on URI encoding.

loadjson(location)

Loads json data from the specified location.

location

String; the supplied location should be either a URL or a relative file path. The `file` protocol is supported as well as the `http/https` protocols.

The complete syntax of a fully qualified URL with the "file" protocol is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/somefolder/somejson.json.

Examples

This sample script retrieves JSON data from a snippet.

```
var localJSON = loadjson('snippets/jsonsnippet.json');
if(localJSON.post){
    results.html("<h3>" + localJSON.post.title + "</h3><p>" + localJSON.post.modified + "</p>");
}
```

This script retrieves a post from a WordPress site.

```
var wpPost = loadjson('http://192.168.101.58/2013/06/leave-the-third-dimension-behind-and-focus-on-real-printing-innovation/?json=1');
if(wpPost.post){
    results.html("<h1>" + wpPost.post.title + "</h1>"
        + wpPost.post.content);
}
```

This script retrieves multiple posts from a WordPress site.

```
var numPosts = 3;
var wpPost = '';
var wpRecentPosts = loadjson('http://192.168.101.58/?json=get_recent_posts&count=' + numPosts);
if(wpRecentPosts.posts){
    for (var i = 0; i < numPosts ; i++) {
        wpPost += "<p>" + wpRecentPosts.posts[i].title + "</p>";
    }
}
results.after(wpPost)
```

loadtext()

Loads text from the specified text file, for example a JavaScript file (.js) or style sheet (.css). The text file may be located inside the template or hosted on a Content Management System or on another location outside the template.

Tip: To load an HTML fragment, you can use use `loadhtml()` (see "loadhtml()Loads HTML content from the specified HTML file. The file may be located inside the template (see "Snippets" on page 1) or hosted on a Content Management System or on another location outside the template.An optional selector allows you to retrieve only the content of matching elements. The specified HTML file is expected to be UTF-8 encoded.If the HTML contains a Dynamic Table, the table will be expanded (i.e. filled with data).Any Handlebars expressions in the retrieved HTML are evaluated, using the record as input, if the current section is set accordingly. (To enable evaluating Handlebars expressions, right-click the section in the Resources pane and choose Properties.)Loadhtml() is cached per batch run (based on the URL) in print/email.To load a JavaScript file (.js) or a style sheet (.css) you can use loadtext(). See "loadtext()" on page 1.External content is not loaded while editing a script. To test a script that loads external content, you can do a preflight; see "Doing a Preflight" on page 1.In OL Connect 2025.1 stricter checking was introduced for the Uniform Resource Identifier (URI) passed into the loadhtml, loadjson and loadtext scripting functions. Special characters such as spaces in a path used to work but now need to be properly URI encoded. Example: // Worked in 2024.2 but not in 2025.1 var result = loadhtml ("file:///C:/PPWorkDir/Hello World/Data/Example.html"); // 2025.1 onwards var result = loadhtml("file:///C:/PPWorkDir/Hello%20World/Data/Example.html");This applies to more than just the space character. Refer to RFC 2396 for the rules on URI encoding.loadhtml(location)Loads all HTML from the specified HTML file or Handlebars snippet (.hbs).locationString containing a path that can be absolute or relative to the section/context. Use snippets/<snippet-name> to retrieve the content from an HTML file residing in the Snippets folder on the Resources panel. In order to retrieve files from outside the template the file protocol is supported as well as the http/https protocols. The complete syntax of a fully qualified URL with the "file" protocol is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in

file:///<path>, for example: file:///c:/somefolder/somecontent.html.If the file is located on another server in your network, the path must contain five slashes after "file:".When using the http/https protocol, remember that only absolute paths are supported inside remote snippets (see "Remote snippets" on page 1).To quickly get the location of a snippet in the template resources, right-click the snippet in the Resources pane and select Copy Resource Location.The path to a remote snippet can be copied from the snippet's properties: right-click the snippet in the Resources pane and select Properties.Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file://///myserver/myfolder/file.txt).ExamplesThis script loads a local HTML snippet (from the Resources panel) directly into the matched elementsresults.loadhtml("snippets/snippet.html");The following script loads a local HTML snippet (Resources panel) into a variable. The replaceWith() command is used to replace the element(s) matched by the script's selector with the contents of the snippet.var mysnippet = loadhtml('snippets/snippet.html'); results.replaceWith (mysnippet);Same result as the previous script, but a different notation:results.replaceWith(loadhtml('snippets/snippet.html'));The following script loads a snippet into a variable and finds/replaces text in the variable before inserting the content into the page. The second find command also adds formatting to the replacing text.var mysnippet = loadhtml ('file:///C:/Users/PParker/Documents/Example.html'); mysnippet.find('@var1@').text('OL Connect 1'); mysnippet.find('@var2@').html('OL Connect 2').css('text-decoration','underline'); results.replaceWith(mysnippet); This last script loads a remote snippet into a variable and retrieves an element from the snippet using query().var mysnippet = loadhtml('http://www.somewebsite.com/text-root-wrapped.html'); var subject = query("#subject", mysnippet).text(); results.append(" " + subject + " ");loadhtml(location, selector)Retrieves specific content from the specified HTML file or Handlebars snippet (.hbs).locationString containing a path that can be absolute or relative to the section/context. Use snippets/<snippet-name> to retrieve the content from an HTML file residing in the Snippets folder on the Resources panel. In order to retrieve files from outside the template the file protocol is supported as well as the http/https protocols. The complete syntax of a fully qualified URL with the "file" protocol is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/somefolder/somecontent.html.If the file is located on another server in your network, the path must contain five slashes after "file:".When using the http/https protocol, remember that only absolute paths are supported inside remote snippets (see "Remote snippets" on page 1).To quickly get the location of a snippet in the template resources, right-click the snippet in the Resources pane and select Copy Resource Location.The path to a remote

snippet can be copied from the snippet's properties: right-click the snippet in the Resources pane and select Properties.Mapped network drives are usually not accessible to the server. Use a UNC path instead (e.g. file:////myserver/myfolder/file.txt).selectorString. The supplied selector should conform to CSS selector syntax and allows you to retrieve only the content of matching elements.If the selected element is a Dynamic Table, the retrieved HTML will contain the expanded table.ExamplesThis script loads a specific element from a snippet and uses that to replace the results (the HTML element or set of HTML elements matched by the selector of the script; see "results" on page 1).var mysnippet = loadhtml('snippets/snippet-selectors.html','#item3'); results.replaceWith (mysnippet);This script loads the children of the selected element.var snippet = loadhtml('file:///C:/Users/PParker/Documents/Example.html','foobar').children (); results.replaceWith(snippet);The next script loads a remote snippet (see "Remote snippets" on page 1), looks for an H1 heading and uses that text.var post = loadhtml ('snippets/post.rhtml');var h1 = query('h1', post).text();results.text(h1);Another example is given in the following how-to: Using a selector to load part of a snippet." on page 1); for JSON, use `loadjson()` ("loadjson()" on page 1497).

Tip: External content is not loaded while editing a script. To test a script that loads external content, you can do a preflight; see "Doing a Preflight" on page 1075.

Note:
In OL Connect 2025.1 stricter checking was introduced for the Uniform Resource Identifier (URI) passed into the `loadhtml`, `loadjson` and `loadtext` scripting functions. Special characters such as spaces in a path used to work but now need to be properly URI encoded.

Example:

```
 // Worked in 2024.2 but not in 2025.1
 var result = loadhtml("file:///C:/PPWorkDir/Hello
World/Data/Example.html");
 // 2025.1 onwards
 var result = loadhtml
("file:///C:/PPWorkDir/Hello%20World/Data/Example.html");
```

This applies to more than just the space character. Refer to RFC 2396 for the rules on URI encoding.

loadtext(location)

Returns the content of a text file. The file extension doesn't have to be .txt. It may also be a JavaScript file (.js) or a style sheet (.css), for instance.

location

String containing a path that can either be a URL or a path that is relative to the section/context.
In order to retrieve files from outside the template the `file` protocol is supported as well as the `http/https` protocols.

The complete syntax of a fully qualified URL with the "file" protocol is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: file:///c:/somefolder/somecontent.js.

Examples

This script loads a JavaScript file (from the Resources pane) directly into a <script> element.

```
var js = loadtext("js/my-script.js");
results.append(query("<script>").text(js));
```

The following script loads a style sheet (from the Resources pane) into the <style> element.

```
var css = loadtext("css/my-styles.css");
results.append(query("<style type='text/css'>").text(css));
```

logger

This is a global `ScriptLogger` object that allows logging messages such as error, warning or informational messages. The messages will appear in the **Messages** pane (see "Preflight Results and Messages" on page 1248 and "Designer User Interface" on page 1128).

Methods

These are the methods of the logger object.

Method	Parameters	Description
error()	message: string	Logs an error message
info()	message: string	Logs an informational message
warn()	message: string	Logs a warning message

merge

In Control Scripts, the root level instance of the object `merge` is the entry point from where you can query and change the way contexts are merged. It gives access to the template with all its contexts and sections.
For more information about Control Scripts, see "Control Scripts" on page 1098 and "Control Script API" on page 1591.
Some of the objects are also useful in Post Pagination Scripts; see "Post Pagination Scripts" on page 1114 and "Post Pagination Script API" on page 1619.
For sample scripts, follow the links to the respective objects.

Field	Type	Description
"Channel" on page 1616	Channel	The final output channel: EMAIL, PRINT or WEB. The channel doesn't change when the output consists of different contexts. When generating an email, for example, the channel is EMAIL, even when merging the Print context to attach it to the email.
"context" on page 1620	Context	The context rendered by this merge run. If for one record, different contexts need to be output (for example, when the Print context is attached to an email) a record is merged multiple times: once per context. Per merge run, `merge.context` shows with which context the record is merged.
"Pagination" on page 1636	Pagination	Contains the total page count and sheet count of all sections in the Print context after pagination.
"section" on page 1627	Section	In Standard Scripts, this object defines the section that is being merged. **Note!** In Control Scripts, `merge.section` is **only** available when the output channel is **WEB**. To make sure that it is defined, use the following statement: `if (merge.channel == Channel.WEB && merge.context.type == ContextType.WEB) { ... }`. To retrieve any section in a Control Script, use: `merge.template.contexts.ContextType.Section['Section name'];` (for example: `merge.template.contexts.PRINT.sections["Section EN"]`). In Post Pagination Scripts, only Print sections are available.
"template" on page 1613	Template	This object contains the template and all of its contexts. It can be used to find out which contexts are available in the template, using `merge.template.contexts` (see "context" on page 1620) and to manipulate the sections in those contexts (see "section" on page 1627).

template

The `template` object represents the template with all its contexts and sections. It is used frequently in Control Scripts (see "Control Scripts" on page 1098 and "Control Script API" on page 1591) but it can also be used in Standard Scripts.

It is retrieved via the merge object: `merge.template` (see "merge" on page 1635).

Which contexts are available in the template can be queried using `merge.template.contexts`. To get access to a specific context, you have to specify the ContextType (see "ContextType" on page 1617).

Field	Type	Description
contexts	Array	Array of contexts (see "context" on page 1620) available in the template. The contexts contain the sections (see "section" on page 1627).
images	Array	The list of image resources (see "ImageResource" on page 1617) included in the template.
masterpages	Array	Array of Master Pages (see "masterpage" on page 1594) available in the template.
"media" on page 1597	Array	Media available to this template (see "Media" on page 652). For each of them you can specify, enable and position the stationery's front and back.
parameters	Parameters	Runtime parameters for content creation. See "Runtime parameters" on page 608.
"properties" on page 1614	Properties	This object contains all default properties of the template as well as any custom properties. (On the menu, select **File > Properties** to view and complement the file properties. See "File Properties dialog" on page 1150).

Example

The following Control Script retrieves two Print sections. Then, depending on a value in the current record, it enables one section or the other, so that only one of the two sections appears in the output.

```
var printSections = merge.template.contexts.PRINT.sections;
printSections['Section EN'].enabled = false;
printSections['Section FR'].enabled = false;

if(record.fields.Language === 'FR'){
    printSections['Section FR'].enabled = true;
} else {
    printSections['Section EN'].enabled = true;
}
```

properties

The `properties` object inside the `template` object (see "template" on page 1613) contains all default properties of the template file as well as any custom properties.

To view and complement the file properties, select **File > Properties** on the menu. See "File Properties dialog" on page 1150.

Following are the default properties.

Field	Type	Description
application	String	Application version when the document was last saved
author	String	Original author of the document
company	String	company name
created	Date	Date and time on which the document was created
customProperties	List	A list of defined custom properties
description	String	Description of the document
file	String	File name of the document
keywords	String	Semicolon-separated list of keywords
modified	Date	Date and time on which the document was last saved

Example

This script stores a default property (author) and a custom property in variables.

```
var author = merge.template.properties.author;
var myProperty = merge.template.properties.customProperties.myPropertyName;
```

ImageResource

The `ImageResource` object represents an image resource in a template.

In script, an `ImageResource` is retrieved via the `images` Array which contains a list of all image resources that are included in the current template (see "template" on page 1613). For example: `var myImage = merge.template.images[0];`.

Functions and fields

Field	Type	Description
modified	Number	Timestamp of the last modification of the image resource, for example: 1566660504000.
name	String	The base name of the image resource, for example: "cat.jpg".
path	String	The path of the image resource, relative to the template, for example: "images/cat.jpg".

Example

The following script adds an image called 'dog.jpg' to the content of the template if the image can be found in the template's resources:

```
const image = merge.template.images.find(image => image.name == 'dog.jpg');
if (image) {
    results.after(`<img src=${image.path}>`);
}
```

masterpage

The `masterpage` object is used to set a Master Page's header and footer or to replace its entire HTML body. It can **only** be accessed in Control Scripts (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). Trying to access it in another type of script will result in an error.
Note that Master Pages are only used in a Print context (see "Master Pages" on page 648).

In script, Master Pages are retrieved via the masterpages Array which contains all Master Pages in the current template (see "template" on page 1613).
For example:

```
var myMasterpage = merge.template.masterpages["Master page 1"];.
```

Functions and fields

Field	Type	Description
"html()" on page 1612		Gets or sets the HTML of the body element.
"margins" on page 1633	Margins	Allows to set the header and footer of a Master Page, using a Measurement string, for example: "2in" (this sets a margin to two inches).

html()

In a Control Script, the `html()` function of a section or Master Page can be used to get the initial contents of its <body>, and modify them. This makes it possible, for example, to populate a section or Master Page with elements retrieved from a Content Management system, before Standard Scripts run.

html()

Gets the initial contents of the <body> of a section or Master Page.

html(value)

Replaces the contents of the <body> of a section or Master Page with the supplied value. This function is only available in Control Scripts. See also: "section" on page 1627 and "masterpage" on page 1594.

value

A String that may contain HTML tags.

Examples

The following script uses `html()` to retrieve the contents of a section and add a paragraph to it.

```
var foo = merge.context.sections["Section 1"].html();
foo += "<p>hello world</p>";
merge.context.sections["Section 1"].html( foo );
```

The following script loads a snippet based on the value of a field, and then replaces the content of a Print section with the snippet using `html()`.

```
var mySection = merge.context.sections["Section 1"];
var promoTxt = loadhtml('snippets/promo-' + record.fields['City'] + '.html');
mySection.html(promoTxt);
```

margins

The `margins` object is used to set a Print section or Master Page's margins in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).
Note that Master Pages are only used in a Print context (see "Master Pages" on page 648).

The `margins` object is retrieved via the `section` object (see "section" on page 1627) or via the `masterpages` array in a template (see "masterpage" on page 1594), respectively.

Fields

Field	Type	Description
	String	These fields allow to set the bottom, left, right and top of a Print section, using a Measurement string, for example: "2in" (this sets a margin to two inches).
	String	These fields allow to set the header and footer of a Master Page, using a Measurement string.

Examples

Setting the margins of a Print section

```
var sectionA = merge.template.contexts.PRINT.sections["Section A"];
sectionA.margins.top = "2in";
sectionA.margins.left = "2in";
sectionA.margins.right = "2in";
sectionA.margins.bottom = "2in";
```

Setting the header and footer of a Master Page

```
var masterA = merge.template.masterpages["Master page A"];
masterA.margins.header = "2in";
masterA.margins.footer = "2in";
```

media

The media object can be used to specify, enable and position a stationery's front and back in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).
Note that Media are only used in Print sections (see "Media" on page 652).

The available media are listed in, and retrieved via, the template object (see "template" on page 1613), for example:

```
var myMedia = merge.template.media.My_Media.
```

Fields

Field	Type	Description
	String	The height of the Media using a Measurement string. For example, "11in" is 11 inches. In a Control Script this property can be used to change the height of a Media. In other scripts it is only possible to get the height of the Media (read-only).
	Stationery	The Stationery's object's `front` and `back` fields are used to set the front and the back of a Media; see "front, back" on page 1597.
	String	The width of the Media using a Measurement string. For example, "8.5in" is 8.5 inches. In a Control Script this property can be used to change the width of a Media. In other scripts it is only possible to get the width of the Media (read-only).

front, back

The `front` and `back` fields of the Stationery object are used to set the front and the back of a Media (see "media" on page 1597).
Both `front` and `back` have the following fields.

Field	Type	Description
	boolean	When enabled, the stationery will be included in the output (Print sections only).
	String	Specifies the image to use as virtual stationery. For a file named My_Media.pdf, stored inside the template resources, the url would be `images/My_Media.pdf`. The complete syntax for an external file is: **file://<host>/<path>**. If the host is "localhost", it can be omitted, resulting in **file:///<path>**, for example: `file:///c:/users/Adminstrator/Pictures/My_Media.jpg`.
	Number	PDF and TIFF files may count more than one page. Specify the page number to use.
	"MediaPosition" on page 1618	The `position` is used to place the stationery on the page (absolute, centered, fit to media).
	Measurement	The vertical offset from the top of the page, used to position the stationery (only when absolute positioning is selected). This value can be negative.
	Measurement	The horizontal offset from the left of the page, used to position the stationery (only when absolute positioning is selected). This value can be negative.

Number functions

Note: The locale also influences the output of some Number functions; see "Locale" on page 921.

Tip: For fields that contain a number, you can also enter a formatting pattern via the Helper Wizard and the Text Script Wizard; see "Using the Helper Wizard" on page 1003, "Using the Text Script Wizard" on page 948, "Formatting variable data" on page 952 and "Number patterns" on the next page).

Note: In output, spaces in a number or amount of money are transformed into non-breaking spaces to ensure that it is not split onto different lines.

currency(value)

Formats a number as an amount of money. Which currency symbol and which thousands separator are used depends on the Locale; see "Locale" on page 921.

value

A number. This can be a value from a field that contains a SmallInteger, BigInteger, Float, SmallCurrency or LargeCurrency.

currency(value, pattern)

Formats a number as an amount of money using a custom pattern. Which currency symbol and which thousands separator are used depends on the Locale; see "Locale" on page 921. For available patterns, see "Number patterns" below.

value

A number. This can be a value from a field that contains a SmallInteger, BigInteger, Float, SmallCurrency or LargeCurrency.

pattern

A custom pattern that may consist of symbols; see "Number patterns" below. Note that the repetition of pattern letters plays a part in determining the exact presentation.

currencyNoSymbol(value)

Formats a number as a currency whilst omitting the currency symbol.

value

A number. This can be a value from a field that contains a SmallInteger, BigInteger, Float, SmallCurrency or LargeCurrency.

grouped(value)

Formats a number using a thousands separator. Which separator is used depends on the Locale, see "Locale" on page 921.

value

A number. This can be a value from a field that contains a SmallInteger, BigInteger, Float, SmallCurrency or LargeCurrency.

Number patterns

Numbers, used in a template and originating from a field in a record set can be displayed using a custom pattern.
You can enter the pattern in an expression (see "Format Helpers" on page 1015), or in the Helper Wizard or Text Script Wizard (see "Using the Helper Wizard" on page 1003, "Using the Text Script Wizard" on page 948 and "Formatting variable data" on page 952). Note that for this to work, in the DataMapper the field that contains the value must be set to Integer, Float, or Currency.

In the Script Editor, the pattern can be passed to a function of the `formatter`; see "formatter" on page 1486. The custom pattern may consist of pattern characters (see below), a prefix and a suffix.

Note that strings need to be converted to a number before they can be formatted this way.

The repetition of pattern letters determines the exact presentation. For example, the pattern "00000" limits the number to 5 digits and adds leading zeros to any numbers that are not 5 digits long.

> **Note:** In output, spaces in a number or amount of money are transformed into non-breaking spaces to ensure that it is not split onto different lines.

Pattern characters

Symbol	Location	Localized?	Meaning
0	Number	Text	Digit
#	Number	Year	Digit, zero shows as absent
.	Number	Year	Decimal separator or monetary decimal separator
-	Number	Month	Minus sign
,	Number	Number	Grouping separator
E	Number	Number	Separates mantissa and exponent in scientific notation. Need not be quoted in prefix or suffix.
;	Subpattern boundary	Number	Separates positive and negative subpatterns
%	Prefix or suffix	Number	Multiply by 100 and show as percentage
\u2030	Prefix or suffix	Number	Multiply by 1000 and show as per mille value
¤ (\u00A4)	Prefix or suffix	Number	Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.
'	Prefix or suffix	Text	Used to quote special characters in a prefix or suffix, for example, "'#'#" formats 123 to "#123". To create a single quote itself, use two in a row: "# o''clock".

Source: http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html.

query()

This function creates a new result set, containing the HTML elements in the *current* section that match the supplied CSS selector. The context (optional) allows you to restrict the search to descendants of one or more context elements.

The new result set is of the type `QueryResults`, just like the `results` object which is also the result of a (hidden) query. All functions that can be used with the `results` object can also be used with this result set; see "results" on page 1623.

Note: The `query()` function can't be used in a Control Script, since Control Scripts don't have access to the DOM.

Tip: To search for elements in *all* sections of the *Print* context in Post Pagination scripts, use the "query(selector)" on page 1621 function of the `context` object (see "context" on page 1620).

query(selector)

Creates a new result set containing the HTML elements in the current section that match the supplied CSS selector.

selector

A String containing a CSS selector. See MDN Web Docs - CSS Selectors and combinators for CSS selectors and combinations of CSS selectors.

Examples

Look for an element with a certain ID

This scripts applies a style rule to the queried elements.

```
query("#test1").css("color", "yellow");
```

Matched element	Matched element after script execution
foo	foo

Look for an element in a snippet

The following script loads a snippet. Then it looks up an element in a snippet and sets its text. Finally, it replaces the elements matched by the script's selector by the snippet.

```
var snippet = loadhtml('snippets/mysnippet.html');
query("#foo", snippet).text("bar");
results.replaceWith(snippet);
```

query(selector, context)

Creates a new result set containing the HTML elements that match the supplied CSS selector. The context (optional) allows you to restrict the search to descendants of one or more context elements.

selector

A String containing a CSS selector. See MDN Web Docs - CSS Selectors and combinators for CSS selectors and combinations of CSS selectors.

context

A result set (the result of another query) or an HTML string. If the passed context is not a result set or HTML string it will be coerced to a String and interpreted as HTML.

Examples

This script performs a query in the results of another query.

```
var table = query("table");
var rows = query("tr", table);
var cells = query("td", rows);
```

Since the `results` object also is the result of a query (for elements that match the selector of the script), it can be passed as context. For example in a script with the selector “table”.

```
var rows = query("tr", results);
```

query(html)

Creates a new HTML element on the fly from the provided string of raw HTML, and returns the new element.

html

A String containing a HTML element. Tags that can contain other elements should be paired with a closing tag.

Example

The following script adds a paragraph to the `results` (elements that match the selector of the script).

```
var div = query( '<div>' );
for ( let i = 0; i < 5; i++ ){
    var p = query( '<p>' );
    p.text( 'This is number ' + i );
    div.append( p );
}
results.after( div );
```

Tip: The Dynamic Attachment script uses this function to add an attachment to an Email section; see "Email attachments" on page 679.

record

The record object gives access to the record that is currently being merged with the template.

Properties

Field	Type	Description
	Object	Shortcut to access a field in the fields array. For example, you can use `record.FirstName` to access `record.fields.FirstName`. **Note:** If a record has a field and a table of the **same name**, `record.<fieldName>` will return the table.
	Array	The field values that belong to this record.
	Number	The id of this record.
	Number	The one-based index of this record, or zero if no data is available.
	Array of records	Shortcut to access a table in the tables array. For example, you can use `record.Products` to access `record.tables.Products`.
	Array	The detail tables that belong to this record.

Accessing fields and tables

You can access a specific **field** value:

- using the field's name: `record.fieldName` (case sensitive)
- via the `fields` array, by name: `record.fields.fieldname` or `record.fields['fieldname']` (case insensitive)

If there is no field with the specified name, `record.fields.fieldname` returns an empty string, whereas `record.fieldname` will return `undefined` (unless there is a table with the same name, in which case the table is returned).

A **detail table** can be accessed using its name: `record.tableName`, or via the `tables` array. Either way the table name should be followed by a numeric index for a record in that detail table.

For example, to access the value of the field "prod_id" in the first record of a detail table called "detail", you can use:

```
record.detail[0].fields.prod_id or
record.tables["detail"][0].fields.prod_id or
record.tables.detail[0].fields.prod_id or
record.detail[0].prod_id
```

If there is no table (and no field) with the specified name, `record.tableName` results in `undefined`.

In order to **loop** over records in a detail table you could use a for(... in ...) loop (see "for(... in ...)" on page 1485), for example:

```
var records = record.tables.detail;
for (var i in records) {
    var rec = records[i];
    ...
}
```

Alternatively you could use an 'Each matched element' script (see "Setting the scope of a script" on page 1071).

Yet another way is to create a standard `for` loop using the table's `length` property:

```
var records = record.tables.detail;
for (var i = 0; i < records.length; i++) {
    var rec = records[i];
    ...
}
```

Examples

record.fields

The following Standard Script evaluates the data field *Country* in the current record. If the value is 'CANADA' it will show the results, otherwise it will hide them. (The results object contains the elements that match the script's selector; see "results" on page 1623 and "Writing your own scripts" on page 1064.)

```
if (record.fields["Country"] == "CANADA") {
    results.show();
} else {
    results.hide();
}
```

Instead of `record.fields["Country"]` you could write `record.Country` (case sensitive), if there is no detail table with the same name.

In a Control Script, an entire section could be enabled or disabled based on the same condition:

```
if (record.Country == "CANADA") {
    merge.template.contexts.PRINT.sections["Section 1"].enabled = true;
} else {
    merge.template.contexts.PRINT.sections["Section 1"].enabled = false;
}
```

(For more information about Control Scripts, see "Control Scripts" on page 1098.)

record.tables

The next script looks up a value in the first record in a detail table called "detail" and shows or hides the results depending on that value.

```
if (record.tables.detail[0].fields.prod_id == "10") {
    results.show();
} else {
    results.hide();
}
```

Note that indexes start counting at 0, so `tables.detail[0]` refers to the first record in the detail table.

results

The `results` object (type: `QueryResults`) is the result of the query for HTML elements that match the selector of the script. The selector of a script can be specified in the Script Editor and is visible in the second column of the Scripts pane, next to the name of the script.

If, for example, a script would have the selector `p.onlyCanada`, the script would apply to all paragraphs that have the class `onlyCanada`. (Classes can be defined in the **Attributes** pane at the right: select the element in the content and type the class(es) in the **Class** field.)
The script could then use the `results` object to hide or show those paragraphs, depending on the value of the data field `Country` in the current record:

```
if (record.fields["Country"] == "CANADA") {
results.show();
} else {
results.hide();
}
```

Note: This object can't be used in Control Scripts, because they don't have a selector.

Tip: The easiest way to access the elements in a result set one by one, is by setting the scope of the script to "Each matched element" and using the this object (see "this" on page 1554).

Property

Field	Type	Description
	Number	**Number of elements in this result set. Equivalent to calling size().**

Functions

The functions below can be called by the `results` object and by any other result set that is returned by a query, see "query()" on page 1513.

Function	Description
"add()" on page 1557	Adds elements to a set of HTML elements.
"addClass()" on page 1559	Adds the specified class to each element in a set of HTML elements. Has no effect if the class is already present.
"after()" on page 1560	Inserts content after each element in a set of HTML elements..
"append()" on page 1562	Inserts content at the end of each element in a set of HTML elements.
"attr()" on page 1565	Change the given attribute of the element or set of HTML elements with the given value.
"before()" on page 1566	Inserts content before an element or before each element in a set of HTML elements.
"children()" on page 1568	Returns the immediate children of an HTML element.
"clone()" on page 1611	Returns a new result set containing a copy of each element in a set of HTML elements.
"closest()" on page 1570	For each element in a set, this function gets the first parent element that matches a selector, by testing the element itself and traversing up through its ancestors in the DOM tree.

Function	Description
"css()" on page 1571	Gets the value of a style property for the first element in set of HTML elements or sets one or more CSS properties for every element in a set of HTML elements.
"empty()" on page 1573	Removes the contents (child elements and inner HTML) from one element or a set of elements in the template.
"filter()" on page 1536	Returns a subset of the current result set.
"find()" on page 1574	Performs a search for a text in the children of each element in a set of HTML elements, and returns a new result set with elements that surround the occurrences.
"get(index)" on page 1538	Returns the element (type: QueryResult) found at the supplied index in a set of HTML elements.
"hasClass()" on page 1574	Returns `true` if the first element in this result set has the specified class.
"height()" on page 1575	Gets or sets the outer height of an element, including padding and borders.
"hide()" on page 1576	Hides the HTML element or set of HTML elements.
"html()" on page 1576	Replaces the inner HTML of the element or of each element in a set of HTML elements with the supplied value, or returns the HTML of the first element if no value is supplied.
"info()" on page 1626	**Post Pagination Scripts only**. Returns pagination information for the first element in this result set.
is(selector)	Returns true if at least one of the elements in a set of HTML elements matches the supplied CSS selector.
"next()" on page 1577	Returns the next sibling of each HTML element in the result set.
"overflows()" on page 1626	Returns a boolean value indicating whether an HTML element overflows its box boundaries.
"pageref()" on page 1580	Returns a **marker** that will be replaced with the element's page number after pagination. This only works for elements in the section that is currently being merged.
"parent()" on page 1580	Returns the parents of the elements in a set of HTML elements.
position()	Returns information about the position (see NodePosition) of the first element in this result set, excluding margins. To include margins, call `position(true)`.

Function	Description
"prepend()" on page 1581	Inserts content at the beginning of an HTML element or of each element in a set of HTML elements.
"prev()" on page 1584	Returns the previous sibling of each HTML element in the result set.
"remove()" on page 1585	Removes an HTML element or a set of HTML elements from the document.
"removeAttr()" on page 1586	Removes the specified attribute from each element in this result set.
"removeClass()" on page 1587	Removes the specified class from an element or from each element in a set of HTML elements. Has no effect if the class is not present.
"replaceWith()" on page 1587	Replaces an HTML element or a set of HTML elements (with a snippet, for example). Returns the result set.
"show()" on page 1588	Shows the HTML element or a set of HTML elements.
size()	Gets the number of elements in this result set. Equivalent to the `length` property.
"tagName()" on page 1588	Returns the HTML tag name of the first element in this result set, in uppercase. For an example see: "Creating a Table Of Contents" on page 1116.
"text()" on page 1589	Replaces the text content of an HTML element or of each element in a set of HTML elements with the supplied value, or returns the text content of the first element if no value is supplied.
"width()" on page 1590	Gets or sets the outer width of an element, including padding and borders.

add()

The add() function adds elements to one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

add(content)

The add() function adds HTML content to one or more HTML elements and returns the union of this result or result set and other content.

content

A query result. This can be an HTML string or a result set.

Examples

This script adds one query result to another and sets the background color to yellow.

```
query("#test1").add(query("#test2")).css("background", "yellow");
```

Note: The way the functions add() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the results of the queries in a variable:

```
var myResult = query("#test1");
myResult.add(query("#test2");
myResult.css("background", "yellow");
```

The following script loads snippets in an iteration and adds their elements to an empty result set (using `query()`). Then it replaces a placeholder in the template with the new result.

```
var chapters = query();
for ( var i = 1; i <= 4; i++) {
chapters = chapters.add(loadhtml('snippets/Chapter' + i + '.html'));
}
results.replaceWith(chapters);
```

Selector	Matched element	Matched element after script execution
#chapters	{{chapters}}	Chapter 1 Lorem ipsum… Chapter 2 Lorem ipsum… Chapter 3 Lorem ipsum… Chapter 4 Lorem ipsum…

addClass()

Adds the specified class(es) to one ore more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

Adding a class has no effect if the class is already present.

addClass(classname)

Adds the specified class(es) to one or more HTML elements. This has no effect if the class is already present.

classname

String, space separated list of class names.

Examples

This script adds a class name to a paragraph.

```
results.addClass("foo");
```

Selector	Matched element	Matched element after script execution
p	Hello world	Hello world

The following script adds two class names to a paragraph.

```
results.addClass("foo bar");
```

Selector	Matched element	Matched element after script execution
p	Hello world	Hello world

after()

Inserts content after one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "before()" on page 1566.

after(content)

Inserts content after one or more HTML elements and creates a new result set.

content

String, HTML string or result set to insert after the matched elements. In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script looks up an element with the ID #salesrep and inserts a paragraph after it.

```
query("#salesrep").after("<p>Lorem ipsum</p>");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem ipsum

This script looks up an element with the ID #salesrep, sets its text color to red and inserts a paragraph after it.

```
query("#salesrep").after("<p>Lorem ipsum</p>").css("color","red");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem ipsum

Note that the way the functions after() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var salesrep = query("#salesrep");
salesrep.after("<p>Lorem ipsum</p>");
salesrep.css("color","red");
```

The following script inserts a paragraph after the elements in the results (the set of HTML elements that match the selector of the script).

```
results.after("<p>Lorem Ipsum</p>");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem ipsum

This script looks for the string "Lorem " in the results (the set of HTML elements that match the selector of the script) and inserts the string "ipsum" right after that text. The string is automatically enclosed in a span.

```
results.find("Lorem ").after("ipsum");
```

Matched element	Matched element after script execution
Lorem dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

This script looks up an element with the ID #salesrep and inserts a string after it. The string is automatically enclosed in a span.

```
query("#salesrep").after("Lorem Ipsum");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem Ipsum

append()

Insert content at the end of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "prepend()" on page 1581.

append(content)

Inserts content as the last element at the end of one or more HTML elements. `Append()` creates a new result set.

content

String, HTML string or result set to insert after the element(s). In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script appends a paragraph to the `results` (the set of HTML elements that match the selector of the script).

```
results.append("<p>Peter Parker</p>");
```

Selector	Matched element	Matched element after script execution
#box	 Personal information 	 Personal information Peter Parker

This script appends a string to the `results` (the HTML elements that match the selector of the script). The string is added to the end of the matched element(s) and wrapped in a Span element.

```
results.append("Peter Parker");
```

Selector	Matched element	Matched element after script execution
.name	 Personal information Name: 	 Personal information Name: Peter Parker

This script's selector is , so the script appends a paragraph to all Div elements in the template.

```
results.append("<p>Peter Parker</p>");
```

Selector	Matched element	Matched element after script execution
div	 Personal information Personal information 	 Personal information Peter Parker Personal information Peter Parker

The following script appends a snippet to a Div element with the ID `box`.

```
var a = loadhtml('snippets/snippet_name.html');
results.append(a);
```

Selector	Matched element	Matched element after script execution
#box	 Personal information 	 Personal information Peter Parker

This script looks for an element with the ID `box` and appends a paragraph to it.

```
query("#box").append("<p>Peter Parker</p>");
```

Matched element	Matched element after script execution
  Personal information 	 Personal information Peter Parker

This script looks for an element with the ID `box`, appends a paragraph to it and colors all text inside the box red.

```
query("#box").append("<p>Peter Parker</p>").css("color","red");
```

Matched element	Matched element after script execution
  Personal information 	 Personal information Peter Parker

Note: The way the functions append() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var box = query("#box");
box.append("<p>Peter Parker</p>");
box.css("color","red");
```

attr()

attr(attributeName) : String

Returns the value of the specified attribute of an HTML element which can be:

- The first element in a set of elements that match the selector of a script (see "results" on page 1623).

- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The first element in a set of elements returned by a query in the template (see "query ()" on page 1513).

attributeName

String; the name of the attribute.

Examples

This script - with the selector `img` - stores the source of the first image in a variable.

```
var src = results.attr("src");
```

The following script looks up an image with the ID #image1 and stores its background color in a variable.

```
var imgURL = query("#image1").attr("src");
```

attr(attributeName, value)

Sets the value of the specified attribute of one HTML element or of each element in a result set.

attributeName

String; the name of the attribute.

value

String; value for the attribute.

Examples

This script looks up an image in an element with the ID #calloutbox and sets its alternative text to a value from a data field.

```
var altText = record.fields.FavHobby;
query("#callout img").attr('alt', altText);
```

The following script sets the background color of a specific table cell in an email to red if the value of the field TOTAL has a negative value in the current record.

```
if(record.fields.TOTAL<0) {
    query("#total").attr("bgcolor","red");
}
```

before()

Inserts content before one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "after()" on page 1560.

before(content)

Before(content) inserts content before one or more HTML elements and creates a new result set.

content

String, HTML string or result set to insert after the elements. In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script looks for an element with the ID `salesrep`and inserts a paragraph before that element.

```
results.before("<p>Lorem Ipsum</p>");
```

Selector	Matched element	Matched element after script execution
#salesrep	Peter Parker	Lorem ipsum Peter Parker

This script does the same, but it uses the query() function to look up the element.

```
query("#salesrep").before("<p>Lorem ipsum</p>");
```

Matched element	Matched element after script execution
Peter Parker	Lorem ipsum Peter Parker

The following script looks for an element with the ID salesrep, inserts a paragraph before that element and colors that element red.

```
query("#salesrep").before("<p>Lorem ipsum</p>").css("color","red");
```

Matched element	Matched element after script execution
Peter Parker	Lorem ipsum Peter Parker

Note: the way the functions `before()` and `css()` are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var salesrep = query("#salesrep");
salesrep.before("<p>Lorem ipsum</p>");
salesrep.css("color","red");
```

The following script searches the results for the string "ipsum" and puts "Lorem " before it. "Lorem " is automatically wrapped in a Span element.

```
results.find("ipsum").before("Lorem ");
```

Matched element	Matched element after script execution
ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

The following script looks for an element with the ID salesrep and inserts the text "Lorem Ipsum" before that element. "Lorem Ipsum" is automatically wrapped in a Span element.

```
query("#salesrep").before("Lorem Ipsum");
```

Matched element	Matched element after script execution
ipsum dolor sit amet, consectetur adipiscing elit.	Lorem Ipsum Peter Parker

children()

Returns the immediate children (inner HTML) of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

Examples

This script retrieves the inner HTML of an element selected from a snippet.

```
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
results.append(snippet);
```

The following script retrieves the inner HTML of the elements and then performs a find/replace.

```
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
snippet.find('@firstname@').text('foobar');
results.append(snippet);
```

clone()

This function returns a copy of one HTML element or of a set of HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To duplicate an existing template element, clone it before calling `append()`; see "append()" on page 1562.

This function can also be used to output a variable number of sections in each document, based on data. See also: "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Examples

This script performs an iteration over the elements in the `results` (the elements that match the selector of the script).

```
var row = query("tbody tr", results).clone();
query("tbody", results).append(row);
```

The following script clones an existing table row to match the number of rows in a detail table. Then it iterates over the rows to populate the fields.

```
// Create the number of rows based on the records in the detail table
// We start at 1 so the boilerplate row is used too and there is no need to delete that row
for(var r = 1; r < record.tables.detail.length; r++) {
results.parent().append(results.clone());
}

// Iterate over the rows and populate them with the data from the accompanying data row
query("#table_2 > tbody > tr").each(function(i) {
this.find('@ItemNumber@').text( record.tables.detail[i].fields["ItemNumber"]);
this.find('@ItemOrdered@').text( record.tables.detail[i].fields["ItemOrdered"]);
this.find('@ItemTotal@').text( record.tables.detail[i].fields["ItemTotal"]);
this.find('@ItemDesc@').text( record.tables.detail[i].fields["ItemDesc"]);
this.find('@nr@').text(i);
});
```

The following script clones and populates a boilerplate row. Once completed you will need to hide the boilerplate row.

```
for(var i = 0; i < record.tables.detail.length; i++) {

var row = results.clone(); //Clone our boilerplate row

row.find('@ItemNumber@').text( record.tables.detail[i].fields["ItemNumber"]);
row.find('@ItemOrdered@').text( record.tables.detail[i].fields["ItemOrdered"]);
row.find('@ItemTotal@').text( record.tables.detail[i].fields["ItemTotal"]);
row.find('@ItemDesc@').text( record.tables.detail[i].fields["ItemDesc"]);
row.find('@nr@').text( i );

results.parent().append(row);
}

// Hide our boilerplate row (note that this doesn't really delete the row).
results.hide();
```

closest()

This function gets the first parent element that matches a selector, by testing the element itself and traversing up through its ancestors in the DOM tree. (In HTML, a parent is an element that contains another element.)

The function can be used for:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To get a child element or all child elements, use `children()` (see "children()" on page 1568).

The `closest()` command is based on the `closest()` command found in the jQuery library: https://api.jquery.com/closest/.

To get an element's sibling element, you can use "prev()" on page 1584 or "next()" on page 1577.

closest(selector)

For one HTML element or for each element in a set, this function gets the first element that matches the selector by testing the element itself and traversing up through its ancestors in the DOM tree.

selector

A String containing an HTML tag (without the angle brackets, <>).

Examples

The following script looks up all table rows in the template that contain an <input> element.

```
query("input").closest("tr");
```

This code gets the closest 'parent' row for each element that matches the selector of the script (collected in the `results` object):

```
results.closest("tr");
```

The rows could be colored red within the same statement:

```
results.closest("tr").css('background-color','red');
```

css()

Gets the value of a style property of one HTML element, or sets one or more CSS properties of one or more HTML elements.

css(styleName) : String

Returns the value of the specified CSS property of an HTML element, which can be:

- The first element in a set of elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The first element in a set of elements returned by a query in the template (see "query ()" on page 1513).

propertyName

String; the name of the CSS property.

Examples

This script stores the text color of the `results` (the HTML elements that match the selector of the script) in a variable.

```
var textcolor = results.css("color");
```

The following script looks up an element with the ID #calloutbox and stores its background color in a variable.

```
var backgroundcolor = query("#calloutbox").css("background-color");
```

css(styleName, value)

Function to set a CSS property of one HTML element or of each element in a result set.

propertyName

String; the name of the CSS property.

value

String; value for the CSS property or a map of property-value pairs to set.

Examples

This script looks up an element with the ID #calloutbox and sets its text color to red.

```
query("#callout p").css('color' , 'red');
```

The following script does the same, but it only sets the text color to red if in the current record the value of the field 'accounttype' is 'PRO'.

```
if(record.fields.accounttype == "PRO") {
query("#callout p").css("color","red");
}
```

This script sets the text color of the results to a hexadecimal color code.

```
results.css('color' , '#669900');
```

This script loads a snippet into a variable. Then it finds/replaces text in the snippet and applies a css property to the replacing text.

```
var mysnippet = loadhtml('snippets/snippet vars.html');
mysnippet.find('@var@').text('OL Connect').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

css(properties)

Function to set one or multiple CSS properties of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

properties

Array; map of property-value pairs to set.

Examples

This script colors the text of the `results` (the set of HTML elements that match the selector of the script) red and makes it bold.

```
results.css({'color' : 'red', 'font-weight' : 'bold'});
```

empty()

Removes the contents (child elements and inner HTML) from one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

Use "remove()" on page 1585 to remove the elements themselves.

Example

This script empties all Span elements found in the template.

```
results.empty();
```

Selector	Paragraph before script execution	Paragraph after script execution
span	Lorem ipsum -dolor sit amet, consectetuer adipiscing elit.	Lorem ipsum amet, consectetuer adipiscing elit.

filter()

filter(callback)

Returns a subset of a set. All elements for which the callback function returns `true` will be included in the result.

callback

A function used as a test for each element in the set. Filter() passes the iteration index and the current element to the callback function. In the scope of the callback function, `this` refers to the current element.

Example

The selector of the following script is **li** (list item), so the `results` object contains all list items in the template. The scripts filters the third and sixth line items from the `results`, taking advantage of the index that is passed to the filter function, and colors them red. It uses the modulus operator (%) to select every item with an index value that, when divided by 3, has a remainder of 2. (The index starts counting at zero.)

```
results.filter(function(index) {
        return index % 3 === 2;
}).css( "background-color", "red" );
```

filter(selector)

Returns a subset of a set. All elements matching the selector will be included in the result.

The difference between `results.filter(selector)` and `query(selector, results)` is that query() searches throughout the entire `results` while filter() only takes the top-level elements into account.

selector

A String containing a CSS selector. See MDN Web Docs - CSS Selectors and combinators for CSS selectors and combinations of CSS selectors.

Example

The selector of the following script is **tr** (table row), so the object `results` contains all rows in the template. The scripts filters all even rows from the `results` and colors them red.

```
results.filter(":nth-child(even)").css("background-color", "red");
```

find()

find(textToFind)

Performs a deep search for textToFind in the children of each element, and returns a new result set with elements that surround the occurrences.

textToFind

A String that contains the search text.

Example

The following piece of code loads a snippet, then looks for placeholders using find(), and replaces them with a text.

```
var mysnippet = loadhtml('snippets/snippet.html');
mysnippet.find('@var1@').text('OL Connect 1');
mysnippet.find('@var2@').html('<i>OL Connect 2</i>').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

get(index)

Returns the element found at the supplied index in a set of HTML elements.

index

Place of the element in the result set. The index starts counting at zero.

Example

This Post Pagination Script copies the page numbers of level 1 and 2 headings in all Print sections to the corresponding entry in a table of contents.

```
var $numbers = query('.number');
merge.context.query("h1, h2").each(function( index ) {
    var pageNo = this.info().pageNo;
    var entry = $numbers.get( index );
    if( entry ) {
        entry.text( pageNo );
    }
});
```

This code is used in a script that inserts a table of contents. For the full script and explanation see "Creating a Table Of Contents" on page 1116.

hasClass()

hasClass(classname) : Boolean

Returns true if the HTML element or the first element in a result set has the specified class.

classname

String containing one class name.

Example

This script checks if the first of the `results` (the set of elements matching the selector of the script) has the class 'green'. If so, it colors the text of all the elements in the `results` green.

```
if (results.hasClass('green')) {
    results.css('color', 'green');
}
```

height()

Gets or sets the outer height, including padding and borders, of an HTML element, or of the first element in a result set (see "results" on page 1623), i.e. the set of HTML elements that match the selector of the script or of another query in the template (see "query()" on page 1513).

height(): Number

Returns the outer height of this element, including padding and borders, excluding margins. To include margins, call `height(true)`.

The returned value is the height in pixels, without measurement unit (e.g. 400).

height(value): void

Sets the outer height of this element, including padding and borders, excluding margins. To include margins, call `height(value, true)`.

value

Number. Give the height in pixels, but without the measurement unit (e.g. 400, not 400px).

Examples

This script adds 20 pixels to the width and height of an element.

```
var h = results.height();
var w = results.width();

h = h + 20;
w = w + 20;

results.height( h );
results.width( w );
```

The following script does the same, but in this case, margins are included.

```
var h = results.height(true);
var w = results.width(true);

h = h + 20;
w = w + 20;

results.height( h, true);
results.width( w, true);
```

hide()

Hides one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

This doesn't remove the elements; to make them visible again, use the function "show()" on page 1588.

These functions are used by the Conditional Script Wizard, as you can see when you open a Conditional Script and click the **Expand** button; see "Showing content conditionally" on page 958.

Example

This script hides or shows the elements matched by the selector of the script (which are stored in the `results` object), depending on the value of the data field `Country` in the current record.

```
if (record.fields["Country"] == "CANADA") {
results.show();
} else {
results.hide();
}
```

html()

html() : String

Returns the inner HTML of an HTML element, which can be:

- The first element in a set of elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The first element in a set of elements returned by a query in the template (see "query()" on page 1513).

html(value)

Replaces the inner HTML of an HTML element or of each element in a result set with the supplied value.

value

A String that may contain HTML tags.

Examples

The following script loads part of a snippet based on the value of a field, and then inserts the content into the document using `html()`.

```
var promoTxt = loadhtml('snippets/promo-en.html', '#' + record.fields['YOGA']);
results.html(promoTxt);
```

The following script loads a snippet. Then it looks for a placeholder (@var2@) in the text of that snippet and replaces every found placeholder by the text 'OL Connect 1'. It uses html() so the HTML formatting (and) will indeed be interpreted as HTML. Finally, it places the snippet in the template.

```
var mysnippet = loadhtml('snippets/snippet.html');
mysnippet.find('@var1@').html('<i>OL Connect 1</i>');
results.replaceWith(mysnippet);
```

info()

Returns pagination information for one HTML element or for the first element in a result set (see "results" on page 1623) of a query across all sections in a Print context (see "query (selector)" on page 1621).

The returned information is of the type `PaginationInfo` (see "PaginationInfo" on page 1622).

This function can only be used in a Post Pagination Script ; see "Post Pagination Script API" on page 1619.

For an example see: "Creating a Table Of Contents" on page 1116.

next()

`next()` returns the next sibling of an HTML element, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

In HTML, siblings are HTML elements that share the same parent, i.e. HTML elements in the same container element.

Note that a sibling can be a different type of element. For example, if a paragraph, an image and a table follow each other in the same Box, they are siblings.

> **Tip:** To get the **previous** sibling, use previous() (see "prev()" on page 1584).

What if there are no siblings?

If you call `prev()` or `next()` on a **collection** of elements (either "results" on page 1623 or the result set of "query()" on page 1513), the function will return a collection of siblings. If none of the elements has a previous or next sibling, the collection will be empty. Actions performed on an empty collection will simply have no effect.

If you call `prev()` or `next()` on a **single** element (this in an 'Each matched element' script), and the element has no previous or next sibling, the function returns `null`. Performing an operation on a `null` value will result in an error, which means that the rest of the script won't be executed. In this case, it is useful to check the return value, unless it's certain that it will never be `null`.

Example

Assume that there are three consecutive paragraphs in a Box and that the second of those paragraphs has an ID that matches the selector of this script. The paragraph is stored in the results object (see "results" on page 1623). The following script changes the font-weight of the next paragraph - the third paragraph in the Box - to bold.

```
results.next().css("font-weight", "bold");
```

If the elements in results don't have siblings, this line of code will have no effect. It also won't result in an error.

In an 'Each matched element' script you would first need to check the return value of this.next():

```
var nextSibling = this.next();
if nextSibling != null { nextSibling.css("font-weight", "bold"); }
```

overflows()

The `overflows()` method returns a boolean value indicating whether an HTML element overflows its box boundaries.

This function could for instance be used in a design where content needs to flow separately from the main text flow, or where a new, full-page, multi-column box should be inserted when the current one overflows.

Note that when called in a Standard script, this method runs *before* the pagination of Print output. After pagination - in the output, and in Preview mode - the element's overflow may have changed, especially when the element is combined with floating elements (i.e. elements that have the CSS style float) and located near a page-break.
To prevent this from happening, it may help to set the element's CSS style to `overflow:hidden`.
The `overflows()` method can also be used in a Post Pagination script.

Tip: You don't need a script to resize text in order to make it fit in a box. The Copy Fit feature automatically scales text to the available space (see "Copy Fit" on page 896).

Examples

```
var $box = query("#mybox");
while ( ! $box.overflows()) {
    //do something
}
```

A slightly shorter version:

```
while ( ! query("#mybox").overflows()) {
    //do something
}
```

The following script selects three boxes that all have the class box (selector: .box). It loops over them, inserting as many products from an array as possible into the box. In fact, it inserts one too many and removes the last one before moving on to the next box.

```
var products = ["Appetizer","Beans","Beef","Lettuce","Sprouts","Coconut","Juice","Soup","Coriander","Cheese","Pasta","Sugar","Vinegar","Bread"];
var i = 0;
results.each( function() {
    // Add elements to the box until it overflows.
    while ( ! this.overflows() && i < products.length ) {
        this.append("<p>" + products[i] + "</p>");
        i++;
    }
    // Go one step back unless we processed all items in our array.
    if( i < products.length ) {
        query("p:last-child", this).remove();
        i--;
    }
});
```

pageref()

Returns a **marker** that will be replaced with the element's page number after pagination. This only works for elements in the section that is currently being merged.

Example

Creating a table of contents

The following script creates a table of contents for all level 1 headings (<h1> elements) with the class `title` in one section.

```
var toc = '<ul ID="toc">';
query('h1.title').each(function()
{toc += '<li>' + this.text() + ' <span class="li_toc">' + this.pageref() + '</span></li>';
});
toc += '</ul>';
results.after(toc);
```

The first line creates a variable for the table of contents, which will be a list (a <ul> element with the ID `toc`). The start tag of the list is added to the variable.
The next line does a query for all level 1 headings (<h1> elements) with the class `title` in the current section. With `each()` the script loops through them. For each of the headings it adds a line item to the list, with the text (`this.text()`) and the page reference of the respective heading.
After the loop, the end tag of the list is added to the variable.
Finally, the script adds the variable - that now contains the table of contents - after the `results`. The `results` object contains the elements that match the selector of the script. So, if the script's selector selects the title of the table of contents, the table of contents will be added after that.

The following style rules, added to the style sheet, will align the chapter titles to the left and the page numbers to the right:

```
#toc li {
text-align:left;
}
#toc span {
float: right;
}
```

Note that these styles use the list's ID, that was defined in the first line of code. For information about style sheets, see "Styling templates with CSS files" on page 881.

parent()

Returns the parent(s) of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

In HTML, a parent is an element that contains another element.
To get an ancestor that matches a particular selector, use `closest()` (see "closest()" on page 1570).

To get an element's sibling element, you can use "prev()" on page 1584 or "next()" on page 1577.

Example

Assume that there are three paragraphs in a Box and that one of those paragraphs matches the selector of this script. The paragraph is stored in the `results` object (see "results" on page 1623). The following script retrieves the Box (which is the parent of the paragraph) using `results.parent()`, and then changes its background color to red.

```
results.parent().css('background-color' , 'red');
```

prepend()

Insert content at the beginning of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "append()" on page 1562.

prepend(content)

Insert content as the first element to each element in the set of HTML elements that match the selector of the script or of another query in the template (see "query()" on page 1513). `Append` creates a new result set.

content

HTML string, string or HTML string to insert after the matched elements. In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script inserts a heading as the first element in an element that has the ID `#box`.

```
results.prepend("<h1>Personal information</h1>");
```

Selector	Matched element	Matched element after script execution
#box	 Peter Parker 	 Personal information Peter Parker

This script inserts a heading as the first element in an element that has the class `name`.

```
results.prepend("<b>Name: </b>");
```

Selector	Matched element	Matched element after script execution
.name	 Personal information Peter Parker 	 Personal information Name: Peter Parker

This script inserts content in multiple elements at the same time.

```
results.prepend("<h1>Personal information</h1>");
```

Selector	Matched element	Matched element after script execution
div	 Peter Parker Peter Parker 	 Personal information Peter Parker Personal information Peter Parker

This script prepends a snippet that contains the text "<h1>Personal information</h1>".

```
var a = loadhtml('snippets/snippet.html');
results.prepend(a);
```

Selector	Matched element	Matched element after script execution
div	 Peter Parker 	 Personal information Peter Parker

This script uses the function query() to find a box. Then it inserts a heading as the first element in that box.

```
query("#box").prepend("<h1>Personal information</h1>");
```

Matched element	Matched element after script execution
  Peter Parker 	 Personal information Peter Parker

This script uses the function query() to find a box, prepends a heading and sets the text color of the entire box to red.

```
query("#box").prepend("<h1>Personal information</h1>").css("color","red");
```

Matched element	Matched element after script execution
  Peter Parker 	 Personal information Peter Parker

Note: The way the functions prepend() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var box = query("#box");
box.prepend("<p>Peter Parker</p>");
box.css("color","red");
```

prev()

`prev()` returns the previous sibling of an HTML element, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

In HTML, siblings are HTML elements that share the same parent, i.e. HTML elements in the same container element.
Note that a sibling can be a different type of element. For example, if a paragraph, an image and a table follow each other in the same Box, they are siblings.

Tip: To get the **next** sibling, use `next()` (see "next()" on page 1577).

What if there are no siblings?

If you call `prev()` or `next()` on a **collection** of elements (either "results" on page 1623 or the result set of "query()" on page 1513), the function will return a collection of siblings. If none of the elements has a previous or next sibling, the collection will be empty. Actions performed on an empty collection will simply have no effect.

If you call `prev()` or `next()` on a **single** element (this in an 'Each matched element' script), and the element has no previous or next sibling, the function returns `null`. Performing an operation on a `null` value will result in an error, which means that the rest of the script

won't be executed. In this case, it is useful to check the return value, unless it's certain that it will never be `null`.

Example

Assume that there are three consecutive paragraphs in a Box and that the second of those paragraphs has an ID that matches the selector of this script. The paragraph is stored in the results object (see "results" on page 1623). The following script changes the font-weight of the previous paragraph - the first paragraph in the Box - to bold.

```
results.prev().css("font-weight", "bold");
```

If the elements in results don't have siblings, this line of code will have no effect. It also won't result in an error.

In an 'Each matched element' script you would first need to check the return value of `this.prev()`:

```
var prevSibling = this.prev();
if prevSibling != null { prevSibling.css("font-weight", "bold"); }
```

remove()

Removes the current element or each element in a set from the DOM.

This function returns a new result set containing each removed element. These can be changed and inserted in the document. This could be beneficial in terms of performance, as manipulating elements inside the DOM is relatively time consuming.

Examples

This script removes all Span elements found in the template.

```
results.remove();
```

Selector	Paragraph before script execution	Paragraph after script execution
span	Lorem ipsum -dolor sit amet, con-sectetuer adipiscing elit.	Lorem ipsum amet, con-sectetuer adipiscing elit.

The selector of the following sample script is `tbody`. Before this script runs, the table body consists of a single placeholder row with three cells. After running the script, it contains thirty rows. To improve performance, most of the DOM manipulation takes place on detached elements.

```
// Detach the placeholder row from the DOM
var row = query("tr", results).remove();

// Modify the cells of this row
var cells = row.children();
cells[0].html("some text").css("background-color", "yellow");
cells[1].html("some text").css("font-weight", "bold");
cells[2].html("some text");

// Create a number of copies
var rows = row.clone();
for (var i = 0; i < 30; i++) {
    rows = rows.add(row.clone());
}

// Attach all copies to the DOM as children of tbody
results.append(rows);
```

removeAttr()

Removes the specified HTML attribute from an element or from each element in a set of elements. To add or change an attribute, use `attr()` (see "attr()" on page 1565).

removeAttr(attributeName)

attributeName

String; the name of the attribute.

Examples

This script looks up an email field in a form (which is an <input> with the ID #email1) and removes its `readonly` attribute.

```
query("#email1").removeAttr('readonly');
```

removeClass()

Removes the specified class from the current HTML element or from each element in a result set. This has no effect if the class is not present.

removeClass(classname)

classname

String, space separated list of class names.

Examples

This script removes the class name "foo" from all elements in the results that have this class.

```
results.removeClass("foo");
```

Selector	Matched element	Matched element after script execution
p	Hello world	Hello world

replaceWith()

Replaces one HTML element or each element in a set of HTML elements.

replaceWith(content)

Replaces one HTML element or each element in a set of HTML elements. Returns the result set.

content

A query result. This can be an HTML string or a result set.

Examples

Replace elements with a snippet

The following script loads a snippet and then replaces the elements matched by the script's selector with the snippet.

```
var snippet = loadhtml('snippets/mysnippet.html');
results.replaceWith(snippet);
```

Replace elements with a set of snippets

The following script loads snippets and adds their elements to a new, empty result set (using `query()`). Then it replaces a placeholder in the template with the set of snippets.

```
var chapters = query();
for ( var i = 1; i <= 4; i++) {
chapters = chapters.add(loadhtml('snippets/Chapter' + i + '.html'));
}
results.replaceWith(chapters);
```

show()

Shows one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To hide elements (again), use the function "hide()" on page 1576.

These functions are used by the Conditional Script Wizard, as you can see when you open a Conditional Script and click the **Expand** button; see "Showing content conditionally" on page 958.

Example

This script hides or shows the elements matched by the selector of the script (which are stored in the `results` object), depending on the value of the data field `Country` in the current record.

```
if (record.fields["Country"] == "CANADA") {
results.show();
} else {
results.hide();
}
```

tagName()

Returns the HTML tag name of one HTML element in uppercase (e.g. H1, P, ARTICLE).

Example

This Post Pagination Script finds level 1 and 2 headings in all Print sections and stores their page number, text and HTML tag name.

```
merge.context.query("h1, h2").each(function() {
    var pageNo = this.info().pageNo;
    var text = this.text();
    var level = this.tagName();
});
```

This code is used in a script that inserts a table of contents. For the full script and explanation see "Creating a Table Of Contents" on page 1116.

text()

text() : String

Returns the text content of an HTML element or of the first element in a result set (see "results" on page 1623).

Example

This script loads a snippet into a variable and retrieves an element from the snippet using `query()` and `text()`.

```
var mysnippet = loadhtml('snippets/text-root-wrapped.html');
var subject = query("#subject", mysnippet).text();
results.append("<p style='font-weight: bold;'>" + subject + "</p>");
```

text(value)

Replaces the text content of one HTML element or of each element in a result set with the supplied value.

Example

This script loads a snippet, then looks for placeholders using find(), and replaces them using text(value).

```
var mysnippet = loadhtml('snippets/snippet.html');
mysnippet.find('@var1@').text('OL Connect 1');
mysnippet.find('@var2@').html('<i>OL Connect 2</i>').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

width()

Gets or sets the outer width, including padding and borders, of an HTML element or of the first element in result set (see "results" on page 1623), i.e. the set of HTML elements that match the selector of the script or of another query in the template (see "query()" on page 1513).

width(): Number

Returns the outer width of this element, including padding and borders, excluding margins. To include margins, call `width(true)`.

The returned value is the width in pixels, without measurement unit (e.g. 400).

width(value): void

Sets the outer width of this element, including padding and borders, excluding margins. To include margins, call `width(value, true)`.

value

Number. Give the width in pixels, but without the measurement unit (e.g. 400, not 400px).

Examples

This script adds 20 pixels to the width and height of an element.

```
var h = results.height();
var w = results.width();

h = h + 20;
w = w + 20;

results.height( h );
results.width( w );
```

The following script does the same, but in this case, margins are included.

```
var h = results.height(true);
var w = results.width(true);

h = h + 20;
w = w + 20;

results.height( h, true);
results.width( w, true);
```

this

A Standard script or Post Pagination script that has its scope set to "Each matched element" (see "Setting the scope of a script" on page 1071) will be called in a loop over the elements that match the selector of the script - the `results` (see "results" on page 1623). In

such a script, `this` is an object of the type QueryResult. It represents the current element in the loop.

> **Note:** The scope of Control Scripts can't be set, because they don't have a selector.

Example

If the selector of a script is `p.onlyCanada`, and its scope is set to "Each matched element", the script will be repeated for all paragraphs that have the `onlyCanada` class; it can access only one paragraph at a time. The script could use `this` to manipulate each paragraph, e.g. hide or show it depending on the value of a data field `Country` in the current record:

```
if (record.fields["Country"] == "CANADA") {
this.show();
} else {
this.hide();
}
```

> **Note:** In a script that has its scope set to "Each matched element", the `results` object is still accessible, and can be used in a non-iterative way (for example, you could use its `length` property to determine the total number of matched elements) but iterating over `results` will produce a warning.

Properties

Field	Type	Description
	Number	Index in the result set, or zero if this element is not part of a result set.
	Record	Represents the current detail record in a (possibly nested) detail table associated with repeating rows in a Dynamic Table. This property is only available when `this` is a row or child element of a row in a Dynamic Table. (See "Dynamic Table" on page 974.) If `this` (the current QueryResult instance) cannot be matched to a detail record, `this.record` is null.
	Number	Can be used to update a running total. It is initialized to zero when a loop over `results` starts.

Functions

The functions below can be called by the `results` object and by any other result set that is returned by a query, see "query()" on page 1513.

Function	Description
"add()" on the next page	Adds elements to the current HTML element.
"addClass()" on page 1559	Adds the specified class to the current HTML element. Has no effect if the class is already present.
"after()" on page 1560	Inserts content after the current HTML element.
"append()" on page 1562	Inserts content at the end of the current HTML element.
"attr()" on page 1565	Changes the given attribute of the current HTML element with the given value.
"before()" on page 1566	Inserts content before the current HTML element.
"children()" on page 1568	Returns the immediate children of the current HTML element.
"clone()" on page 1611	Returns a copy of the current HTML element.
"closest()" on page 1570	This function gets the first parent element that matches a selector, by testing the element itself and traversing up through its ancestors in the DOM tree.
"css()" on page 1571	Gets the value of a style property for the HTML element or sets one or more CSS properties for the HTML element.
"empty()" on page 1573	Removes the contents (child elements and inner HTML) from the current element.
"find()" on page 1574	Performs a search for a text in the children of the current HTML element and returns a new result set with elements that surround the occurrences.
"hasClass()" on page 1574	Returns `true` if the current HTML element has the specified class.
"height()" on page 1575	Gets or sets the outer height of an element, including padding and borders.
"hide()" on page 1576	Hides the HTML element.
"html()" on page 1576	Replaces the inner HTML of the element with the supplied value, or returns the HTML of the current element if no value is supplied.

Function	Description
"info()" on page 1626	**Post Pagination Scripts only**. Returns pagination information for the current HTML element.
is(selector)	Returns true if the current HTML element matches the supplied CSS selector.
"next()" on page 1577	Returns the next sibling of the current HTML element.
"overflows()" on page 1626	Returns a boolean value indicating whether the current HTML element overflows its box boundaries.
"pageref()" on page 1580	Returns a **marker** that will be replaced with the element's page number after pagination. This only works for elements in the section that is currently being merged.
"parent()" on page 1580	Returns the parent of the HTML element.
"prepend()" on page 1581	Inserts content at the beginning of an HTML element or of each element in a set of HTML elements.
"prev()" on page 1584	Returns the previous sibling of the current HTML element.
"remove()" on page 1585	Removes an HTML element or a set of HTML elements from the document.
"removeAttr()" on page 1586	Removes the specified attribute from the current HTML element.
"removeClass()" on page 1587	Removes the specified class from the current HTML element. Has no effect if the class is not present.
"replaceWith()" on page 1587	Replaces the current HTML element (with a snippet, for example). Returns the current element.
"show()" on page 1588	Shows the HTML element in the template.
"tagName()" on page 1588	Returns the HTML tag name of the element, in uppercase. For an example see: "Creating a Table Of Contents" on page 1116.
"text()" on page 1589	Replaces the text content of the HTML element with the supplied value, or returns the text content of the element if no value is supplied.
"width()" on page 1590	Gets or sets the outer width of the HTML element, including padding and borders.

add()

The add() function adds elements to one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

add(content)

The add() function adds HTML content to one or more HTML elements and returns the union of this result or result set and other content.

content

A query result. This can be an HTML string or a result set.

Examples

This script adds one query result to another and sets the background color to yellow.

```
query("#test1").add(query("#test2")).css("background", "yellow");
```

Note: The way the functions add() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the results of the queries in a variable:

```
var myResult = query("#test1");
myResult.add(query("#test2");
myResult.css("background", "yellow");
```

The following script loads snippets in an iteration and adds their elements to an empty result set (using `query()`). Then it replaces a placeholder in the template with the new result.

```
var chapters = query();
for ( var i = 1; i <= 4; i++) {
chapters = chapters.add(loadhtml('snippets/Chapter' + i + '.html'));
}
results.replaceWith(chapters);
```

Selector	Matched element	Matched element after script execution
#chapters	{{chapters}}	Chapter 1 Lorem ipsum... Chapter 2 Lorem ipsum... Chapter 3 Lorem ipsum... Chapter 4 Lorem ipsum...

addClass()

Adds the specified class(es) to one ore more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

Adding a class has no effect if the class is already present.

addClass(classname)

Adds the specified class(es) to one or more HTML elements. This has no effect if the class is already present.

classname

String, space separated list of class names.

Examples

This script adds a class name to a paragraph.

```
results.addClass("foo");
```

Selector	Matched element	Matched element after script execution
p	Hello world	Hello world

The following script adds two class names to a paragraph.

```
results.addClass("foo bar");
```

Selector	Matched element	Matched element after script execution
p	Hello world	Hello world

after()

Inserts content after one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "before()" on page 1566.

after(content)

Inserts content after one or more HTML elements and creates a new result set.

content

String, HTML string or result set to insert after the matched elements. In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script looks up an element with the ID #salesrep and inserts a paragraph after it.

```
query("#salesrep").after("<p>Lorem ipsum</p>");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem ipsum

This script looks up an element with the ID #salesrep, sets its text color to red and inserts a paragraph after it.

```
query("#salesrep").after("<p>Lorem ipsum</p>").css("color","red");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem ipsum

Note that the way the functions after() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var salesrep = query("#salesrep");
salesrep.after("<p>Lorem ipsum</p>");
salesrep.css("color","red");
```

The following script inserts a paragraph after the elements in the results (the set of HTML elements that match the selector of the script).

```
results.after("<p>Lorem Ipsum</p>");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem ipsum

This script looks for the string "Lorem " in the results (the set of HTML elements that match the selector of the script) and inserts the string "ipsum" right after that text. The string is automatically enclosed in a span.

```
results.find("Lorem ").after("ipsum");
```

Matched element	Matched element after script execution
Lorem dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

This script looks up an element with the ID #salesrep and inserts a string after it. The string is automatically enclosed in a span.

```
query("#salesrep").after("Lorem Ipsum");
```

Matched element	Matched element after script execution
Peter Parker	Peter Parker Lorem Ipsum

append()

Insert content at the end of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "prepend()" on page 1581.

append(content)

Inserts content as the last element at the end of one or more HTML elements. `Append()` creates a new result set.

content

String, HTML string or result set to insert after the element(s). In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script appends a paragraph to the `results` (the set of HTML elements that match the selector of the script).

```
results.append("<p>Peter Parker</p>");
```

Selector	Matched element	Matched element after script execution
#box	 Personal information 	 Personal information Peter Parker

This script appends a string to the `results` (the HTML elements that match the selector of the script). The string is added to the end of the matched element(s) and wrapped in a Span element.

```
results.append("Peter Parker");
```

Selector	Matched element	Matched element after script execution
.name	 Personal information Name: 	 Personal information Name: Peter Parker

This script's selector is , so the script appends a paragraph to all Div elements in the template.

```
results.append("<p>Peter Parker</p>");
```

Selector	Matched element	Matched element after script execution
div	 Personal information Personal information 	 Personal information Peter Parker Personal information Peter Parker

The following script appends a snippet to a Div element with the ID `box`.

```
var a = loadhtml('snippets/snippet_name.html');
results.append(a);
```

Selector	Matched element	Matched element after script execution
#box	 Personal information 	 Personal information Peter Parker

This script looks for an element with the ID `box` and appends a paragraph to it.

```
query("#box").append("<p>Peter Parker</p>");
```

Matched element	Matched element after script execution
  Personal information 	 Personal information Peter Parker

This script looks for an element with the ID `box`, appends a paragraph to it and colors all text inside the box red.

```
query("#box").append("<p>Peter Parker</p>").css("color","red");
```

Matched element	Matched element after script execution
  Personal information 	 Personal information Peter Parker

Note: The way the functions append() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var box = query("#box");
box.append("<p>Peter Parker</p>");
box.css("color","red");
```

attr()

attr(attributeName) : String

Returns the value of the specified attribute of an HTML element which can be:

- The first element in a set of elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The first element in a set of elements returned by a query in the template (see "query ()" on page 1513).

attributeName

String; the name of the attribute.

Examples

This script - with the selector `img` - stores the source of the first image in a variable.

```
var src = results.attr("src");
```

The following script looks up an image with the ID #image1 and stores its background color in a variable.

```
var imgURL = query("#image1").attr("src");
```

attr(attributeName, value)

Sets the value of the specified attribute of one HTML element or of each element in a result set.

attributeName

String; the name of the attribute.

value

String; value for the attribute.

Examples

This script looks up an image in an element with the ID #calloutbox and sets its alternative text to a value from a data field.

```
var altText = record.fields.FavHobby;
query("#callout img").attr('alt', altText);
```

The following script sets the background color of a specific table cell in an email to red if the value of the field TOTAL has a negative value in the current record.

```
if(record.fields.TOTAL<0) {
    query("#total").attr("bgcolor","red");
}
```

before()

Inserts content before one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "after()" on page 1560.

before(content)

Before(content) inserts content before one or more HTML elements and creates a new result set.

content

String, HTML string or result set to insert after the elements. In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script looks for an element with the ID `salesrep`and inserts a paragraph before that element.

```
results.before("<p>Lorem Ipsum</p>");
```

Selector	Matched element	Matched element after script execution
#salesrep	Peter Parker	Lorem ipsum Peter Parker

This script does the same, but it uses the query() function to look up the element.

```
query("#salesrep").before("<p>Lorem ipsum</p>");
```

Matched element	Matched element after script execution
Peter Parker	Lorem ipsum Peter Parker

The following script looks for an element with the ID salesrep, inserts a paragraph before that element and colors that element red.

```
query("#salesrep").before("<p>Lorem ipsum</p>").css("color","red");
```

Matched element	Matched element after script execution
Peter Parker	Lorem ipsum Peter Parker

Note: the way the functions `before()` and `css()` are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var salesrep = query("#salesrep");
salesrep.before("<p>Lorem ipsum</p>");
salesrep.css("color","red");
```

The following script searches the results for the string "ipsum" and puts "Lorem " before it. "Lorem " is automatically wrapped in a Span element.

```
results.find("ipsum").before("Lorem ");
```

Matched element	Matched element after script execution
ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

The following script looks for an element with the ID salesrep and inserts the text "Lorem Ipsum" before that element. "Lorem Ipsum" is automatically wrapped in a Span element.

```
query("#salesrep").before("Lorem Ipsum");
```

Matched element	Matched element after script execution
ipsum dolor sit amet, consectetur adipiscing elit.	Lorem Ipsum Peter Parker

children()

Returns the immediate children (inner HTML) of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

Examples

This script retrieves the inner HTML of an element selected from a snippet.

```
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
results.append(snippet);
```

The following script retrieves the inner HTML of the elements and then performs a find/replace.

```
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
snippet.find('@firstname@').text('foobar');
results.append(snippet);
```

clone()

This function returns a copy of one HTML element or of a set of HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To duplicate an existing template element, clone it before calling `append()`; see "append()" on page 1562.

This function can also be used to output a variable number of sections in each document, based on data. See also: "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Examples

This script performs an iteration over the elements in the `results` (the elements that match the selector of the script).

```
var row = query("tbody tr", results).clone();
query("tbody", results).append(row);
```

The following script clones an existing table row to match the number of rows in a detail table. Then it iterates over the rows to populate the fields.

```
// Create the number of rows based on the records in the detail table
// We start at 1 so the boilerplate row is used too and there is no need to delete that row
for(var r = 1; r < record.tables.detail.length; r++) {
results.parent().append(results.clone());
}

// Iterate over the rows and populate them with the data from the accompanying data row
query("#table_2 > tbody > tr").each(function(i) {
this.find('@ItemNumber@').text( record.tables.detail[i].fields["ItemNumber"]);
this.find('@ItemOrdered@').text( record.tables.detail[i].fields["ItemOrdered"]);
this.find('@ItemTotal@').text( record.tables.detail[i].fields["ItemTotal"]);
this.find('@ItemDesc@').text( record.tables.detail[i].fields["ItemDesc"]);
this.find('@nr@').text(i);
});
```

The following script clones and populates a boilerplate row. Once completed you will need to hide the boilerplate row.

```
for(var i = 0; i < record.tables.detail.length; i++) {
```

```
var row = results.clone(); //Clone our boilerplate row

row.find('@ItemNumber@').text( record.tables.detail[i].fields["ItemNumber"]);
row.find('@ItemOrdered@').text( record.tables.detail[i].fields["ItemOrdered"]);
row.find('@ItemTotal@').text( record.tables.detail[i].fields["ItemTotal"]);
row.find('@ItemDesc@').text( record.tables.detail[i].fields["ItemDesc"]);
row.find('@nr@').text( i );

results.parent().append(row);
}

// Hide our boilerplate row (note that this doesn't really delete the row).
results.hide();
```

closest()

This function gets the first parent element that matches a selector, by testing the element itself and traversing up through its ancestors in the DOM tree. (In HTML, a parent is an element that contains another element.)

The function can be used for:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To get a child element or all child elements, use `children()` (see "children()" on page 1568).

The `closest()` command is based on the `closest()` command found in the jQuery library: https://api.jquery.com/closest/.

To get an element's sibling element, you can use "prev()" on page 1584 or "next()" on page 1577.

closest(selector)

For one HTML element or for each element in a set, this function gets the first element that matches the selector by testing the element itself and traversing up through its ancestors in the DOM tree.

selector

A String containing an HTML tag (without the angle brackets, <>).

Examples

The following script looks up all table rows in the template that contain an <input> element.

```
query("input").closest("tr");
```

This code gets the closest 'parent' row for each element that matches the selector of the script (collected in the `results` object):

```
results.closest("tr");
```

The rows could be colored red within the same statement:

```
results.closest("tr").css('background-color','red');
```

css()

Gets the value of a style property of one HTML element, or sets one or more CSS properties of one or more HTML elements.

css(styleName) : String

Returns the value of the specified CSS property of an HTML element, which can be:

- The first element in a set of elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The first element in a set of elements returned by a query in the template (see "query ()" on page 1513).

propertyName

String; the name of the CSS property.

Examples

This script stores the text color of the `results` (the HTML elements that match the selector of the script) in a variable.

```
var textcolor = results.css("color");
```

The following script looks up an element with the ID #calloutbox and stores its background color in a variable.

```
var backgroundcolor = query("#calloutbox").css("background-color");
```

css(styleName, value)

Function to set a CSS property of one HTML element or of each element in a result set.

propertyName

String; the name of the CSS property.

value

String; value for the CSS property or a map of property-value pairs to set.

Examples

This script looks up an element with the ID #calloutbox and sets its text color to red.

```
query("#callout p").css('color' , 'red');
```

The following script does the same, but it only sets the text color to red if in the current record the value of the field 'accounttype' is 'PRO'.

```
if(record.fields.accounttype == "PRO") {
query("#callout p").css("color","red");
}
```

This script sets the text color of the results to a hexadecimal color code.

```
results.css('color' , '#669900');
```

This script loads a snippet into a variable. Then it finds/replaces text in the snippet and applies a css property to the replacing text.

```
var mysnippet = loadhtml('snippets/snippet vars.html');
mysnippet.find('@var@').text('OL Connect').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

css(properties)

Function to set one or multiple CSS properties of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

properties

Array; map of property-value pairs to set.

Examples

This script colors the text of the `results` (the set of HTML elements that match the selector of the script) red and makes it bold.

```
results.css({'color' : 'red', 'font-weight' : 'bold'});
```

empty()

Removes the contents (child elements and inner HTML) from one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

Use "remove()" on page 1585 to remove the elements themselves.

Example

This script empties all Span elements found in the template.

```
results.empty();
```

Selector	Paragraph before script execution	Paragraph after script execution
span	Lorem ipsum -dolor sit amet, consectetuer adipiscing elit.	Lorem ipsum amet, consectetuer adipiscing elit.

find()

find(textToFind)

Performs a deep search for textToFind in the children of each element, and returns a new result set with elements that surround the occurrences.

textToFind

A String that contains the search text.

Example

The following piece of code loads a snippet, then looks for placeholders using find(), and replaces them with a text.

```
var mysnippet = loadhtml('snippets/snippet.html');
mysnippet.find('@var1@').text('OL Connect 1');
mysnippet.find('@var2@').html('<i>OL Connect 2</i>').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

hasClass()

hasClass(classname) : Boolean

Returns true if the HTML element or the first element in a result set has the specified class.

classname

String containing one class name.

Example

This script checks if the first of the `results` (the set of elements matching the selector of the script) has the class 'green'. If so, it colors the text of all the elements in the `results` green.

```
if (results.hasClass('green')) {
    results.css('color', 'green');
}
```

height()

Gets or sets the outer height, including padding and borders, of an HTML element, or of the first element in a result set (see "results" on page 1623), i.e. the set of HTML elements that match the selector of the script or of another query in the template (see "query()" on page 1513).

height(): Number

Returns the outer height of this element, including padding and borders, excluding margins. To include margins, call `height(true)`.

The returned value is the height in pixels, without measurement unit (e.g. 400).

height(value): void

Sets the outer height of this element, including padding and borders, excluding margins. To include margins, call `height(value, true)`.

value

Number. Give the height in pixels, but without the measurement unit (e.g. 400, not 400px).

Examples

This script adds 20 pixels to the width and height of an element.

```
var h = results.height();
var w = results.width();

h = h + 20;
w = w + 20;

results.height( h );
results.width( w );
```

The following script does the same, but in this case, margins are included.

```
var h = results.height(true);
var w = results.width(true);

h = h + 20;
w = w + 20;

results.height( h, true);
results.width( w, true);
```

hide()

Hides one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

This doesn't remove the elements; to make them visible again, use the function "show()" on page 1588.

These functions are used by the Conditional Script Wizard, as you can see when you open a Conditional Script and click the **Expand** button; see "Showing content conditionally" on page 958.

Example

This script hides or shows the elements matched by the selector of the script (which are stored in the `results` object), depending on the value of the data field `Country` in the current record.

```
if (record.fields["Country"] == "CANADA") {
results.show();
} else {
results.hide();
}
```

html()

html() : String

Returns the inner HTML of an HTML element, which can be:

- The first element in a set of elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The first element in a set of elements returned by a query in the template (see "query()" on page 1513).

html(value)

Replaces the inner HTML of an HTML element or of each element in a result set with the supplied value.

value

A String that may contain HTML tags.

Examples

The following script loads part of a snippet based on the value of a field, and then inserts the content into the document using `html()`.

```
var promoTxt = loadhtml('snippets/promo-en.html', '#' + record.fields['YOGA']);
results.html(promoTxt);
```

The following script loads a snippet. Then it looks for a placeholder (@var2@) in the text of that snippet and replaces every found placeholder by the text 'OL Connect 1'. It uses html() so the HTML formatting (and) will indeed be interpreted as HTML. Finally, it places the snippet in the template.

```
var mysnippet = loadhtml('snippets/snippet.html');
mysnippet.find('@var1@').html('<i>OL Connect 1</i>');
results.replaceWith(mysnippet);
```

info()

Returns pagination information for one HTML element or for the first element in a result set (see "results" on page 1623) of a query across all sections in a Print context (see "query (selector)" on page 1621).

The returned information is of the type `PaginationInfo` (see "PaginationInfo" on page 1622).

This function can only be used in a Post Pagination Script ; see "Post Pagination Script API" on page 1619.

For an example see: "Creating a Table Of Contents" on page 1116.

next()

`next()` returns the next sibling of an HTML element, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

In HTML, siblings are HTML elements that share the same parent, i.e. HTML elements in the same container element.

Note that a sibling can be a different type of element. For example, if a paragraph, an image and a table follow each other in the same Box, they are siblings.

> **Tip:** To get the **previous** sibling, use previous() (see "prev()" on page 1584).

What if there are no siblings?

If you call `prev()` or `next()` on a **collection** of elements (either "results" on page 1623 or the result set of "query()" on page 1513), the function will return a collection of siblings. If none of the elements has a previous or next sibling, the collection will be empty. Actions performed on an empty collection will simply have no effect.

If you call `prev()` or `next()` on a **single** element (this in an 'Each matched element' script), and the element has no previous or next sibling, the function returns `null`. Performing an operation on a `null` value will result in an error, which means that the rest of the script won't be executed. In this case, it is useful to check the return value, unless it's certain that it will never be `null`.

Example

Assume that there are three consecutive paragraphs in a Box and that the second of those paragraphs has an ID that matches the selector of this script. The paragraph is stored in the results object (see "results" on page 1623). The following script changes the font-weight of the next paragraph - the third paragraph in the Box - to bold.

```
results.next().css("font-weight", "bold");
```

If the elements in results don't have siblings, this line of code will have no effect. It also won't result in an error.

In an 'Each matched element' script you would first need to check the return value of this.next():

```
var nextSibling = this.next();
if nextSibling != null { nextSibling.css("font-weight", "bold"); }
```

overflows()

The `overflows()` method returns a boolean value indicating whether an HTML element overflows its box boundaries.

This function could for instance be used in a design where content needs to flow separately from the main text flow, or where a new, full-page, multi-column box should be inserted when the current one overflows.

Note that when called in a Standard script, this method runs *before* the pagination of Print output. After pagination - in the output, and in Preview mode - the element's overflow may have changed, especially when the element is combined with floating elements (i.e. elements that have the CSS style float) and located near a page-break.
To prevent this from happening, it may help to set the element's CSS style to `overflow:hidden`.
The `overflows()` method can also be used in a Post Pagination script.

Tip: You don't need a script to resize text in order to make it fit in a box. The Copy Fit feature automatically scales text to the available space (see "Copy Fit" on page 896).

Examples

```
var $box = query("#mybox");
while ( ! $box.overflows()) {
    //do something
}
```

A slightly shorter version:

```
while ( ! query("#mybox").overflows()) {
    //do something
}
```

The following script selects three boxes that all have the class box (selector: .box). It loops over them, inserting as many products from an array as possible into the box. In fact, it inserts one too many and removes the last one before moving on to the next box.

```
var products = ["Appetizer","Beans","Beef","Lettuce","Sprouts","Coconut","Juice","Soup","Coriander","Cheese","Pasta","Sugar","Vinegar","Bread"];
var i = 0;
results.each( function() {
    // Add elements to the box until it overflows.
    while ( ! this.overflows() && i < products.length ) {
        this.append("<p>" + products[i] + "</p>");
        i++;
    }
    // Go one step back unless we processed all items in our array.
    if( i < products.length ) {
        query("p:last-child", this).remove();
        i--;
    }
});
```

pageref()

Returns a **marker** that will be replaced with the element's page number after pagination. This only works for elements in the section that is currently being merged.

Example

Creating a table of contents

The following script creates a table of contents for all level 1 headings (<h1> elements) with the class `title` in one section.

```
var toc = '<ul ID="toc">';
query('h1.title').each(function()
{toc += '<li>' + this.text() + ' <span class="li_toc">' + this.pageref() + '</span></li>';
});
toc += '</ul>';
results.after(toc);
```

The first line creates a variable for the table of contents, which will be a list (a <ul> element with the ID `toc`). The start tag of the list is added to the variable.
The next line does a query for all level 1 headings (<h1> elements) with the class `title` in the current section. With `each()` the script loops through them. For each of the headings it adds a line item to the list, with the text (`this.text()`) and the page reference of the respective heading.
After the loop, the end tag of the list is added to the variable.
Finally, the script adds the variable - that now contains the table of contents - after the `results`. The `results` object contains the elements that match the selector of the script. So, if the script's selector selects the title of the table of contents, the table of contents will be added after that.

The following style rules, added to the style sheet, will align the chapter titles to the left and the page numbers to the right:

```
#toc li {
text-align:left;
}
#toc span {
float: right;
}
```

Note that these styles use the list's ID, that was defined in the first line of code. For information about style sheets, see "Styling templates with CSS files" on page 881.

parent()

Returns the parent(s) of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

In HTML, a parent is an element that contains another element.
To get an ancestor that matches a particular selector, use `closest()` (see "closest()" on page 1570).

To get an element's sibling element, you can use "prev()" on page 1584 or "next()" on page 1577.

Example

Assume that there are three paragraphs in a Box and that one of those paragraphs matches the selector of this script. The paragraph is stored in the `results` object (see "results" on page 1623). The following script retrieves the Box (which is the parent of the paragraph) using `results.parent()`, and then changes its background color to red.

```
results.parent().css('background-color' , 'red');
```

prepend()

Insert content at the beginning of one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

See also: "append()" on page 1562.

prepend(content)

Insert content as the first element to each element in the set of HTML elements that match the selector of the script or of another query in the template (see "query()" on page 1513). `Append` creates a new result set.

content

HTML string, string or HTML string to insert after the matched elements. In case a plain text string is provided, it is automatically wrapped in a element to avoid orphan text nodes to appear in the <body> element.

Examples

This script inserts a heading as the first element in an element that has the ID `#box`.

```
results.prepend("<h1>Personal information</h1>");
```

Selector	Matched element	Matched element after script execution
#box	 Peter Parker 	 Personal information Peter Parker

This script inserts a heading as the first element in an element that has the class `name`.

```
results.prepend("<b>Name: </b>");
```

Selector	Matched element	Matched element after script execution
.name	 Personal information Peter Parker 	 Personal information Name: Peter Parker

This script inserts content in multiple elements at the same time.

```
results.prepend("<h1>Personal information</h1>");
```

Selector	Matched element	Matched element after script execution
div	 Peter Parker Peter Parker 	 Personal information Peter Parker Personal information Peter Parker

This script prepends a snippet that contains the text "<h1>Personal information</h1>".

```
var a = loadhtml('snippets/snippet.html');
results.prepend(a);
```

Selector	Matched element	Matched element after script execution
div	 Peter Parker 	 Personal information Peter Parker

This script uses the function query() to find a box. Then it inserts a heading as the first element in that box.

```
query("#box").prepend("<h1>Personal information</h1>");
```

Matched element	Matched element after script execution
  Peter Parker 	 Personal information Peter Parker

This script uses the function query() to find a box, prepends a heading and sets the text color of the entire box to red.

```
query("#box").prepend("<h1>Personal information</h1>").css("color","red");
```

Matched element	Matched element after script execution
  Peter Parker 	 Personal information Peter Parker

Note: The way the functions prepend() and css() are used in this script is called 'chaining'. Chaining is optional; the same could be achieved by storing the result of the query in a variable:

```
var box = query("#box");
box.prepend("<p>Peter Parker</p>");
box.css("color","red");
```

prev()

`prev()` returns the previous sibling of an HTML element, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

In HTML, siblings are HTML elements that share the same parent, i.e. HTML elements in the same container element.
Note that a sibling can be a different type of element. For example, if a paragraph, an image and a table follow each other in the same Box, they are siblings.

Tip: To get the **next** sibling, use `next()` (see "next()" on page 1577).

What if there are no siblings?

If you call `prev()` or `next()` on a **collection** of elements (either "results" on page 1623 or the result set of "query()" on page 1513), the function will return a collection of siblings. If none of the elements has a previous or next sibling, the collection will be empty. Actions performed on an empty collection will simply have no effect.

If you call `prev()` or `next()` on a **single** element (this in an 'Each matched element' script), and the element has no previous or next sibling, the function returns `null`. Performing an operation on a `null` value will result in an error, which means that the rest of the script

won't be executed. In this case, it is useful to check the return value, unless it's certain that it will never be `null`.

Example

Assume that there are three consecutive paragraphs in a Box and that the second of those paragraphs has an ID that matches the selector of this script. The paragraph is stored in the results object (see "results" on page 1623). The following script changes the font-weight of the previous paragraph - the first paragraph in the Box - to bold.

```
results.prev().css("font-weight", "bold");
```

If the elements in results don't have siblings, this line of code will have no effect. It also won't result in an error.

In an 'Each matched element' script you would first need to check the return value of `this.prev()`:

```
var prevSibling = this.prev();
if prevSibling != null { prevSibling.css("font-weight", "bold"); }
```

remove()

Removes the current element or each element in a set from the DOM.

This function returns a new result set containing each removed element. These can be changed and inserted in the document. This could be beneficial in terms of performance, as manipulating elements inside the DOM is relatively time consuming.

Examples

This script removes all Span elements found in the template.

```
results.remove();
```

Selector	Paragraph before script execution	Paragraph after script execution
span	Lorem ipsum -dolor sit amet, con-sectetuer adipiscing elit.	Lorem ipsum amet, con-sectetuer adipiscing elit.

The selector of the following sample script is `tbody`. Before this script runs, the table body consists of a single placeholder row with three cells. After running the script, it contains thirty rows. To improve performance, most of the DOM manipulation takes place on detached elements.

```
// Detach the placeholder row from the DOM
var row = query("tr", results).remove();

// Modify the cells of this row
var cells = row.children();
cells[0].html("some text").css("background-color", "yellow");
cells[1].html("some text").css("font-weight", "bold");
cells[2].html("some text");

// Create a number of copies
var rows = row.clone();
for (var i = 0; i < 30; i++) {
    rows = rows.add(row.clone());
}

// Attach all copies to the DOM as children of tbody
results.append(rows);
```

removeAttr()

Removes the specified HTML attribute from an element or from each element in a set of elements. To add or change an attribute, use `attr()` (see "attr()" on page 1565).

removeAttr(attributeName)

attributeName

String; the name of the attribute.

Examples

This script looks up an email field in a form (which is an <input> with the ID #email1) and removes its `readonly` attribute.

```
query("#email1").removeAttr('readonly');
```

removeClass()

Removes the specified class from the current HTML element or from each element in a result set. This has no effect if the class is not present.

removeClass(classname)

classname

String, space separated list of class names.

Examples

This script removes the class name "foo" from all elements in the results that have this class.

```
results.removeClass("foo");
```

Selector	Matched element	Matched element after script execution
p	Hello world	Hello world

replaceWith()

Replaces one HTML element or each element in a set of HTML elements.

replaceWith(content)

Replaces one HTML element or each element in a set of HTML elements. Returns the result set.

content

A query result. This can be an HTML string or a result set.

Examples

Replace elements with a snippet

The following script loads a snippet and then replaces the elements matched by the script's selector with the snippet.

```
var snippet = loadhtml('snippets/mysnippet.html');
results.replaceWith(snippet);
```

Replace elements with a set of snippets

The following script loads snippets and adds their elements to a new, empty result set (using `query()`). Then it replaces a placeholder in the template with the set of snippets.

```
var chapters = query();
for ( var i = 1; i <= 4; i++) {
chapters = chapters.add(loadhtml('snippets/Chapter' + i + '.html'));
}
results.replaceWith(chapters);
```

show()

Shows one or more HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To hide elements (again), use the function "hide()" on page 1576.

These functions are used by the Conditional Script Wizard, as you can see when you open a Conditional Script and click the **Expand** button; see "Showing content conditionally" on page 958.

Example

This script hides or shows the elements matched by the selector of the script (which are stored in the `results` object), depending on the value of the data field `Country` in the current record.

```
if (record.fields["Country"] == "CANADA") {
results.show();
} else {
results.hide();
}
```

tagName()

Returns the HTML tag name of one HTML element in uppercase (e.g. H1, P, ARTICLE).

Example

This Post Pagination Script finds level 1 and 2 headings in all Print sections and stores their page number, text and HTML tag name.

```
merge.context.query("h1, h2").each(function() {
    var pageNo = this.info().pageNo;
    var text = this.text();
    var level = this.tagName();
});
```

This code is used in a script that inserts a table of contents. For the full script and explanation see "Creating a Table Of Contents" on page 1116.

text()

text() : String

Returns the text content of an HTML element or of the first element in a result set (see "results" on page 1623).

Example

This script loads a snippet into a variable and retrieves an element from the snippet using `query()` and `text()`.

```
var mysnippet = loadhtml('snippets/text-root-wrapped.html');
var subject = query("#subject", mysnippet).text();
results.append("<p style='font-weight: bold;'>" + subject + "</p>");
```

text(value)

Replaces the text content of one HTML element or of each element in a result set with the supplied value.

Example

This script loads a snippet, then looks for placeholders using find(), and replaces them using text(value).

```
var mysnippet = loadhtml('snippets/snippet.html');
mysnippet.find('@var1@').text('OL Connect 1');
mysnippet.find('@var2@').html('<i>OL Connect 2</i>').css('text-decoration','underline');
results.replaceWith(mysnippet);
```

width()

Gets or sets the outer width, including padding and borders, of an HTML element or of the first element in result set (see "results" on page 1623), i.e. the set of HTML elements that match the selector of the script or of another query in the template (see "query()" on page 1513).

width(): Number

Returns the outer width of this element, including padding and borders, excluding margins. To include margins, call `width(true)`.

The returned value is the width in pixels, without measurement unit (e.g. 400).

width(value): void

Sets the outer width of this element, including padding and borders, excluding margins. To include margins, call `width(value, true)`.

value

Number. Give the width in pixels, but without the measurement unit (e.g. 400, not 400px).

Examples

This script adds 20 pixels to the width and height of an element.

```
var h = results.height();
var w = results.width();

h = h + 20;
w = w + 20;

results.height( h );
results.width( w );
```

The following script does the same, but in this case, margins are included.

```
var h = results.height(true);
var w = results.width(true);

h = h + 20;
w = w + 20;

results.height( h, true);
results.width( w, true);
```

translate()

When a template has been prepared to use the translation feature (see "Translating templates" on page 1120), this feature is also accessible via a template script.

The `translate()` function gets the translation for a message, provided that a translation entry can be found for it (with the same context, if given).

If the text to be translated does not already exist in the template, you can create a translation entry for it by using the **Add Empty String** button on the "Translations pane" on page 1260.

Note that calling the `translate()` function may be unnecessary if a script adds text to an element that has the `data-translate` attribute; see "Tagging text that is inserted by a script" on page 1124. In that case, any existing translation will be applied automatically.

translate(message, context)

The `translate()` function gets the translation of a source text with an (optional) context. It returns the message as-is if no translation is found.

message

String containing the message (the source text).

context

The message's context (optional). A context may be specified when creating a translation entry. Specifying a context makes it possible to translate a certain text in different ways.

Example

The following line of code gets the translation of the string "Submit" that was registered with the context "button".

```
var str = translate( "Submit", "button" );
```

Control Script API

The table below lists the objects that are the most important in Control Scripts. Click through to the object to find a description and sample scripts.
See "Control Scripts" on page 1098 for information about this kind of scripts, how to insert them and what you can do with them.

Other objects that are available to Control Scripts

The list below isn't exhaustive: most of the objects listed in the Designer API (see "Standard Script API" on page 1481) are also available in Control Scripts. Not all of those objects can be used in Control Scripts, however. This is because Control Scripts differ from template scripts (see "What Control Scripts are" on page 1099). For example, the `query()` function

can't be used in a Control Script because it requires access to the DOM which a Control Script doesn't have.

Object	Usage
channel (see "Channel" on page 1616)	The channel for which output is generated. This is registered in the `merge` object: `merge.channel`. Note that the channel doesn't change when the output consists of different contexts. When generating email, for example, the channel is EMAIL, even when merging the Print context to attach it to the email.
"context" on page 1620	Object that contains one context and its sections. It is accessed through the `template` object: `merge.template.contexts`. To get access to one context, you have to specify the ContextType (see "ContextType" on page 1617), for example: `var printContext = merge.template.contexts.PRINT;`. Through the `merge` object you can find out which context is currently being merged: `merge.context`.
"masterpage" on page 1594	The `masterpage` object is used to set a Master Page's header and footer or to replace its entire HTML body.
"merge" on page 1635	The `merge` object gives access to the template with all of its contexts and sections.
"record" on page 1515	The current record in the main data set. To get the value of a field in the record, use `record.fields['fieldname']` or `record.fields.fieldname`.
"section" on page 1627	Much of the Control Script magic is performed by setting one of the fields of the `section` object. Via the `section` object you can omit, select and clone sections; add a background to a Print section; add a header to an email; etc. A section can be retrieved via the context that it belongs to, using `merge.template.contexts.ContextType.sections["section name"]`. For example: `merge.template.contexts.PRINT.sections["Section EN"]`.
"template" on page 1613	The `template` object contains all contexts and sections. It is accessed through the `merge` object: `merge.template`.

automation

This object (of the type: Automation) encapsulates the properties of the OL Connect Workflow process that triggered the current operation.

> **Note:** This object is only available in a **Web** context.
> The `automation` object available in DataMapper scripts is not of the same type. It has different properties.

Properties

The following table lists the properties of the **Automation** object.

Property	Description
jobInfos	Returns an object containing JobInfo 1 to 9 values from OL Connect Workflow (see Job Info variables in the Workflow Online Help).
Properties	Returns an object containing additional information (file name, process name and task ID) from OL Connect Workflow.

Accessing automation properties

To access Job Info 1 to 9 (defined in Workflow; see Job Info variables in the Workflow Online Help):

automation.jobInfos.JobInfo1;

To access ProcessName, OriginalFilename or TaskIndex (defined in Workflow):

automation.properties.OriginalFilename;

Example

Assume that a Workflow process can be triggered when an XML file appears in a certain folder. The XML file contains data that you want to show on a web page.

Add a Set Job Infos and Variables Task to the Workflow process (see Set Job Infos and Variables in the Workflow Online Help) . Define a Job Info variable (see Job Info variables), say, %9, and fill it with data from the XML, for example:

```
xmlget('/request[1]/values[1]/first[1]',Value,KeepCase,NoTrim)
```

In Connect Designer, in a Web section, you can use the `automation` object to retrieve the value in a script, like this:

```
var my_var = automation.jobInfos.JobInfo9;
```

context

In a Control Script, the `context` object represents one context in the template.

Which contexts are available in the template can be queried using `merge.template.contexts`.

The context being merged can be queried using `merge.context`.

Field	Type	Description
	Array	Array of sections (see "section" on page 1627) inside a particular context defined in the template. **Note:** When using `merge.context.sections` keep in mind that for example 'Section X' might only exist in your Print context, so using `merge.context.sections['Section X']` without enclosing it in the `if` statement `if (merge.context.type == ContextType.PRINT) {}` will yield an error when the script runs for other contexts. Alternatively, use the `template` object to access a specific context: `merge.template.contexts.PRINT.sections['Section X']`. Alternatively, use the `template` object to access the Print context: `merge.template.contexts.PRINT.sections['Section X']`.
	"ContextType" on page 1617	The context type: PRINT, EMAIL or WEB.

Function	Description
"query(selector)" on page 1621	Runs a query across all sections in the Print context. This function is only available in Post Pagination Scripts, which are only applied to the Print context. See "Post Pagination Scripts" on page 1114.

Example

This script checks if the output channel is EMAIL and if the context to be merged is the Print context (which happens if the Print context is attached to an email). If this is the case, it includes and excludes certain Print sections from the output.

```
if (channel == Channel.EMAIL) {
    if (merge.context.type == ContextType.PRINT) {
        merge.context.sections['Section 1'].enabled = false;
        merge.context.sections['Section 2'].enabled = false;
        merge.context.sections['Section 3'].enabled = true;
    }
}
```

masterpage

The `masterpage` object is used to set a Master Page's header and footer or to replace its entire HTML body. It can **only** be accessed in Control Scripts (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). Trying to access it in another type of script will result in an error.
Note that Master Pages are only used in a Print context (see "Master Pages" on page 648).

In script, Master Pages are retrieved via the masterpages Array which contains all Master Pages in the current template (see "template" on page 1613).
For example:

```
var myMasterpage = merge.template.masterpages["Master page 1"];.
```

Functions and fields

Field	Type	Description
"html()" on page 1612		Gets or sets the HTML of the body element.
"margins" on page 1633	Margins	Allows to set the header and footer of a Master Page, using a Measurement string, for example: "2in" (this sets a margin to two inches).

html()

In a Control Script, the `html()` function of a section or Master Page can be used to get the initial contents of its <body>, and modify them. This makes it possible, for example, to populate a section or Master Page with elements retrieved from a Content Management system, before Standard Scripts run.

html()

Gets the initial contents of the <body> of a section or Master Page.

html(value)

Replaces the contents of the <body> of a section or Master Page with the supplied value. This function is only available in Control Scripts. See also: "section" on page 1627 and "masterpage" on the previous page.

value

A String that may contain HTML tags.

Examples

The following script uses `html()` to retrieve the contents of a section and add a paragraph to it.

```
var foo = merge.context.sections["Section 1"].html();
foo += "<p>hello world</p>";
merge.context.sections["Section 1"].html( foo );
```

The following script loads a snippet based on the value of a field, and then replaces the content of a Print section with the snippet using `html()`.

```
var mySection = merge.context.sections["Section 1"];
var promoTxt = loadhtml('snippets/promo-' + record.fields['City'] + '.html');
mySection.html(promoTxt);
```

margins

The `margins` object is used to set a Print section or Master Page's margins in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).
Note that Master Pages are only used in a Print context (see "Master Pages" on page 648).

The `margins` object is retrieved via the `section` object (see "section" on page 1627) or via the `masterpages` array in a template (see "masterpage" on page 1594), respectively.

Fields

Field	Type	Description
	String	These fields allow to set the bottom, left, right and top of a Print section, using a Measurement string, for example: "2in" (this sets a margin to two inches).
	String	These fields allow to set the header and footer of a Master Page, using a Measurement string.

Examples

Setting the margins of a Print section

```
var sectionA = merge.template.contexts.PRINT.sections["Section A"];
sectionA.margins.top = "2in";
sectionA.margins.left = "2in";
sectionA.margins.right = "2in";
sectionA.margins.bottom = "2in";
```

Setting the header and footer of a Master Page

```
var masterA = merge.template.masterpages["Master page A"];
masterA.margins.header = "2in";
masterA.margins.footer = "2in";
```

media

The media object can be used to specify, enable and position a stationery's front and back in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).
Note that Media are only used in Print sections (see "Media" on page 652).

The available media are listed in, and retrieved via, the template object (see "template" on page 1613), for example:

```
var myMedia = merge.template.media.My_Media.
```

Fields

Field	Type	Description
	String	The height of the Media using a Measurement string. For example, "11in" is 11 inches. In a Control Script this property can be used to change the height of a Media. In other scripts it is only possible to get the height of the Media (read-only).
	Stationery	The Stationery's object's `front` and `back` fields are used to set the front and the back of a Media; see "front, back" below.
	String	The width of the Media using a Measurement string. For example, "8.5in" is 8.5 inches. In a Control Script this property can be used to change the width of a Media. In other scripts it is only possible to get the width of the Media (read-only).

front, back

The `front` and `back` fields of the Stationery object are used to set the front and the back of a Media (see "media" above).
Both `front` and `back` have the following fields.

Field	Type	Description
	boolean	When enabled, the stationery will be included in the output (Print sections only).
	String	Specifies the image to use as virtual stationery. For a file named My_Media.pdf, stored inside the template resources, the url would be `images/My_Media.pdf`. The complete syntax for an external file is: **file://<host>/<path>**. If the host is "localhost", it can be omitted, resulting in **file:///<path>**, for example: `file:///c:/users/Adminstrator/Pictures/My_Media.jpg`.
	Number	PDF and TIFF files may count more than one page. Specify the page number to use.

Field	Type	Description
	"MediaPosition" on page 1618	The `position` is used to place the stationery on the page (absolute, centered, fit to media).
	Measurement	The vertical offset from the top of the page, used to position the stationery (only when absolute positioning is selected). This value can be negative.
	Measurement	The horizontal offset from the left of the page, used to position the stationery (only when absolute positioning is selected). This value can be negative.

merge

In Control Scripts, the root level instance of the object `merge` is the entry point from where you can query and change the way contexts are merged. It gives access to the template with all its contexts and sections.
For more information about Control Scripts, see "Control Scripts" on page 1098 and "Control Script API" on page 1591.
Some of the objects are also useful in Post Pagination Scripts; see "Post Pagination Scripts" on page 1114 and "Post Pagination Script API" on page 1619.
For sample scripts, follow the links to the respective objects.

Field	Type	Description
"Channel" on page 1616	Channel	The final output channel: EMAIL, PRINT or WEB. The channel doesn't change when the output consists of different contexts. When generating an email, for example, the channel is EMAIL, even when merging the Print context to attach it to the email.
"context" on page 1620	Context	The context rendered by this merge run. If for one record, different contexts need to be output (for example, when the Print context is attached to an email) a record is merged multiple times: once per context. Per merge run, `merge.context` shows with which context the record is merged.
"Pagination" on page 1636	Pagination	Contains the total page count and sheet count of all sections in the Print context after pagination.
"section" on page 1627	Section	In Standard Scripts, this object defines the section that is being merged. **Note!** In Control Scripts, `merge.section` is **only** available when the output channel is **WEB**. To make sure that it is defined, use the following statement: `if (merge.channel == Channel.WEB && merge.context.type == ContextType.WEB) { ... }`. To retrieve any section in a Control Script, use: `merge.template.contexts.ContextType.Section['Section name'];` (for example: `merge.template.contexts.PRINT.sections["Section EN"]`). In Post Pagination Scripts, only Print sections are available.

Field	Type	Description
"template" on page 1613	Template	This object contains the template and all of its contexts. It can be used to find out which contexts are available in the template, using `merge.template.contexts` (see "context" on page 1620) and to manipulate the sections in those contexts (see "section" on page 1627).

resource()

The `resource()` function returns information about an image resource. This function is useful in a Control Script, for example to check the number of pages or the page height and width before setting it as a background (see "Control Script: Setting a Print section's background" on page 1106).

This function can also be used to check if a file exists. It returns `null` if the file does not exist. There is no need to explicitly check for `null`; a check to see if a resource exists can simply be written as:

```
var info = resource(path);
if (!info) {
// File does not exist
}
```

resource(location, pageNumber)

location

The location can be either:

- A path relative to the template root, like images/Image.png
- An absolute URL, like
  - http://image-server.com/Image.png
  - file:///C:/images/Image.png
  - file://C:/images/Image.png
  - file://192.168.1.123/images/Image.png
- An absolute local file system path, like C:\Images\image.png

pageNumber (optional)

The desired page. Counting starts at 1. If no page number is given, information about the first page will be retrieved.

The returned object is of the type ImageInfo. It has the following fields:

Field	Type	Description
	float	The height of the current page (in points).
	Number	Current page number (counting from 1) within the resource.
	Number	The total number of pages in the resource.
	"Permissions" below	PDF only. Allows to verify if a PDF is password-protected or has restrictions for printing.
	float	The width of the current page (in points)

Examples

This script retrieves the second page of a PDF that is present in the template's resources.

```
var pdf = resource("images/stamp.pdf", 2);
var height = pdf.height;
var width = pdf.width;
var numberOfPages = pdf.pages;
```

In the following script, the function is used to check if a file exists.

```
if(resource("C:/paw.pdf")){
    //exists
} else {
    //oops
}
```

Permissions

Allows to verify if a PDF is password-protected or has restrictions for printing.
This is part of the information about an image that is returned by the `resource()` function; see "resource()" on the previous page).

Note: If a PDF that does not allow content copying, Connect cannot handle the PDF at all and we cannot determine the permissions. In that case `hasPassword` will be `true` (even if the password is not set) and `printingAllowed`will be `unknown`.

Field	Type	Description
	Boolean	Will be `true` if the resource is a password protected PDF.
	String	Will be either "highres" (full permissions), "lowres", "none", or "unknown".

Example

The following script logs the permissions for a PDF.

```
var pdf = resource("images/myPDF.pdf");
var willPrint = pdf.permissions.printingAllowed;
var hasPW = pdf.permissions.hasPassword;
logger.info('Printing restrictions: ' + willPrint);
logger.info('Password-protected: ' + hasPW);
```

section

The section object can be used to query and modify how the section (and the related context) will be outputted. It is one of the most important objects in Control Scripts (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).

Retrieving a section

A section can be retrieved using `merge.template.contexts.ContextType.sections["section name"]`, for example: `merge.template.contexts.PRINT.sections["Section EN"]`.

A section can also be retrieved via `merge.context.sections['section name']`. Remember, however, that when several contexts need to be merged (for example, when the Print context is attached to an email), the script needs to check if the current context is of the type that contains the desired section (for example: `if (merge.context.type == ContextType.PRINT) {}`). When sections in different contexts have the same name, it is safer to use `merge.template.contexts.ContextType.sections["section name"]`.

Fields

Field	Type	Description
"background" on page 1607	Background	**Print sections only.** Used to set a PDF background on a Print section. See "Control Script: Setting a Print section's background" on page 1106 and "BackgroundResource" on page 1615.

Field	Type	Description
enabled	Boolean	Enables or disables this section for output (see "Examples" on page 1605). Note that even if a Print section is disabled, the `part` and `restartPageNumber` fields are still effective to define the parts division and page numbering over multiple sections when applicable. The default enabled state for sections (before any control script runs) is as follows: • For **Web** channel requests, the requested Web section is enabled by default. It is possible to redirect to another section by disabling the requested section and enabling another section. • For **Email** channel requests on the **Web** context, only the default section is enabled by default. It is possible to enable different or multiple sections, to control which sections will be attached to the email. • For **Email** channel requests on the **Print** context **all** Print sections are enabled by default. It is possible to enable different or multiple sections to control which sections will be attached to the email. • For **Email** channel requests, the requested section is enabled by default. It is possible to output another section by disabling the requested section and enabling another section. • For **Print** channel requests on the **Print** context **all** sections are enabled by default. It is possible to enable different or multiple sections to control which sections will be outputted.
headers	String	***DEPRECATED. Use the addHeader() function instead.*** **Email sections only**. Used to set custom email headers.
height	String	**Print sections only.** The height of the section using a Measurement string. For example, "11in" is 11 inches. In a Control Script this property can be used to set the height of the section. In other scripts it is only possible to get the height of the section (read-only).
"html()" on page 1612		Gets or sets the HTML of the body element.
margins	Margins	**Print sections only.** Used to set the print margins of a section. You can set the bottom, left, right and top margin using a Measurement string; for example, "2in" sets a margin to 2 inches.
meta	Object	The `meta` object can hold additional information about a section. This property is initially empty, and reset after every record. `Meta` entries can be added in a script (preferably a Control Script). The values stored in `section.meta` can have any type; they are not converted to strings. `Meta` entries can be read in a Standard Script and in Handlebars expressions. For examples, see meta and "Cloning sections" on page 965. **Note:** If a `meta` entry is added by a Standard Script, it cannot be accessed in a Handlebars expression, since Handlebars expressions are evaluated *before* Standard Scripts run.
minPages	Number	Minimum number of pages in the section. The default is 1.

Field	Type	Description
name	String	Used to get or set the name of the section. Note that section names must be unique and that sections cannot have an integer as its name. The name should always include alphanumeric characters. To rename email attachments, use the field `part`.
ownerPassword	String	**Print sections only.** Used to set the owner password for a PDF attachment.***** Setting only the owner password creates a secured PDF that can be freely viewed, but cannot be manipulated unless the owner password is provided. (Note that the recipient needs Adobe Acrobat to do this, because the Acrobat Reader does not allow users to enter the owner password.) See "Control Script: Securing PDF attachments" on page 1112.
pagination	"Pagination" on page 1635	**Post Pagination scripts only.** Contains the total page count, sheet count and start/end page numbers of a single section.
part	String	Name for the part. `part` is used to specify where a new part starts and the title for the part. This is used to split Email attachments. The Email output can, for example, attach 3 PDFs generated from the Print context. The part name will be used as the file name for the attachment. See "Parts: splitting and renaming email attachments" on page 1104.
password	String	**Print sections only.** Used to set the user password and owner password for a PDF attachment to the same value. See "Control Script: Securing PDF attachments" on page 1112.*****
restartPageNumber	Boolean	**Print sections only.** Enables or disables a restart of the page numbering. When generating Print output this can be used to let page numbering continue over multiple sections. The default value is `false`, meaning that each section will start with page 1 (to emulate behavior of previous versions).
"sheetConfig" on page 1609	SheetConfig	**Print sections only.** Overrides the Master Page, Media and Duplex printing options of first/middle/last/single or all sheets in a Print section.
width	String	**Print sections only.** The width of the section using a Measurement string. For example, "8.5in" is 8.5 inches. In a Control Script this property can be used to change the width of the section. In other scripts it is only possible to get the width of the section (read-only).

***** The password(s) should be set on the first Print section when producing a single attachment, or on the first section of each part when producing multiple attachments. Each of the parts (attachments) may have a different (or no) set of passwords.
Passwords set in the Control Script override the password set through the Email PDF password script (see "Email PDF password" on page 678). This allows you to change or remove the password from a specific part. Removal is done by setting the password field to null or "" (empty string).

Note: When a Control Script changes the size of a section, it should also change the size of the linked Media; this is not done automatically. While the output may still look good, a size mismatch can cause an issue if a script or another component assumes that the media size matches the section size.
In case of a size mismatch a preflight will show a warning (see "Doing a Preflight" on page 1075).

Batching settings

Batching improves performance and reduces memory use for templates that generate many pages per record. On the rare occasions where this may cause issues in the output, the following fields can be used to change the settings.

Field	Type	Description
batchedDOMPagesThreshold batchedPrintWindowThreshold	Integer	Cutoff point for batched pagination / batched printing. Default: 15. Batching can be disabled by setting the property to a high value (eg. 99999). Setting it to 0 causes an error.

Functions

Note: For cloned sections, only the `html()` function is available.

Function	Description
"clone()" on page 1611	Clone this section. See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.
	Add a cloned section after this section.
	Add a cloned section before this section. Note that with multiple clones, the next clone is always added **after** the previous clone. With `addBefore()`, the code `original.addBefore(clone1); original.addBefore(clone2);` will result in "clone1, clone2, original". With `addAfter()` the code `original.addAfter(clone1); original.addAfter(clone2);` results in "original, clone1, clone2".
	Email sections only. Used to set custom email headers. The function expects a key and a value (both of type string). For examples, see "Adding custom ESP handling instructions" on page 1679.
"html()" on page 1612	Sets the initial HTML of the body element.

Function	Description
"paginate()" on page 1634	**Post Pagination Scripts only**. Paginates the content of a Print section.

Examples

Conditionally skipping or printing Print sections

This script disables all Print sections and then re-enables one of them, depending on a value in the current record.

```
var printSections = merge.template.contexts.PRINT.sections;
printSections['Section EN'].enabled = false;
printSections['Section FR'].enabled = false;

if(record.fields.Language === 'FR'){
	printSections['Section FR'].enabled = true;
} else {
	printSections['Section EN'].enabled = true;
}
```

Positioning the background of a Print section

These scripts both set the background of a Print section to the same PDF, but they position it differently.

Using abolute positioning

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.position = MediaPosition.ABSOLUTE;
activeSection.background.left = "10mm";
activeSection.background.top = "10mm";
activeSection.background.url = "images/somepage.pdf";
```

Scaling to Media size

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.position = MediaPosition.FIT_TO_MEDIA;
activeSection.background.url = "images/somepage.pdf";
```

See also: "BackgroundResource" on page 1615, "MediaPosition" on page 1618 and "Control Script: Setting a Print section's background" on page 1106.

Cloning a Print section

If documents in the output need to have a variable number of sections, based on data, Print sections can be cloned (copied) on the fly. See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Selecting different sections for Print output and Email PDF attachment

This script selects a different Print section for output, depending on the output channel (Email or Print).

```
var printSections = merge.template.contexts.PRINT.sections;

if(merge.channel === Channel.EMAIL){
    printSections['Section 1'].enabled = false;
    printSections['Section 2'].enabled = true;
}

if(merge.channel === Channel.PRINT){
    printSections['Section 1'].enabled = true;
    printSections['Section 2'].enabled = false;
}
```

Setting the name of Email PDF attachments

This script renames the file name of an attachment by setting the part name of a section (see "Parts: splitting and renaming email attachments" on page 1104).

```
var section = merge.template.contexts.PRINT.sections['Section 1'];
section.part = 'Invoice ' + record.fields['InvoiceNo'];
```

Controlling multiple Email attachments

The following script attaches the following sections to an email:

- Print section 3 + 4 as attachment with continued page numbers
- Print section 6 as separate attachment (also see "Parts: splitting and renaming email attachments" on page 1104)
- Web sections A and B as separate attachment

```
if (channel == Channel.EMAIL) { // only when generating Email output
    if (merge.context.type == ContextType.PRINT) {
        merge.context.sections['Section 1'].enabled = false;
        merge.context.sections['Section 2'].enabled = false;
        merge.context.sections['Section 3'].enabled = true;
        merge.context.sections['Section 3'].part = "PDFAttach1";
        merge.context.sections['Section 4'].enabled = true;
        merge.context.sections['Section 4'].restartPageNumber = false;
        merge.context.sections['Section 5'].enabled = false;
        merge.context.sections['Section 6'].enabled = true;
```

```
        merge.context.sections['Section 6'].part = "PDFAttach2";
    }
    else if (merge.context.type == ContextType.WEB) {    // disable whatever is the default section
        merge.context.sections['default Section'].enabled = false;
        merge.context.sections['Section A'].enabled = true;
         merge.context.sections['Section A'].part = "WebPartA";
        merge.context.sections['Section B'].enabled = true;
         merge.context.sections['Section B'].part = "WebPartB";
    }
}
```

Note: For another example, see this how-to: Output sections conditionally.

Note: If the Email PDF Password Script Wizard defines a password, and a template has a Control Script that creates multiple PDF attachments, all the attachments are secured by the same password by default. Using a Control Script, you can set set different passwords for attachments; see "Control Script: Securing PDF attachments" on page 1112.

background

The `background` object holds the PDF background of a Print section (see "section" on page 1627 and "Control Script: Setting a Print section's background" on page 1106).

Note: Setting a page range using the `start` and `end` fields automatically sets `background.allPages` to `false`.
When you first define a page range and then set `background.allPages` to `true`, the page range will be disabled.

Fields

Field	Type	Description
	Boolean	Show all pages from the PDF.
	Number	The end page of the PDF to use as a background for the section.
	Measurement	The left offset of the PDF background (only when absolute positioning is selected).
	"MediaPosition" on page 1618	Set the position of the PDF background (Absolute, centered, fit to media).
	Number	Set the rotation of the PDF background to 0, 90, 180 or -90 degrees.

Field	Type	Description
	Number or Object	Set the size of the PDF background as a percentage of the original image. Accepted values are: • A numeric value. • A string containing a number and percentage character: "<number>%". • An object with two properties, "x" and "y", with numeric values: "{ x: 100, y: 100 }". If x and y are different the image will be stretched.
	"BackgroundResource" on page 1615	Set the source of the PDF background (NONE, Datamapper, PDF Resource). **Note:** Setting the background to NONE does not reset any other option, such as the position and scale of the background.
	Number	The start page of the PDF to use as a background for the section.
	Measurement	The top offset of the PDF background (only when absolute positioning is selected).
	String	The location of the PDF to use as a background for the section. Three different forms of URLs are supported: • A relative path to an image resource; this always starts with "images/". For a file named background.pdf, stored inside the template resources, the URL would be `images/background.pdf`. • A URL (either `file:`, `http:`, or `https:`). The complete syntax for an external file is: file://<host>/<path>. If the host is "localhost", it can be omitted, resulting in file:///<path>, for example: `file:///c:/resources/images/image.jpg`. • An absolute Windows-style path. A relative path and a URL are expected to be URL encoded. For example, a space needs to be encoded as `%20`. An absolute Windows-style path does not need to be URL encoded, but in JavaScript any backslashes in a String need to be double-escaped: "C:\\Images\\Image.pdf".

This script sets a background on a Print section using absolute positioning.

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.url = "images/somepage.pdf";
activeSection.background.position = MediaPosition.ABSOLUTE;
activeSection.background.left = "10mm";
activeSection.background.top = "10mm";
```

You could replace the last three lines of the previous script by the following line to scale the Print section background to Media size:

```
activeSection.background.position = MediaPosition.FIT_TO_MEDIA;
```

Tip: For more examples, see "Control Script: Setting a Print section's background" on page 1106.

sheetConfig

Sheet configuration settings are the Master Page, Media and Duplex printing options of first/middle/last/single or all sheets in a **Print** section. The `sheetConfig` object holds these options and can be used to set (or rather, override) them via a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591). This is especially useful when you need identical sections with different sheet configuration settings.

The `sheetConfig` object can be retrieved via the `section` object (see "section" on page 1627); see the example below.

Fields

The fields of the `sheetConfig` object correspond to settings in the Sheet Configuration dialog; see "Sheet Configuration dialog" on page 1208.

Field	Type	Description
	Boolean	Enables or disables duplex (two-sided) printing.
	Boolean	(Only with duplex printing.) When `facingPages` is set to `true`, the margins of the section switch alternately, so that pages are printed as if in a magazine or book.
	Number	Rotates the Media (to be more accurate: the Virtual Stationery images specified for this Media) by 0, 90, 180, or -90 degrees.
	Boolean	(Only with duplex printing.) Resets a page to Simplex if it has an empty back side. This may reduce volume printing costs.

	Positions (see "Position" on the facing page)	Used to modify the Master Page and Media and to allow content on sheets in different positions; for example:
		`var section1 = merge.template.contexts.PRINT.sections[0];`
		`section1.sheetConfig.positions.all.media = "Media 1";`
		For an overview of the available settings in each position see "Position" on the facing page.
		Can also be used to repeat the sheet configuration *within* a document.

	Boolean	(Only with duplex printing.) When `tumble` is set to `true`, pages are printed like in a calendar. (On Portrait output, this would be equivalent to short-edge duplex.)

Example

This script retrieves a section and changes its sheet configuration settings.

```
let section = merge.template.contexts.PRINT.sections["Section 1"];
section.sheetConfig.duplex = true;
section.sheetConfig.omitEmptyBackside = true;
section.sheetConfig.facingPages = true;
section.sheetConfig.mediaRotation = 0;
```

Position

The `positions` property of the `sheetConfig` object (see "sheetConfig" on the previous page) can be used to repeat the sheet configuration after a certain number of pages, or to retrieve and modify settings for sheets in different positions. Available positions are:

- **single**: The only sheet in a section.
- **first**: The first sheet in the section.
- **middle**: All after the first sheet and before the last sheet in a section.
- **last**: The last sheet in a section.
- **all**: All sheets in the section.

> **Note:** Settings for all sheets may conflict with settings for a particular position. In such cases, the setting that was specified last overrides the other.

Each of these positions has the following fields.

Field	Type	Description
	AllowContent	Used to specify on which sides of a sheet (in a certain position) content is allowed. Possible values: • AllowContent.ALL_SIDES • AllowContent.FRONT_ONLY (only with duplex printing) • AllowContent.BACK_ONLY (only with duplex printing) • AllowContent.NONE (not allowed on all, single and middle positions)
	String	Specifies the Master Page to use on the backside of sheets in this position.
	String	Specifies the Master Page to use on the front of sheets in this position.
	String	Specifies the Media to use with sheets in this position.
	Number	Repeat the sheet configuration every *n* number of pages. This is useful e.g. when documents can be so long that they cannot fit in one envelope.

Note: To remove a Media or Master Page via script, give it the value `undefined`, or `null`, or an empty string (""); see the examples below.

Examples

This script retrieves a Print section and modifies a number of settings for all of its sheets.

```
let section = merge.template.contexts.PRINT.sections["Section 1"];
section.sheetConfig.positions.all.allowContent = AllowContent.ALL_SIDES;
section.sheetConfig.positions.all.media = "MyMedia";
section.sheetConfig.positions.all.masterFront = "Master page 1";
section.sheetConfig.positions.all.masterBack = undefined; // or null, or an empty string
```

The following script ensures that empty backsides of single sheets are omitted.

```
let section = merge.template.contexts.PRINT.sections["Section 1"];
section.sheetConfig.positions.single.omitMasterOnEmptyBackside = true;
```

clone()

This function returns a copy of one HTML element or of a set of HTML elements, which can be:

- The elements that match the selector of a script (see "results" on page 1623).
- One element that matches the selector of a script that runs for "Each matched element" (see "this" on page 1554 and "Setting the scope of a script" on page 1071).
- The elements returned by a query in the template (see "query()" on page 1513).

To duplicate an existing template element, clone it before calling `append()`; see "append()" on page 1562.

This function can also be used to output a variable number of sections in each document, based on data. See also: "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Examples

This script performs an iteration over the elements in the `results` (the elements that match the selector of the script).

```
var row = query("tbody tr", results).clone();
query("tbody", results).append(row);
```

The following script clones an existing table row to match the number of rows in a detail table. Then it iterates over the rows to populate the fields.

```
// Create the number of rows based on the records in the detail table
// We start at 1 so the boilerplate row is used too and there is no need to delete that row
for(var r = 1; r < record.tables.detail.length; r++) {
results.parent().append(results.clone());
}

// Iterate over the rows and populate them with the data from the accompanying data row
query("#table_2 > tbody > tr").each(function(i) {
this.find('@ItemNumber@').text( record.tables.detail[i].fields["ItemNumber"]);
this.find('@ItemOrdered@').text( record.tables.detail[i].fields["ItemOrdered"]);
this.find('@ItemTotal@').text( record.tables.detail[i].fields["ItemTotal"]);
this.find('@ItemDesc@').text( record.tables.detail[i].fields["ItemDesc"]);
this.find('@nr@').text(i);
});
```

The following script clones and populates a boilerplate row. Once completed you will need to hide the boilerplate row.

```
for(var i = 0; i < record.tables.detail.length; i++) {

var row = results.clone(); //Clone our boilerplate row

row.find('@ItemNumber@').text( record.tables.detail[i].fields["ItemNumber"]);
row.find('@ItemOrdered@').text( record.tables.detail[i].fields["ItemOrdered"]);
row.find('@ItemTotal@').text( record.tables.detail[i].fields["ItemTotal"]);
row.find('@ItemDesc@').text( record.tables.detail[i].fields["ItemDesc"]);
row.find('@nr@').text( i );

results.parent().append(row);
}

// Hide our boilerplate row (note that this doesn't really delete the row).
results.hide();
```

html()

In a Control Script, the `html()` function of a section or Master Page can be used to get the initial contents of its <body>, and modify them. This makes it possible, for example, to populate a section or Master Page with elements retrieved from a Content Management system, before Standard Scripts run.

html()

Gets the initial contents of the <body> of a section or Master Page.

html(value)

Replaces the contents of the <body> of a section or Master Page with the supplied value. This function is only available in Control Scripts. See also: "section" on page 1627 and "masterpage" on page 1594.

value

A String that may contain HTML tags.

Examples

The following script uses `html()` to retrieve the contents of a section and add a paragraph to it.

```
var foo = merge.context.sections["Section 1"].html();
foo += "<p>hello world</p>";
merge.context.sections["Section 1"].html( foo );
```

The following script loads a snippet based on the value of a field, and then replaces the content of a Print section with the snippet using `html()`.

```
var mySection = merge.context.sections["Section 1"];
var promoTxt = loadhtml('snippets/promo-' + record.fields['City'] + '.html');
mySection.html(promoTxt);
```

paginate()

This method of the `section` object (see "section" on page 1627) triggers pagination of the current section. The pagination process re-establishes page boundaries, updates page numbers and page counts, and reapplies Master Pages.
Depending on whether page numbering restarts in each section this may affect the page numbers in other sections as well (see "Configuring page numbers" on page 645).

When the pagination process has ended, the script resumes.

The 'current section' is always a Print section, since this method can only be used in Post Pagination Scripts, and Post Pagination Scripts only run on the Print context (see "Post Pagination Scripts" on page 1114).
You only need to call `merge.section.paginate()` in a Post Pagination Script if the script has added or removed content to such an extent that the page boundaries need to be renewed.

For an example see: "Creating a Table Of Contents" on page 1116.

template

The `template` object represents the template with all its contexts and sections. It is used frequently in Control Scripts (see "Control Scripts" on page 1098 and "Control Script API" on page 1591) but it can also be used in Standard Scripts.

It is retrieved via the merge object: `merge.template` (see "merge" on page 1635).

Which contexts are available in the template can be queried using `merge.template.contexts`. To get access to a specific context, you have to specify the ContextType (see "ContextType" on page 1617).

Field	Type	Description
contexts	Array	Array of contexts (see "context" on page 1620) available in the template. The contexts contain the sections (see "section" on page 1627).
images	Array	The list of image resources (see "ImageResource" on page 1617) included in the template.
masterpages	Array	Array of Master Pages (see "masterpage" on page 1594) available in the template.
"media" on page 1597	Array	Media available to this template (see "Media" on page 652). For each of them you can specify, enable and position the stationery's front and back.
parameters	Parameters	Runtime parameters for content creation. See "Runtime parameters" on page 608.
"properties" below	Properties	This object contains all default properties of the template as well as any custom properties. (On the menu, select **File > Properties** to view and complement the file properties. See "File Properties dialog" on page 1150).

Example

The following Control Script retrieves two Print sections. Then, depending on a value in the current record, it enables one section or the other, so that only one of the two sections appears in the output.

```
var printSections = merge.template.contexts.PRINT.sections;
printSections['Section EN'].enabled = false;
printSections['Section FR'].enabled = false;

if(record.fields.Language === 'FR'){
    printSections['Section FR'].enabled = true;
} else {
    printSections['Section EN'].enabled = true;
}
```

properties

The `properties` object inside the `template` object (see "template" on the previous page) contains all default properties of the template file as well as any custom properties.

To view and complement the file properties, select **File > Properties** on the menu. See "File Properties dialog" on page 1150.

Following are the default properties.

Field	Type	Description
application	String	Application version when the document was last saved
author	String	Original author of the document
company	String	company name
created	Date	Date and time on which the document was created
customProperties	List	A list of defined custom properties
description	String	Description of the document
file	String	File name of the document
keywords	String	Semicolon-separated list of keywords
modified	Date	Date and time on which the document was last saved

Example

This script stores a default property (author) and a custom property in variables.

```
var author = merge.template.properties.author;
var myProperty = merge.template.properties.customProperties.myPropertyName;
```

BackgroundResource

BackgroundResource is an enumeration for the types of background resources for a Print section (see "background" on page 1607 and "section" on page 1627).

A Print section can be retrieved in script using `merge.template.contexts.ContextType.sections["section name"]`, for example `merge.template.contexts.PRINT.sections["Section EN"]`.

Field	Description
	A PDF file retrieved via the active data mapping configuration. This can be the PDF file that was used as input file, or another type of input file, converted to PDF.
	No PDF background.
	A PDF or other image file which is stored in the template or on the network. Note that it isn't possible to use a remotely stored image file as a section's background.

Example

The following script sets the background for a section called 'Policy' to `RESOURCE_PDF` and specifies a path for it, using a data value:

```
// Enable the section background and specify that the PDF should be read
// from a resource file rather than using a PDF DataMapper background
merge.template.contexts.PRINT.sections['Policy'].background.source = BackgroundResource.RESOURCE_PDF;

// Specify the path
var resourceUrl = 'images/policy-' + record.fields.policy + '.pdf';
merge.template.contexts.PRINT.sections['Policy'].background.url = resourceUrl;
```

Note: To learn how to set a PDF file as a background image on a Print section without a Control Script, see "Using a PDF file or other image as background" on page 634.

Channel

`Channel` is an enumeration for the output channels. The active output channel is registered in `merge.channel`.

The channel doesn't change when the output consists of different contexts. When generating email, for example, the channel is EMAIL, even when merging the Print context to attach it to the email.

Value	Description
	The merge request is for output to Email.
	The merge request is for output to Print.
	The merge request is for output to Web.
	The merge request is for generating a template preview.

Example

The following Control Script selects different sections for Print output and for Email with the Print context attached to it.

```
var printSections = merge.template.contexts.PRINT.sections;

if(merge.channel === Channel.EMAIL){
```

```
    printSections['Section 1'].enabled = false;
    printSections['Section 2'].enabled = true;
}

if(merge.channel === Channel.PRINT){
    printSections['Section 1'].enabled = true;
    printSections['Section 2'].enabled = false;
}
```

ContextType

ContextType is an enumeration for the context types.

The type of the context that is going to be merged next can be retrieved via `merge.context.type`.

The context type needs to be specified when retrieving a section with `merge.template.contexts.ContextType.sections["section name"]`, for example `merge.template.contexts.PRINT.sections["Section EN"]`.

Value	Description
	The context is the Email context.
	The context is the Print context.
	The context is the Web context.

Example

This script retrieves two Print sections. Then, depending on a value in the current record, it enables one section or the other, so that only one of the two sections appears in the output.

```
var printSections = merge.template.contexts.PRINT.sections;
printSections['Section EN'].enabled = false;
printSections['Section FR'].enabled = false;

if(record.fields.Language === 'FR'){
    printSections['Section FR'].enabled = true;
} else {
    printSections['Section EN'].enabled = true;
}
```

ImageResource

The `ImageResource` object represents an image resource in a template.

In script, an `ImageResource` is retrieved via the `images` Array which contains a list of all image resources that are included in the current template (see "template" on page 1613). For example: `var myImage = merge.template.images[0];`.

Functions and fields

Field	Type	Description
modified	Number	Timestamp of the last modification of the image resource, for example: 1566660504000.
name	String	The base name of the image resource, for example: "cat.jpg".
path	String	The path of the image resource, relative to the template, for example: "images/cat.jpg".

Example

The following script adds an image called 'dog.jpg' to the content of the template if the image can be found in the template's resources:

```
const image = merge.template.images.find(image => image.name == 'dog.jpg');
if (image) {
    results.after(`<img src=${image.path}>`);
}
```

MediaPosition

In a Control Script, the `position` is an enumeration for the position of background resources for a Print section. It is retrieved and set via `background.position`.

Field	Description
	Places the PDF at a specific location on the page. Set the background's top (`background.top`) and left (`background.left`) measured from the top and left side of the section.
	Centers the PDF on the page, vertically and horizontally.
	Stretches the PDF to fit the page size.

Examples

This script applies **absolute positioning** to the background of a Print section.

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.position = MediaPosition.ABSOLUTE;
activeSection.background.left = "10mm";
activeSection.background.top = "10mm";
activeSection.background.url = "images/somepage.pdf";
```

The next script scales the background of a Print section to the size of the **Media**.

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.position = MediaPosition.FIT_TO_MEDIA;
activeSection.background.url = "images/somepage.pdf";
```

Post Pagination Script API

The table below lists the functions and objects that are **only** available in Post Pagination Scripts. Click through to the object to find a description and sample scripts.
See "Post Pagination Scripts" on page 1114 for information about this kind of scripts, how to insert them and what you can do with them.

Note: Post Pagination Scripts only apply to the Print context. These functions and objects cannot be used with other contexts.

Tip: It is possible to *define* a function in a Control Script which calls these functions, as long as that function is only invoked from a Post Pagination Script.

Object or function	Usage
merge.context."query(selector)" on page 1621	Call `merge.context.query(selector)` to run a query across all sections in the Print context.
merge.section."paginate()" on page 1634	A call to `merge.section.paginate()` triggers pagination of the current section. Note: `section` is assumed to refer to a variable that holds a Section object (see "Retrieving a section" on page 1627).
merge.section.pagination	Used to get the total page count, sheet count and start/end page numbers for a single section. See "Pagination" on page 1635. Note: `section` is assumed to refer to a variable that holds a Section object (see "Retrieving a section" on page 1627).
merge.pagination	Used to get the total page count, sheet count and start/end page numbers for all sections in the Print context after pagination. See "Pagination" on page 1636.
"PaginationInfo" on page 1622	This object contains essential information about one element in order to create a table of contents (see "Creating a Table Of Contents" on page 1116).
results."info()" on page 1626	A call to results.info() returns pagination information (see "PaginationInfo" on page 1622) for the first element in the result set of a query (see "query(selector)" on page 1621).

Other objects that are available to Post Pagination Scripts

The list above isn't exhaustive: the objects listed in the Designer API (see "Standard Script API" on page 1481) and in the Control Script API (see "Control Script API" on page 1591) are also available in Post Pagination Scripts.

contentitem

The `contentitem` object holds the Print Content Item that will be written to the Connect database when generating Print output.
Its `properties` field allows to add custom properties to the Print Content Item, in the form of key-value pairs (a JSON string).

Custom properties can be utilized for further processing in a Workflow configuration with the Retrieve Items task. The Retrieve Items task retrieves custom properties along with the base record information (see Retrieve Items in Workflow's Online Help).

The `contentitem` object can be used in any type of script. However, if you want to add production information - such as the page, size, and position of elements after pagination - to the `properties`, you have to write a Post Pagination Script (see "Post Pagination Scripts" on page 1114).

Field	Type	Description
	Properties	Key-value pairs containing custom properties. Note that a property value is always stored as a string in the database.

Example

This following code would add a property called 'myProperty' with the value 'myvalue':

```
contentitem.properties.myProperty = 'myvalue';
```

You can replace 'myProperty' and 'myvalue' with whatever name and value best suits the use case.

If the name of the property contains a space you'll need to put it between brackets and quotation marks:

```
contentitem.properties['name with spaces'] = 'value';
```

If the value is a number you don't need to put it between quotation marks:

```
contentitem.properties.custom_property = 123;
```

context

In a Control Script, the `context` object represents one context in the template.

Which contexts are available in the template can be queried using `merge.template.contexts`.

The context being merged can be queried using `merge.context`.

Field	Type	Description
	Array	Array of sections (see "section" on page 1627) inside a particular context defined in the template. **Note:** When using `merge.context.sections` keep in mind that for example 'Section X' might only exist in your Print context, so using `merge.context.sections['Section X']` without enclosing it in the `if` statement `if (merge.context.type == ContextType.PRINT) {}` will yield an error when the script runs for other contexts. Alternatively, use the `template` object to access a specific context: `merge.template.contexts.PRINT.sections['Section X']`. Alternatively, use the `template` object to access the Print context: `merge.template.contexts.PRINT.sections['Section X']`.
	"ContextType" on page 1617	The context type: PRINT, EMAIL or WEB.

Function	Description
"query(selector)" below	Runs a query across all sections in the Print context. This function is only available in Post Pagination Scripts, which are only applied to the Print context. See "Post Pagination Scripts" on page 1114.

Example

This script checks if the output channel is EMAIL and if the context to be merged is the Print context (which happens if the Print context is attached to an email). If this is the case, it includes and excludes certain Print sections from the output.

```
if (channel == Channel.EMAIL) {
    if (merge.context.type == ContextType.PRINT) {
        merge.context.sections['Section 1'].enabled = false;
        merge.context.sections['Section 2'].enabled = false;
        merge.context.sections['Section 3'].enabled = true;
    }
}
```

query(selector)

This function of the `context` object (see "context" on the previous page) returns a result set, containing the HTML elements in **all** sections of the Print context that match the supplied CSS selector.

The new result set is of the type `QueryResults`, just like the `results` object which is also the result of a (hidden) query (see "results" on the next page), but it is **read-only**.

This function can only be called in a Post Pagination Script (see "Post Pagination Scripts" on page 1114). It is indispensable in a script that creates a table of contents, as described in the following topic: "Creating a Table Of Contents" on page 1116.

selector

A String containing a CSS selector. See MDN Web Docs - CSS Selectors and combinators for CSS selectors and combinations of CSS selectors.

Example

This script returns all level 1 headings in all sections of the Print context.

```
var headings = merge.context.query("h1");
```

PaginationInfo

The PaginationInfo object is returned by the `info()` function of the `results` object (see "info()" on page 1626 and "results" on the next page). It contains information about where an element is located: on which page and which sheet, and the total number of pages and sheets in the page bundle in which the element is located.
A **page bundle** is a group of sections for which the Restart page numbering option is turned off. In other words, page numbering continues from the start of the first section to the end of the last section in the group. (See "Configuring page numbers" on page 645 and "Control Script: Page numbering" on page 1102.)

The PaginationInfo object (as well as the `info()` function) is only available in Post Pagination Scripts (see "Post Pagination Scripts" on page 1114). It is essential in a Post Pagination Script that creates a table of contents, as described in the following topic: "Creating a Table Of Contents" on page 1116.

For the total page count, sheet count and start/end page numbers of a **single section**, use the `section` object's "Pagination" on page 1635 (see also: "section" on page 1627).

For the total page and sheet count of **all Print sections** together use the `merge` object's "Pagination" on page 1636 (see also: "merge" on page 1635).

Field	Type	Description
pageCount	Number	The number of pages in the page bundle associated with the section in which the first element of the results is located.

Field	Type	Description
pageNo	Number	The page number on which the first element of the `results` is located.
"section" on page 1627	Section	The section in which the element is located. This is always a Print section, since Post Pagination Scripts only operate on the Print context.
sheetCount	Number	The number of sheets in the page bundle associated with the section in which the first element of the `results` is located.
sheetNo	Number	The sheet number on which the first element of the `results` is located.

results

The `results` object (type: `QueryResults`) is the result of the query for HTML elements that match the selector of the script. The selector of a script can be specified in the Script Editor and is visible in the second column of the Scripts pane, next to the name of the script.

If, for example, a script would have the selector `p.onlyCanada`, the script would apply to all paragraphs that have the class `onlyCanada`. (Classes can be defined in the **Attributes** pane at the right: select the element in the content and type the class(es) in the **Class** field.)
The script could then use the `results` object to hide or show those paragraphs, depending on the value of the data field `Country` in the current record:

```
if (record.fields["Country"] == "CANADA") {
results.show();
} else {
results.hide();
}
```

Note: This object can't be used in Control Scripts, because they don't have a selector.

Tip: The easiest way to access the elements in a result set one by one, is by setting the scope of the script to "Each matched element" and using the this object (see "this" on page 1554).

Property

Field	Type	Description
	Number	**Number of elements in this result set. Equivalent to calling size().**

Functions

The functions below can be called by the `results` object and by any other result set that is returned by a query, see "query()" on page 1513.

Function	Description
"add()" on page 1557	Adds elements to a set of HTML elements.
"addClass()" on page 1559	Adds the specified class to each element in a set of HTML elements. Has no effect if the class is already present.
"after()" on page 1560	Inserts content after each element in a set of HTML elements..
"append()" on page 1562	Inserts content at the end of each element in a set of HTML elements.
"attr()" on page 1565	Change the given attribute of the element or set of HTML elements with the given value.
"before()" on page 1566	Inserts content before an element or before each element in a set of HTML elements.
"children()" on page 1568	Returns the immediate children of an HTML element.
"clone()" on page 1611	Returns a new result set containing a copy of each element in a set of HTML elements.
"closest()" on page 1570	For each element in a set, this function gets the first parent element that matches a selector, by testing the element itself and traversing up through its ancestors in the DOM tree.
"css()" on page 1571	Gets the value of a style property for the first element in set of HTML elements or sets one or more CSS properties for every element in a set of HTML elements.
"empty()" on page 1573	Removes the contents (child elements and inner HTML) from one element or a set of elements in the template.
"filter()" on page 1536	Returns a subset of the current result set.
"find()" on page 1574	Performs a search for a text in the children of each element in a set of HTML elements, and returns a new result set with elements that surround the occurrences.
"get(index)" on page 1538	Returns the element (type: QueryResult) found at the supplied index in a set of HTML elements.

Function	Description
"hasClass()" on page 1574	Returns `true` if the first element in this result set has the specified class.
"height()" on page 1575	Gets or sets the outer height of an element, including padding and borders.
"hide()" on page 1576	Hides the HTML element or set of HTML elements.
"html()" on page 1576	Replaces the inner HTML of the element or of each element in a set of HTML elements with the supplied value, or returns the HTML of the first element if no value is supplied.
"info()" on the facing page	**Post Pagination Scripts only**. Returns pagination information for the first element in this result set.
is(selector)	Returns true if at least one of the elements in a set of HTML elements matches the supplied CSS selector.
"next()" on page 1577	Returns the next sibling of each HTML element in the result set.
"overflows()" on the facing page	Returns a boolean value indicating whether an HTML element overflows its box boundaries.
"pageref()" on page 1580	Returns a **marker** that will be replaced with the element's page number after pagination. This only works for elements in the section that is currently being merged.
"parent()" on page 1580	Returns the parents of the elements in a set of HTML elements.
position()	Returns information about the position (see NodePosition) of the first element in this result set, excluding margins. To include margins, call `position(true)`.
"prepend()" on page 1581	Inserts content at the beginning of an HTML element or of each element in a set of HTML elements.
"prev()" on page 1584	Returns the previous sibling of each HTML element in the result set.
"remove()" on page 1585	Removes an HTML element or a set of HTML elements from the document.
"removeAttr()" on page 1586	Removes the specified attribute from each element in this result set.
"removeClass()" on page 1587	Removes the specified class from an element or from each element in a set of HTML elements. Has no effect if the class is not present.

Function	Description
"replaceWith()" on page 1587	Replaces an HTML element or a set of HTML elements (with a snippet, for example). Returns the result set.
"show()" on page 1588	Shows the HTML element or a set of HTML elements.
size()	Gets the number of elements in this result set. Equivalent to the `length` property.
"tagName()" on page 1588	Returns the HTML tag name of the first element in this result set, in uppercase. For an example see: "Creating a Table Of Contents" on page 1116.
"text()" on page 1589	Replaces the text content of an HTML element or of each element in a set of HTML elements with the supplied value, or returns the text content of the first element if no value is supplied.
"width()" on page 1590	Gets or sets the outer width of an element, including padding and borders.

info()

Returns pagination information for one HTML element or for the first element in a result set (see "results" on page 1623) of a query across all sections in a Print context (see "query (selector)" on page 1621).

The returned information is of the type `PaginationInfo` (see "PaginationInfo" on page 1622).

This function can only be used in a Post Pagination Script ; see "Post Pagination Script API" on page 1619.

For an example see: "Creating a Table Of Contents" on page 1116.

overflows()

The `overflows()` method returns a boolean value indicating whether an HTML element overflows its box boundaries.

This function could for instance be used in a design where content needs to flow separately from the main text flow, or where a new, full-page, multi-column box should be inserted when the current one overflows.

Note that when called in a Standard script, this method runs *before* the pagination of Print output. After pagination - in the output, and in Preview mode - the element's overflow may have changed, especially when the element is combined with floating elements (i.e. elements that have the CSS style float) and located near a page-break.
To prevent this from happening, it may help to set the element's CSS style to

`overflow:hidden`.
The `overflows()` method can also be used in a Post Pagination script.

> **Tip:** You don't need a script to resize text in order to make it fit in a box. The Copy Fit feature automatically scales text to the available space (see "Copy Fit" on page 896).

Examples

```
var $box = query("#mybox");
while ( ! $box.overflows()) {
    //do something
}
```

A slightly shorter version:

```
while ( ! query("#mybox").overflows()) {
    //do something
}
```

The following script selects three boxes that all have the class box (selector: .box). It loops over them, inserting as many products from an array as possible into the box. In fact, it inserts one too many and removes the last one before moving on to the next box.

```
var products = ["Appetizer","Beans","Beef","Lettuce","Sprouts","Coconut","Juice","Soup","Coriander","Cheese","Pasta","Sugar","Vinegar","Bread"];
var i = 0;
results.each( function() {
    // Add elements to the box until it overflows.
    while ( ! this.overflows() && i < products.length ) {
        this.append("<p>" + products[i] + "</p>");
        i++;
    }
    // Go one step back unless we processed all items in our array.
    if( i < products.length ) {
        query("p:last-child", this).remove();
        i--;
    }
});
```

section

The section object can be used to query and modify how the section (and the related context) will be outputted. It is one of the most important objects in Control Scripts (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).

Retrieving a section

A section can be retrieved using `merge.template.contexts.ContextType.sections["section name"]`, for example: `merge.template.contexts.PRINT.sections["Section EN"]`.

A section can also be retrieved via `merge.context.sections['section name']`. Remember, however, that when several contexts need to be merged (for example, when the Print context is attached to an email), the script needs to check if the current context is of the type that contains the desired section (for example: `if (merge.context.type == ContextType.PRINT) {}`). When sections in different contexts have the same name, it is safer to use `merge.template.contexts.ContextType.sections["section name"]`.

Fields

Field	Type	Description
"background" on page 1607	Background	**Print sections only.** Used to set a PDF background on a Print section. See "Control Script: Setting a Print section's background" on page 1106 and "BackgroundResource" on page 1615.
enabled	Boolean	Enables or disables this section for output (see "Examples" on page 1631). Note that even if a Print section is disabled, the `part` and `restartPageNumber` fields are still effective to define the parts division and page numbering over multiple sections when applicable. The default enabled state for sections (before any control script runs) is as follows: • For **Web** channel requests, the requested Web section is enabled by default. It is possible to redirect to another section by disabling the requested section and enabling another section. • For **Email** channel requests on the **Web** context, only the default section is enabled by default. It is possible to enable different or multiple sections, to control which sections will be attached to the email. • For **Email** channel requests on the **Print** context **all** Print sections are enabled by default. It is possible to enable different or multiple sections to control which sections will be attached to the email. • For **Email** channel requests, the requested section is enabled by default. It is possible to output another section by disabling the requested section and enabling another section. • For **Print** channel requests on the **Print** context **all** sections are enabled by default. It is possible to enable different or multiple sections to control which sections will be outputted.
headers	String	***DEPRECATED. Use the addHeader() function instead.*** **Email sections only**. Used to set custom email headers.
height	String	**Print sections only.** The height of the section using a Measurement string. For example, "11in" is 11 inches. In a Control Script this property can be used to set the height of the section. In other scripts it is only possible to get the height of the section (read-only).
"html()" on page 1612		Gets or sets the HTML of the body element.

Field	Type	Description
margins	Margins	**Print sections only.** Used to set the print margins of a section. You can set the bottom, left, right and top margin using a Measurement string; for example, "2in" sets a margin to 2 inches.
meta	Object	The `meta` object can hold additional information about a section. This property is initially empty, and reset after every record. `Meta` entries can be added in a script (preferably a Control Script). The values stored in `section.meta` can have any type; they are not converted to strings. `Meta` entries can be read in a Standard Script and in Handlebars expressions. For examples, see meta and "Cloning sections" on page 965. **Note:** If a `meta` entry is added by a Standard Script, it cannot be accessed in a Handlebars expression, since Handlebars expressions are evaluated *before* Standard Scripts run.
minPages	Number	Minimum number of pages in the section. The default is 1.
name	String	Used to get or set the name of the section. Note that section names must be unique and that sections cannot have an integer as its name. The name should always include alphanumeric characters. To rename email attachments, use the field `part`.
ownerPassword	String	**Print sections only.** Used to set the owner password for a PDF attachment.* Setting only the owner password creates a secured PDF that can be freely viewed, but cannot be manipulated unless the owner password is provided. (Note that the recipient needs Adobe Acrobat to do this, because the Acrobat Reader does not allow users to enter the owner password.) See "Control Script: Securing PDF attachments" on page 1112.
pagination	"Pagination" on page 1635	**Post Pagination scripts only.** Contains the total page count, sheet count and start/end page numbers of a single section.
part	String	Name for the part. `part` is used to specify where a new part starts and the title for the part. This is used to split Email attachments. The Email output can, for example, attach 3 PDFs generated from the Print context. The part name will be used as the file name for the attachment. See "Parts: splitting and renaming email attachments" on page 1104.
password	String	**Print sections only.** Used to set the user password and owner password for a PDF attachment to the same value. See "Control Script: Securing PDF attachments" on page 1112.*
restartPageNumber	Boolean	**Print sections only.** Enables or disables a restart of the page numbering. When generating Print output this can be used to let page numbering continue over multiple sections. The default value is `false`, meaning that each section will start with page 1 (to emulate behavior of previous versions).

Field	Type	Description
"sheetConfig" on page 1609	SheetConfig	**Print sections only.** Overrides the Master Page, Media and Duplex printing options of first/middle/last/single or all sheets in a Print section.
width	String	**Print sections only.** The width of the section using a Measurement string. For example, "8.5in" is 8.5 inches. In a Control Script this property can be used to change the width of the section. In other scripts it is only possible to get the width of the section (read-only).

* The password(s) should be set on the first Print section when producing a single attachment, or on the first section of each part when producing multiple attachments. Each of the parts (attachments) may have a different (or no) set of passwords.
Passwords set in the Control Script override the password set through the Email PDF password script (see "Email PDF password" on page 678). This allows you to change or remove the password from a specific part. Removal is done by setting the password field to null or "" (empty string).

Note: When a Control Script changes the size of a section, it should also change the size of the linked Media; this is not done automatically. While the output may still look good, a size mismatch can cause an issue if a script or another component assumes that the media size matches the section size.
In case of a size mismatch a preflight will show a warning (see "Doing a Preflight" on page 1075).

Batching settings

Batching improves performance and reduces memory use for templates that generate many pages per record. On the rare occasions where this may cause issues in the output, the following fields can be used to change the settings.

Field	Type	Description
batchedDOMPagesThreshold batchedPrintWindowThreshold	Integer	Cutoff point for batched pagination / batched printing. Default: 15. Batching can be disabled by setting the property to a high value (eg. 99999). Setting it to 0 causes an error.

Functions

Note: For cloned sections, only the `html()` function is available.

Function	Description
"clone()" on page 1611	Clone this section. See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.
	Add a cloned section after this section.
	Add a cloned section before this section. Note that with multiple clones, the next clone is always added **after** the previous clone. With `addBefore()`, the code `original.addBefore(clone1); original.addBefore(clone2);` will result in "clone1, clone2, original". With `addAfter()` the code `original.addAfter(clone1); original.addAfter(clone2);` results in "original, clone1, clone2".
	Email sections only. Used to set custom email headers. The function expects a key and a value (both of type string). For examples, see "Adding custom ESP handling instructions" on page 1679.
"html()" on page 1612	Sets the initial HTML of the body element.
"paginate()" on page 1634	**Post Pagination Scripts only**. Paginates the content of a Print section.

Examples

Conditionally skipping or printing Print sections

This script disables all Print sections and then re-enables one of them, depending on a value in the current record.

```
var printSections = merge.template.contexts.PRINT.sections;
printSections['Section EN'].enabled = false;
printSections['Section FR'].enabled = false;

if(record.fields.Language === 'FR'){
	printSections['Section FR'].enabled = true;
} else {
	printSections['Section EN'].enabled = true;
}
```

Positioning the background of a Print section

These scripts both set the background of a Print section to the same PDF, but they position it differently.

Using abolute positioning

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.position = MediaPosition.ABSOLUTE;
activeSection.background.left = "10mm";
activeSection.background.top = "10mm";
activeSection.background.url = "images/somepage.pdf";
```

Scaling to Media size

```
var activeSection = merge.template.contexts.PRINT.sections['Section 1'];
activeSection.background.source = BackgroundResource.RESOURCE_PDF;
activeSection.background.position = MediaPosition.FIT_TO_MEDIA;
activeSection.background.url = "images/somepage.pdf";
```

See also: "BackgroundResource" on page 1615, "MediaPosition" on page 1618 and "Control Script: Setting a Print section's background" on page 1106.

Cloning a Print section

If documents in the output need to have a variable number of sections, based on data, Print sections can be cloned (copied) on the fly. See "Cloning sections" on page 965 and "Clone Section scripts" on page 1108.

Selecting different sections for Print output and Email PDF attachment

This script selects a different Print section for output, depending on the output channel (Email or Print).

```
var printSections = merge.template.contexts.PRINT.sections;

if(merge.channel === Channel.EMAIL){
    printSections['Section 1'].enabled = false;
    printSections['Section 2'].enabled = true;
}

if(merge.channel === Channel.PRINT){
    printSections['Section 1'].enabled = true;
    printSections['Section 2'].enabled = false;
}
```

Setting the name of Email PDF attachments

This script renames the file name of an attachment by setting the part name of a section (see "Parts: splitting and renaming email attachments" on page 1104).

```
var section = merge.template.contexts.PRINT.sections['Section 1'];
section.part = 'Invoice ' + record.fields['InvoiceNo'];
```

Controlling multiple Email attachments

The following script attaches the following sections to an email:

- Print section 3 + 4 as attachment with continued page numbers
- Print section 6 as separate attachment (also see "Parts: splitting and renaming email attachments" on page 1104)
- Web sections A and B as separate attachment

```
if (channel == Channel.EMAIL) { // only when generating Email output
    if (merge.context.type == ContextType.PRINT) {
        merge.context.sections['Section 1'].enabled = false;
        merge.context.sections['Section 2'].enabled = false;
        merge.context.sections['Section 3'].enabled = true;
        merge.context.sections['Section 3'].part = "PDFAttach1";
        merge.context.sections['Section 4'].enabled = true;
        merge.context.sections['Section 4'].restartPageNumber = false;
        merge.context.sections['Section 5'].enabled = false;
        merge.context.sections['Section 6'].enabled = true;
        merge.context.sections['Section 6'].part = "PDFAttach2";
    }
    else if (merge.context.type == ContextType.WEB) {    // disable whatever is the default section
        merge.context.sections['default Section'].enabled = false;
        merge.context.sections['Section A'].enabled = true;
         merge.context.sections['Section A'].part = "WebPartA";
        merge.context.sections['Section B'].enabled = true;
         merge.context.sections['Section B'].part = "WebPartB";
    }
}
```

Note: For another example, see this how-to: Output sections conditionally.

Note: If the Email PDF Password Script Wizard defines a password, and a template has a Control Script that creates multiple PDF attachments, all the attachments are secured by the same password by default. Using a Control Script, you can set set different passwords for attachments; see "Control Script: Securing PDF attachments" on page 1112.

margins

The `margins` object is used to set a Print section or Master Page's margins in a Control Script (see "Control Scripts" on page 1098 and "Control Script API" on page 1591).
Note that Master Pages are only used in a Print context (see "Master Pages" on page 648).

The `margins` object is retrieved via the `section` object (see "section" on page 1627) or via the `masterpages` array in a template (see "masterpage" on page 1594), respectively.

Fields

Field	Type	Description
	String	These fields allow to set the bottom, left, right and top of a Print section, using a Measurement string, for example: "2in" (this sets a margin to two inches).
	String	These fields allow to set the header and footer of a Master Page, using a Measurement string.

Examples

Setting the margins of a Print section

```
var sectionA = merge.template.contexts.PRINT.sections["Section A"];
sectionA.margins.top = "2in";
sectionA.margins.left = "2in";
sectionA.margins.right = "2in";
sectionA.margins.bottom = "2in";
```

Setting the header and footer of a Master Page

```
var masterA = merge.template.masterpages["Master page A"];
masterA.margins.header = "2in";
masterA.margins.footer = "2in";
```

paginate()

This method of the `section` object (see "section" on page 1627) triggers pagination of the current section. The pagination process re-establishes page boundaries, updates page numbers and page counts, and reapplies Master Pages.
Depending on whether page numbering restarts in each section this may affect the page numbers in other sections as well (see "Configuring page numbers" on page 645).

When the pagination process has ended, the script resumes.

The 'current section' is always a Print section, since this method can only be used in Post Pagination Scripts, and Post Pagination Scripts only run on the Print context (see "Post Pagination Scripts" on page 1114).
You only need to call `merge.section.paginate()` in a Post Pagination Script if the script

has added or removed content to such an extent that the page boundaries need to be renewed.

For an example see: "Creating a Table Of Contents" on page 1116.

Pagination

The Pagination object that is accessed via the `section` object (see "section" on page 1627) contains the total page count, sheet count and start/end page numbers for a **single section**. These properties are read-only and are only available in Post Pagination Scripts (see "Post Pagination Scripts" on page 1114).

For the total page and sheet count of **all Print sections** together use the `merge` object's "Pagination" on the facing page (see also: "merge" below).

For information about where an element is located and the counts of pages and sheets in a **page bundle** - a group of sections in which the page numbering continues - use "info()" on page 1626.

Field	Type	Description
	Number	The total number of pages in this section.
	Number	The index (1-based) of the last page in this section.
	Number	The index (1-based) of the first page in this section.
	Number	The total number of sheets in this section.
	Number	The index (1-based) of the last sheet in this section.
	Number	The index (1-based) of the first sheet in this section.

merge

In Control Scripts, the root level instance of the object `merge` is the entry point from where you can query and change the way contexts are merged. It gives access to the template with all its contexts and sections.
For more information about Control Scripts, see "Control Scripts" on page 1098 and "Control Script API" on page 1591.
Some of the objects are also useful in Post Pagination Scripts; see "Post Pagination Scripts" on page 1114 and "Post Pagination Script API" on page 1619.
For sample scripts, follow the links to the respective objects.

Field	Type	Description
"Channel" on page 1616	Channel	The final output channel: EMAIL, PRINT or WEB. The channel doesn't change when the output consists of different contexts. When generating an email, for example, the channel is EMAIL, even when merging the Print context to attach it to the email.
"context" on page 1620	Context	The context rendered by this merge run. If for one record, different contexts need to be output (for example, when the Print context is attached to an email) a record is merged multiple times: once per context. Per merge run, `merge.context` shows with which context the record is merged.
"Pagination" below	Pagination	Contains the total page count and sheet count of all sections in the Print context after pagination.
"section" on page 1627	Section	In Standard Scripts, this object defines the section that is being merged. **Note!** In Control Scripts, `merge.section` is **only** available when the output channel is **WEB**. To make sure that it is defined, use the following statement: `if (merge.channel == Channel.WEB && merge.context.type == ContextType.WEB) { ... }`. To retrieve any section in a Control Script, use: `merge.template.contexts.ContextType.Section['Section name'];` (for example: `merge.template.contexts.PRINT.sections["Section EN"]`). In Post Pagination Scripts, only Print sections are available.
"template" on page 1613	Template	This object contains the template and all of its contexts. It can be used to find out which contexts are available in the template, using `merge.template.contexts` (see "context" on page 1620) and to manipulate the sections in those contexts (see "section" on page 1627).

Pagination

The Pagination object that is accessed via the `merge` object (see "merge" on the previous page) contains the total page count and sheet count of the **entire Print context** after pagination. These properties are read-only and are only available in Post Pagination Scripts (see "Post Pagination Scripts" on page 1114).

For information about where an element is located and the counts of pages and sheets in a **page bundle** - a group of sections in which the page numbering continues - use "info()" on page 1626.

For the total page count, sheet count and start/end page numbers of a **single section**, use the `section` object's "Pagination" on the previous page (see also: "section" on page 1627).

Field	Type	Description
pageCount	Number	The total number of pages in all Print sections together.
sheetCount	Number	The total number of sheets in all Print sections together.

Generating output

When merged with a record set, the templates made in the Designer can generate three types of output: Print, Email and Web.

You can print and send email from the Designer, but if you want to do this in an automated process, or if you want to produce web pages that are not attached to an email, you need to build Workflow processes with OL Connect tasks. For more information, see: "Workflow processes in OL Connect projects" on page 301.

Print output

Connect supports a number of different types of print outputs. These include:

- AFP (Enterprise edition only)
- IPDS (Enterprise edition only) It produces a spool file, with printer control available through the "Print Manager" on page 1689)
- PCL (Printer Control Language)
- PDF (Portable Document Format)
- PostScript (including the VIPP and VPS variants)
- PPML (Personalized Print Markup Language)

Print *sections*) are part of the Print context. They are meant to be printed directly to a printer or a printer stream/spool file, or to a PDF file (see "Generating Print output" on page 1639).
The Print context can also be added to Email output as a PDF attachment; see "Generating Email output" on page 1673.
When generating output from the Print context, each of the Print sections is added to the output document, one after the other in sequence, for each record.

To dynamically select a section for output, use a Control Script; see "Control Scripts" on page 1098.

There is a number of settings in the Print context and Print sections that have an impact on how the Print context is printed; see "Print settings in the Print context and sections" on page 626.

To split the Print output into several files, see "Splitting printing into more than one file" on page 1657.

Fax output

It is possible to generate Fax output from OL Connect through a Print output job creating PDF/VT output, with some additional Workflow steps.

See "Generating Fax output" on page 1670 for details.

Email output

The Email context outputs HTML email with embedded formatting to an email client through the use of an email server. The HTML generated by this context is meant to be compatible with as many clients and as many devices as possible.

Although the Email context can contain multiple Email templates, only one of them can be merged with each record. Which one is used, depends on a setting; see "Email output settings in the Email context and sections" on page 1675.

Email Output can be generated in two different ways: from the Designer or via Workflow. In both cases, email is sent in a single batch for the whole record set.

To test a template, you can test the scripts (see "Testing scripts" on page 1074) and send a test email first (see "Send (Test) Email" on page 1204), before actually sending the email (see "Generating Email output" on page 1673).

Attachments

Output generated from an Email template, can have the following attachments (and they will be attached in this order):

- Inline image attachments (in the order in which they are encountered in the email).
- The contents of the Print context, in the form of a single PDF attachment. (Compression options for PDF attachments can be specified in the Email context's properties; see "Compressing PDF attachments" on page 668.)
- Attachments specified in the head of the email; an image or a PDF leaflet for example. These can be dynamic. (See "Dynamic attachments: creating file names based on data fields" on page 680.)
- The output of the Web context, as a self-contained HTML file.

Attaching the Print context and/or the Web context is one of the options in the "Send (Test) Email" on page 1204 dialog.

These options are also available in the Create Email Content task in Workflow.

To learn how to attach other files, see "Email attachments" on page 679.

Web output

The Web context outputs an HTML web page that contains the HTML text and all the resources necessary to display it.

Web output can be generated in two different ways: it can be attached to an Email template when generating Email output (see above), or it can be generated using Workflow; see "Generating Web output" on page 1682.

Although the Web context can contain multiple Web pages, only one of them can be merged with each record. Which one is used, depends on a setting; see "Web output settings in the Web context and sections" on page 1683.

Generating Print output

In OL Connect, Print output can be generated directly from the Designer, or via an automated process in Workflow.

Connect supports a number of different types of print outputs. These include:

- AFP (Enterprise edition only)
- IPDS (Enterprise edition only) It produces a spool file, with printer control available through the "Print Manager" on page 1689)
- PCL (Printer Control Language)
- PDF (Portable Document Format)
- PostScript (including the VIPP and VPS variants)
- PPML (Personalized Print Markup Language)

Note: The maximum number of pages is 9999 per record for each Print section.

Generating Print output from the Designer

Print output can only be generated from the Designer when a data set is available (see "Loading data" on page 926). The Designer merges all sections in the Print context (see "Print context" on page 624) with the data set, and generates the output using those data values.

To generate Print output, select **File** from the menu and choose **Print** or **Proof Print**.

Note:
Print uses the Print Service of the Connect Server. The default Connect Server and (if it is secured) an authenticated user must be configured in the Preferences (see

"Connect Servers preferences" on page 1061).
If these settings have not been applied, the "Enter Credentials dialog" on page 1148 is launched to allow selection of Server credentials.

Proof Print generates output directly from the Designer, without using the Print Service of the Connect Server. A Proof Print run won't impact upon production printing

- **File > Print...** allows the following printing options for both "Job Preset" on page 1359 and "Output Presets" on page 1375:
  - Using the **Default** output settings.
    For more details, see "Print using standard print output settings" on page 1643
  - Using the same settings that were **last used** to produce printed output.
    For more details, see "Print using standard print output settings" on page 1643
  - Using **entirely new output settings** set via the Advanced option, which allows selection from a myriad of print output options.

    Note: These settings cannot be saved for later re-use. To do that, one should instead create print presets, which are designed to allow just this behavior.

    For a detailed description see "Print using Advanced Printer Wizard " on page 1651.
  - Using previously saved **Printing Preset** options.
    See "Print Presets" on page 1644 for more details.
- **File > Proof Print...** allows either the default output settings; the last used output settings or previously saved Output Presets.
  For more detailed information on this option see "Print using standard print output settings" on page 1643.

To dynamically select a section for output, use a Control Script; see "Control Scripts" on page 1098.

To split the Print output into several files, see "Splitting printing into more than one file" on page 1657.

Saving printing options in Print Presets

Selecting **File > New > Presets** allows you to create Print Presets (which contain all the printing options), which can be saved for re-use in later print runs. This can be particularly handy when creating special print runs, that need to be run periodically.

These presets make it possible to do such things as filtering and sorting records, grouping documents and splitting the print jobs into smaller print jobs, as well as the more standard selection of printing options, such as binding, OMR markings and the like.

See "Print Presets" on page 1644 for more details.

Generating Print output from Workflow

If you want to generate Print output via an automated process, this means that you have to design a Print process in the Workflow configuration tool.
For information about Connect Print processes in Workflow, see "Print processes with OL Connect tasks" on page 306.

Before creating the print process in the Workflow configuration tool, you will need to create:

- A **template** with a Print context. (See "Creating a template" on page 589)

In addition, the process may use:

- A **data mapping configuration**, or a data model at the very least, if the documents should contain variable data. (See "Creating a new data mapping configuration" on page 338.)
- A **Job Preset** (see "Job Presets" on page 1646). A Job Preset defines where the output goes and makes it possible to filter and sort records, group documents, and add metadata.
- An **Output Preset** (see "Output Presets" on page 1648). An Output Preset can split a print job into smaller print jobs, and set printing options such as binding, OMR markings and the like.

All of these files are made with the Designer and need to be sent to OL Connect Workflow before they can be used in the Workflow process (see "Sending files to Workflow" on page 593.

When the necessary files have been created and sent to Workflow, the next step is to create a process in OL Connect Workflow that generates Print output, using these files (see "Print processes with OL Connect tasks" on page 306).

Tip: For more information about how to create a process in Workflow, please refer to the Online Help of Workflow.

Tip: An easy way to setup a print project in OL Connect, including the print process and the files that it needs, is to use a **Sample Project**. There are two Sample Projects that create a sample print project. See "Sample Projects" on page 1185.

There is also a **Walkthrough sample** that helps you build a Print process for Connect documents in the Workflow Configuration tool by yourself, step-by-step: Creating a Print process in Workflow.

Print settings in a template

There are a number of settings for the Print context and Print sections that have an impact on how Print sections are printed, which **cannot** be made in the Print Wizard or influenced through either a Job Preset or an Output Preset. They are made in and saved with the template.

These settings are:

- **Duplex printing**. Duplex printing has to be enabled for a Print section, in order to print that section on both sides of the paper. The same applies to Mixplex printing.
  See "Enabling double-sided printing (Duplex, Mixplex)" on page 639.
- **Finishing**. The Print context , as well as each of the Print sections, can have its own **Finishing** settings. In printing, Finishing is the way pages are bound together after they are printed.
  See "Setting the binding style for the Print context" on page 627 and "Setting the binding style for a Print section" on page 639.
  Also see "Finishing options" on page 1366 for an explanation of the Finishing options.
- **Bleed**. The margins *around* a page are called the Bleed. It can be used on some printers to ensure that no unprinted edges occur in the final trimmed document.
  See "Page settings: size, margins and bleed" on page 642.
- **Black overprint**. The option to print small black text over other colors is referred to as black overprint. See "Overprint and black overprint" on page 628.

Aborting content creation

You may want the content creation process to be aborted in certain situations; for example, when a template script fails to load remote content. To abort the content creation process, you may raise a fatal error from within a script in the template; see "fatalError(message)" on page 1484.

When a script calls this function in Preview mode, the script that triggers it is marked with an error icon in the Scripts pane, and the given message is displayed in a hint.

When generating output from the Designer, the Designer will log the error and display an error dialog with the given message. Content creation is aborted.

When generating output from Workflow, the entire job fails. Workflow will log the error and execute any follow-up actions that are defined in the On Error tab of the respective OL Connect Content Creation task (All in One, Create Email Content, Create Print Content, Create Preview PDF, Create Web Content and Render Email Content). For more information about how to set up follow-up actions, see Using the On Error tab in the Workflow Help.

Print using standard print output settings

When using either the **File > Print...** or the **File > Proof Print...** option, the Print dialog appears. This dialog allows you to print the template using **Default** printer settings, or the **Last Used** printer settings, or selected Job and Output Presets (either new, or previously saved).

To learn how to create Presets please see Job Presets and Output Presets.

Dialog Interface

- **Job Preset**: Leave the selection at the Default or use the drop-down to create a new or select an existing Job Preset.

  Use the Browse button to open an existing Preset file, or use the Edit button to edit the currently selected Preset file.
- **Output Creation**: Leave the selection at the Default or use the drop-down to create a new or select an existing Output Preset.

  Use the Browse button to open an existing Preset file, or use the Edit button to edit the currently selected Preset file.
- **Summary**: Displays a summary of the settings for the currently selected printing options.

  > **Note:** The Default output type (PDF Output) is actually a built in system Preset, whilst the Last Used settings can likewise be considered an un-named and un-saved Preset.

- **Records** Group:
  - **All**: Outputs all records in the active dataset.
  - **Selection**: Allows selection of a range of records or a custom selection.
    You can specify individual records separated by semi-colons (;) or ranges using dashes.
    For example: 2;4;6-10 would print pages 2, 4, 6, 7, 8, 9 and 10.

- **Copies** Group :
  - **Copies**: Enter the number of output copies you want.
  - **Collate**: When printing multiple copies you can check this checkbox to have the record copies printed together.
    For example, in a three record job the records would print out as 1-1-2-2-3-3 rather than 1-2-3-1-2-3.

Wizard navigation buttons

- **Advanced** button: Click to open the "Print using Advanced Printer Wizard " on page 1651where you can manually change the printing options.

  > **Note:** Any settings made within the **Advanced Print Wizard** do not permanently update any Preset(s) being used.

- **Print** button: Click to produce print output according to the current settings.
- **Cancel** button: Cancels the Print Wizard, without creating any printout.

Print Presets

Print Presets are files that contain **settings for a print job and printer**. These files enable a Workflow Print process to generate Print output automatically (see "Print processes with OL Connect tasks" on page 306), and they allow you to re-use your job and printer settings in the Designer. .

All of OL Connect's print job and printer options are also available in the Advanced Printer Wizard, which appears when you select **File > Print** or **Proof Print** in the menu and then click the **Advanced** button, but the wizard cannot save your settings.

Print Preset types

There are two types of Print Presets: "Job Presets" on page 1646 and "Output Presets" on page 1648. The settings in each are used in a different phase of the print process.

- A **Job Preset** is used by the OL Connect Server after a template and data have been merged. Based on the settings in a Job Preset the server can filter, sort, and group the print content items, add meta data and make finishing settings.
  The resulting print job is then handed to the Weaver engine for output creation (see also: "OL Connect: a peek under the hood" on page 16). This engine generates the final print output file, or files.

- The settings in an **Output Preset** define the destination and all the appropriate print options and printer settings.

How to use Print Presets

Print Presets are not bound to a particular template. They can be used with different Print templates, as long as the data contains the required information.

Job Presets and Output Presets can also be used in different combinations, although not randomly:

- Groups of documents, such as invoices for a specific customer, can be printed separately. This requires that the documents are grouped in advance. In other words, the Separation settings in the Output Preset and the Grouping settings in the Job Preset must match. (See also:"Splitting printing into more than one file" on page 1657.)
- Conditions and output file names in an Output Preset can only make use of those meta data tags that have actually been added to the print job via the Include meta data setting in the Job Preset.
- The finishing options selected in the Job Preset should match the capabilities of the printer selected in the Output Preset.

To apply Print Presets in the **Designer**, select **File > Print** or **Proof Print** and then select an Output Preset and (optional) Job Preset in the Print dialog. It will be indicated if the current data doesn't contain the fields required by the selected Job Preset.

In OL Connect **Workflow** there are special plugins that work with Print Presets; for more information see "Print processes with OL Connect tasks" on page 306.
In order to use Print Presets in a print process they must be sent to Workflow in advance; see "Sending files to Workflow" on page 593.

Creating, opening and saving Print Presets

To create a Print Preset from the **Welcome** screen that appears after startup, choose **New** at the left, then **Presets** at the right.

Tip: Click the Home icon at the top right to reopen the Welcome screen.

Alternatively, select **File > New** from the menu and expand the Presets folder.
Then select the type of print preset that you want to create.

To open a Print Preset, select **File > Open** and browse to the Print Preset.
Or, find the preset using Windows File Explorer and double-click it or press Enter.

Where Print Presets are stored

You can save Print Presets anywhere you want.

As of version 2023.1 Print Presets are no longer automatically stored in OL Connect's workspace folder: `C:\Users\[UserName]\Connect\workspace\configurations`. Where `[username]` is replaced by the appropriate Windows user name.

When it opens, the Designer checks for Print Presets in this folder and suggests to move them to the user's `Documents\OL Connect\Print Presets` folder. You can select a different folder or decide not to have the Presets moved.

Tip: Actually, the path may not begin with 'C:\Users', as this is language-dependent. On a French system, for example, it would be 'C:\Utilisateurs'.

Type `%userprofile%` in a Windows File Explorer and press Enter to open the actual current user's home directory.

Job Presets

A **Job Preset** (or **Job Creation Preset**) is a file that contains settings for the creation of a print job. It is used after a template has been merged with data, but before the actual print output file is produced. Merging a template with data results in a set of **print content items**. Based on the settings in a Job Preset, the OL Connect server can take a number of actions on a set of print content items. The actions are described below.

In addition, **runtime parameters** can be defined in a Job Preset. Filled with a value at runtime, these can be used for comparisons against conditions within the job creation, or in external sorting programs.

Select **File > New > Presets > Job Creation** to create a new Job Preset with the "Job Preset" on page 1359 wizard.

Note: The job creation options are also available in the Advanced Printer Wizard, which appears when you select **File > Print** (or **Proof Print**) and click the **Advanced** button. However, the Advanced Printer Wizard cannot save your settings to file for re-use.

Tip: When a Print template is merged with data by a process in Workflow, the resulting print content items are saved in the OL Connect database. These items can be retrieved later on (using the Retrieve Items task in the Workflow) so that they can be re-used and combined in new print runs. This is called **batching and commingling**. (See also: "Batching and commingling" on page 311.)

Filter and sort

While creating a print job, the OL Connect server can filter out certain print content items based on the conditions defined in a Job Preset (see "Data filtering options" on page 1363). The conditions are evaluated per record and can be based upon such diverse criteria as the value of data fields, the value of runtime parameters (see "Runtime Parameter Options" on page 1360), the binding options used in a source document, or its size.

Note that the *overriding* finishing options, which can also be defined in a Job Preset, are not taken into account while filtering. First, the source documents are filtered. Any overriding finishing options are applied at a later stage.

The remaining items can be sorted, either by the OL Connect Server, based on data fields, or by some external sorting software (see "Sorting options" on page 1368).

Tip: External sorting benefits considerably from runtime parameters, defined in the Job Preset (see "Runtime Parameter Options" on page 1360).

Grouping

While creating a print job, the OL Connect Server can group Print content items into Jobs, Job Segments and Document Sets (see "Grouping options" on page 1372).
This allows to organize content in a certain way, for instance, to gather all foreign mail pieces into one job segment, or to commingle multiple sets of Print content items in one job.

The existence of groups in a print job also allows the Weaver engine, in the next phase of the print process, to separate the output accordingly into different (spool) files (see "Separation" on page 1650).
It is always possible to separate the output on the document level, i.e. output individual documents as separate print files.
In order to split the output on other levels, however, the Print content items must be grouped first.

Include meta data

Meta data can be added to a print job at the Job, Job Segment, Document, Document Set and Page level (see "Meta Data options" on page 1374).

In the next phase of the print process these meta data tags can be used in **output file names** and in various **conditions**.
If the output format is PDF/VT, the meta data will also be included in the print output itself. For AFP (Enterprise edition only), this is optional.

Note that these meta data tags can *only* be used by the engine that creates the final print output (see "OL Connect: a peek under the hood" on page 16). They are not available in a Workflow process and are not in any way related to a process's Metadata in Workflow.

If runtime parameters are defined in the Job Preset (see "Runtime Parameter Options" on page 1360), their value can be used as meta data.

Override finishing options

The Print context and Print sections can have their own finishing options (see "Print settings in the Print context and sections" on page 626.) A template could be setup with a glue binding, for example.
When the printer doesn't support the chosen finishing options you could override the finishing options, and use staple binding, for instance.
More importantly, the Override finishing options option allows you to make **dynamic** finishing settings, and this not only on the Section level but also on the Document, Document Set and Job Segment level.

If runtime parameters are defined in the Job Preset, the passed information can be used in conditions (see "Runtime Parameter Options" on page 1360).

Output Presets

An **Output Preset** (or **Output Creation Preset**) is a file that defines the printer model and output type, and all desired **print options** and **printer settings**. It is used by the Weaver engine (see "OL Connect: a peek under the hood" on page 16), which generates the final print output file (or files).

The input of the Weaver engine is a print job, assembled by the Connect Server. The set or sets of print content items in the print job may have been filtered, sorted and grouped. The server may have added meta data and finishing settings to the job, according to the instructions in a Job Preset (see "Job Presets" on page 1646).

Select **File > New > Presets > Output Creation** to create a new Output Preset with the "Output Presets" on page 1375 wizard.

> **Note:** The Output Creation options are also available in the Advanced Printer Wizard, which appears when you select **File > Print** (or **Proof Print**) and click the **Advanced** button. However, the Advanced Printer Wizard cannot save your settings to file for re-use.

Printer model and output type

An Output Preset specifies the type of printer the output should be suitable for, and determines where that output should then be sent: to a folder, LPR queue or Windows printer (see "Print options" on page 1380).

Printer definitions

A printer model and its capabilities for OL Connect are defined in a printer model definition file (.OL-printerdef). OL Connect comes with a number of such files. However, there are so many different printer models with their own settings and capabilities that it is simply impossible to provide definition files for all of them.
If there is no definition file for your PostScript printer you may create a customized printer definition file yourself using the PostScript Printer Definition (PPD) file provided by the printer manufacturer (see "Dynamic PPD Options" on page 1468).
Other customized Printer Definitions can be obtained from Upland OL Support. They need to be tailor-made based on the information that you provide about both your printer and your specific needs.

Certain printer definition files will work with several types of printers (consecutive versions of a certain printer model, for example). Note that those printers may still have different capabilities.
When creating or selecting Print Presets, always consider the actual capabilities of your specific printer.

Mapping media types to printer trays

An Output Preset can map media types to printer trays (see the "Printer settings" on page 1377 dialog). This option is available for AFP, IPDS, PCL, PPML and PostScript printers that are configured for cut-sheet printing.

> **Note:** AFP and IPDS are only available in OL Connect Enterprise edition.

Print virtual stationery

The virtual stationery in a Print template (also called "Media" on page 652) is in fact an image that is normally only visible in the Designer in the background of the template to make designing a template easier.
If indicated in the Output Preset, the image will also be printed. The output can then be printed on plain paper instead of pre-printed paper. Printing virtual stationery is also very useful when producing PDF output for digital viewing.

Imposition

An Output Preset can tell a printer to print multiple pages on a single sheet (Simplex, or Duplex). This is called **Imposition**, also known as **N-Up** printing.

The two Imposition types are Cut & Stack and Stack by Column (see "Imposition options" on page 1456).
With each Imposition type there is a wide range of options to control the exact layout of the output: the number of pages that go on one sheet, their order, the way they are stacked, the stack depth, the margins, markings, etc.

Note that when Imposition is used, the possibilities to separate the output (see "Separation" below) and to add additional content ("Additional content" below) are limited. Groups and meta data on lower levels than the Job Segment level (as defined in a Job Preset) are no longer available after the Imposition process.

Booklet Imposition

Booklet Imposition defines how to generate booklets in the output (see "Booklet Options" on page 1454).

As printing booklets implies printing multiple pages on a single sheet, this is used in conjunction with some Imposition settings.

Separation

The print output can be split - 'separated' - into discrete portions. An Output Preset can either separate documents, document sets, job segments or jobs, or it can split the output between *counts* of sheets, documents, document sets or job segments. In addition the preset can tell the printer to add slip sheets and/or jog the output. See "Separation options" on page 1477.

Using **document sets**, **job segments** or **jobs** in the Separation settings requires that documents in the print job be grouped that way in advance, following the instructions in a Job Preset (see "Grouping" on page 1647).

> **Note:** Separating the output using Grouping on levels lower than the Job Segment is not possible when you use Imposition.
> With Booklet Imposition, however, the output can be separated on all levels.

Additional content

Via an Output Preset, content (**Text**, **Images**, **Barcodes** and **OMR Marks**) can be added to the output at the time of output creation. The additional content can be either static or

variable (see "Additional Content" on page 1409).

Additional content is particularly useful when you might need to drive custom processes on production machines using either Barcodes or OMR Marks, or if you need to add some last minute additions to the print job via text and/or images.
It helps to keep print production specific additions out of the templates, so templates don't have to be changed if production equipment or postal delivery requirements change.

In additional content you may use the meta data that was included during job creation (see "Include meta data" on page 1647).

> **Note:** When you combine **Imposition** with additional content, the content needs to be added to every page - not once per sheet - in order to have access to the meta data on all levels. If the option **Output once per sheet** is checked, only the meta data at the Job Segment level will be available.

Inserter marks

Inserter marks can be added to the print output in accordance with the settings in an Output Preset (see "Inserter options" on page 1464). The available options are dependent on the selected High Capacity Feeder (HCF) model. These machines are also commonly referred to as Inserters or Folder-Inserters.
If no HCF file is available for a particular inserter machine, adding OMR Marks and barcodes as Additional Content is an alternate way to drive an inserter (see "Additional content" on the previous page).

Print using Advanced Printer Wizard

The **Advanced Printer Wizard** allows you to select from any and all output settings.

The Wizard can be used to generate once-off print runs (either entirely from scratch, or based upon selected pre-existing Presets).

> **Note:** These print runs cannot be saved as presets and can only be replicated in the following print run, using the **Last Used** option.

The output settings are determined by selections made throughout the Wizard. For example, if you want to add Inserter Marks to the output, you select the Add Inserter Marks option on the first page of the Wizard, and the Inserter Options page will then appear later in the Wizard.

The first page of the Advanced Printer Wizard is the "Print options" on page 1380 page.

Adding print output Models to the Print Wizard

Connect comes with several pre-prepared print output Models. These include Advanced Function Presentation (AFP), Intelligent Printer Data Stream (IPDS), Printer Control Language (PCL), Portable Document Format (PDF), Personalized Print Markup Language (PPML) and PostScript (including the PostScript variants of VIPP and VPS).

To keep the Print Wizard interface manageable only a limited range of print output Models are available by default. Additional print output Models can be added to the list at any time, though. They can be selected from the range of pre-prepared Models that come bundled with Connect, or from tailored Models prepared for you by Upland OL, or from PostScript printer Models that you define yourself, through importation of the PostScript Printer Definition (PPD) file.

The following topic describes how to add new Models to the Print Wizard. These Models will always be available in the Print Wizard thereafter, unless deliberately removed.

How to add print output Models within the Print Wizard

Here is how you can add print output Models to the Print Wizard from within the Print Wizard dialog itself.

1. Select **File > Print...** from the menu. The Print dialog will be launched.
2. Click on the **Advanced** button. The Print Wizard will be launched.
3. Click the Settings button at the end of the Model selection list box.
   Select from one of the options:

a. Select **Import Definition** to add a customized Printer Definition. These are customized files created by Upland Software, Inc. in conjunction with you. They will have been customized specifically for your printer(s).

b. Select **Create Definition from PPD** to add a customizable PostScript Printer Model using the printer manufacturer provided PostScript Printer Definition (PPD) file.
   This launches the **PPD Import Wizard**. In the Wizard, do the following.

   - Select the printer's PPD file. Once selected details about that specific printer will load in the information box, for confirmation.

     Note: The PPD information extraction and processing can take some seconds.

- If you are satisfied that the printer information is correct, then press the **Finish** button to complete the importation process.
  This will then launch another dialog to allow setting the new Pinter Model name.

Once a PPD has been selected, then the "Dynamic PPD Options" on page 1468 page will be presented at some point in the Print Wizard. When it appears is dependant upon other choices made in the Wizard, This page provides the functionality for assigning printer specific options to the Connect Template.

See "Dynamic PPD Options" on page 1468 for more details.

c. Select **Edit available printers** to add standard Printer Definitions. These are the standard print outputs installed with Connect.
   - In the **Preferences** dialog that launches, select the print output model(s) to be added to the Print Wizard, then click OK.

d. Select **Export printer definition from output preset** to extract a Printer Definition from a new or modified Printer Model. This launches the Save Printer Definition Dialog.

How to add print output models from within the Designer

Here is how to add print output Models from within the main Designer interface itself.

1. Select **Window > Preferences...** from the menu. Preference dialog is launched.
2. Select **Print > Available Printers** from the options.
3. In the **Available Printers** area, select the print output options to be added to the Print Wizard, then click OK.

Banner pages

In printing, a banner page is a page that may be printed before a print job or before each part of a print job. Banner pages are similar to slip sheets, except that slip sheets are blank, while banners normally contain information such as the date and time, user and job name.

Enabling banner pages

Here's how to add banner pages to your print job in OL Connect.

1. **Group documents.** If you want the banner page to appear only at the start of the job you can skip this step, but to make it appear before each part in a separated job - a job segment, document set... -, the output needs to be grouped at the appropriate level.

You can do this in the Advanced Print Wizard (select **File > Print**, then click **Advanced**) or in a **Job Preset**.

Select **Use grouping**. Then make the settings on the "Grouping options" on page 1372 page.

> **Tip:** Also select **Include metadata** and set meta data tags if you want to print meta data tags on a banner page (see "Variable text on a banner page" below).

2. **Enable banner pages.** You can do this in the Advanced Print Wizard (select **File > Print**, then click **Advanced**) or in an **Output Preset**.

   On the first page of the Advanced Print Wizard or Output Preset, select:

   - **Separation**
   - **Additional content**

   Click Next until you get to the "Separation options" on page 1477 page, where you will find the **Banners** option. To enable banner pages, select where you want them: in front of your job, job segment, document set, or document.

   To have a banner page as the start of every output file, set Banners to the same level as the Separation Settings.

3. **Add content.** Click Next until you get to the "Additional Content" on page 1409 page. This is where you can add content to the banner page. To learn how to add dynamic content, see "Variable text on a banner page" below.

   > **Note:** It is important to **set a condition on each piece of additional content**. Additional content will appear on each page if no condition is set. To make content appear only on banner pages, enter **page.banner** in the Condition field. (See also: "How to set up conditions" on page 1453.)

   > **Note:** The **position** is set as a distance from the left and from the **bottom** of the page.

Variable text on a banner page

The information on a banner page usually contains variable data like the date and time, job name or job ID. You can include document information variables in your banner page texts by using "Content variables" on page 1662.

Content variables provide access to a number of predefined properties, as well as to any **meta data tags** set in the "Meta Data options" on page 1374 page (in the Job Preset or Advanced Print Wizard).

Note: Meta data tags set on the **Job** level are **not** accessible in banner page text (or any other Additional Content).

Metadata tags can be filled with text or with the value of a data field or runtime parameter. In order to use meta data tags based on **runtime parameters**, these runtime parameters must be created in the "Runtime Parameter Options" on page 1360 page of the Job Preset or Advanced Print Wizard.
Remember to also set the actual value of the parameter in the automation tool that you use.

Examples

Date and time

To add a timestamp, you can use the `${system.time}` variable and format it as explained in "Formatting date and number values" on page 1669. .

Example: Created: ${system.time, "yyyy-MM-dd hh:mm"}

The formatting string in this example can be changed to your preference. For instance, ${system.time, "d MMM yy hh:mm", fr} will return the date in French (note the third parameter: **fr**).

Job name or file name

If the job is separated into multiple jobs, segments, document sets or documents, you can include the job name or file name - depending on where you are printing to - on the banner page using the `name` property.

This property is only available at the actual separation level. So, the following example can only be used when separating the job on the document set level:

Example: Job name: ${set.name}

When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job name specified in the Job Name field on the "Print options" on page 1380 page.

When printing to a **directory**, this variable contains the Job Output Mask (specified on the "Print options" on page 1380 page) without the extension.

Job owner

If the job is separated into multiple jobs, segments, document sets or documents, *and* it is printed to a **Windows Printer** or **LPR Queue**, the `owner` property contains the job owner

name specified in the Job Owner Name field on the "Print options" on page 1380 page (in the Output Preset or Advanced Print Wizard).

This property is only available at the actual separation level. So, the following example can only be used when the job is separated at the job segment level.

Example: Job name: ${segment.owner}

Note: When printing to a **directory**, this property contains an empty string.

Number of pages and documents

In the output variables `count.pages`, `count.documents`, `count.sheets`, and `count.sets`, banner pages are included in the count.
To exclude them from the number of pages and documents, subtract the cover pages and documents from the total number of pages and documents, using the output variables: `coverPages` and `coverDocuments`, respectively.

Example: Total number of pages in the document set: ${set.count.pages - set.count.coverPages}
Total number of documents in the document set: ${set.count.documents- set.count.coverDocuments}

Runtime parameter

Let's say you have created a runtime parameter for the job, called `campaign`. You have then added that runtime parameter to the Meta data tags on the Job Segment level.

This parameter is now available for use on a banner page using the `metadata` property of that same level.

Example: Campaign: ${segment.metadata.campaign}

Job ID or Job Set ID

The ID of the Job or Job Set can be added to the banner page as follows.

Example: Job id: ${job.source.name}

Banner pages and inserter marks

Banner pages are often not fed into the inserter, so they should not get any inserter marks.

Inserter marks added using the Add Inserter Marks feature (with .HCF files) never get added to a banner.

If banner pages are fed to the inserter, and the inserter marks are added through Additional Content, you need to add a condition to the inserter marks so that they don't appear on the banner pages.

Tip: The condition `page.content` evaluates to true for all pages that are part of the content, which banner pages are not.

Banner pages and impositioning

When combined with impositioning, the banner page is added to the impositioned job. It always takes the same paper type as the page it is inserted in front of. For instance, when impositioning A4 2-up on a landscape A3, the banner page will be landscape A3.

Banner pages and duplex printing

Banner pages always take the same duplex/simplex setting and paper type as the page they are inserted in front of.

In case of duplex banner pages, the banner page content also appears on the back of the banner page, unless the condition states that the page must be both a banner page and a front page.

Example: page.banner && page.front

Example: Adding a banner page to the Print Transactional Jobs sample project

To see how to add a banner page to the Print Transactional Jobs sample project, please see this step-by-step OL Connect Blog post: https://olresourcecenter.uplandsoftware.com/blog/adding-banner-pages-in-ol-connect/

Splitting printing into more than one file

By default, when Connect saves the print output spool file to a directory, it creates one spool file that contains all the generated documents (one document per data record). It is, however, possible to output one spool file per document, or to create groups of documents and store those in separate spool files. It is also possible to split documents that are longer than a certain number of pages into multiple output files. This topic explains how to do that.

Splitting one document into multiple files

If a *document* - the output of one *record* - can have too many pages so that it cannot fit into one envelope, it can be split automatically into equal parts using the **Repeat sheet configuration** option in the Sheet Configuration dialog. For instructions see "Applying a Master Page to a page in a Print section" on page 651 and "Sheet Configuration dialog" on page 1208.

Grouping documents

Documents in a print job can be grouped on three levels: Job, Job Segment, and Document Set, via a **Job Preset** (see "Job Presets" on page 1646).
For instance, in a mailing destined for recipients in both Canada and the United States, you might want to group the documents by country (Job level) in order to separate the US and CA recipients. You could further sort the mail pieces by state/province (Job Segment level) and then by individual postal codes (Document Set level).

Creating separate output files

To make **each document** or **groups of documents** go into a separate file, a print job needs to be 'separated'. *Separation* is one of the options to set in an **Output Preset** (see "Output Presets" on page 1648). An Output Preset also determines where the output will go.

For example, if a mailing has two groups on the Job level: one of recipients in Canada and one of recipients in the United States, separating the output on the Job level and printing to PDF would result in two PDF files.
If the same documents were also grouped by state/province on the Job Segment level, then splitting the job at the Job Segment level would result in one file per state/province.

Naming output files

When output is split into multiple files, each file probably needs to get a name that identifies the document or group of documents inside it.
For example, if documents are grouped by state or province on the Job Segment level, and the job is split at the Job Segment level, the output files probably need to be named after a state or province.

Here's how to do that with the Advanced Print Wizard.

1. To open the Advanced Print Wizard: select **File > Print**, and in the Print dialog, click the **Advanced** button at the bottom.
2. Select **Directory** in the output options.
3. Check the options **Include meta data** and **Separation**. Click Next.

4. Define **meta data** at the intended separation level. (See "Meta Data options" on page 1374.)
5. Click Next to make the **separation** settings on a following page.
6. Go back to the first page of the wizard and use those meta data in the **Job Output Mask** (see "Output options" on page 1381).

If you are working with a Job Preset and Output Preset:

1. First, define **meta data** at the intended separation level. This is done in the Job Preset (see "Include meta data" on page 1647 and "Meta Data options" on page 1374).
2. In the Output Preset, make the separation settings, and use the meta data in the **Job Output Mask** (see "Output options" on page 1381).

To use meta data in the file name, click the button next to the Job Output Mask field. The Job Output Mask dialog appears. Select a meta data field and click the icon with the plus sign to insert it in the Job Output Mask (see "Output options" on page 1381).

Meta data fields are inserted into the file name as variables. For example: `${segment.metadata.State}` refers to a meta data tag `State` defined on the job segment level.
For the complete list of output variables, see "Print output variables" below.

Note: Only meta data defined on the actual separation level are accessible to the Job Output Mask.

Tip: For an example see this how-to: Configure personalized output file names.

Print output variables

In Print output, File name variables can be used to create dynamic output file names, while Content variables can be used in additional content and in the Conditional field when selectively adding inserts in the "Inserter options" on page 1464.

A number of variables can be used as both File name variables and Content variables:

- The template and file variables.
- If the output is grouped and separated, Content variables (except for the owner and name properties) on the separation level and above are **also available as File name variables**. For example, if the output is grouped on the job segment and document set

level, and is to be separated on the Document Set level, the **set** and **segment** variables can also be used in the Job Output Mask field.

File name variables

File name variables are available in a few places in the "Print options" on page 1380:

- In the **Job Output Mask** and **Job Output Folder** fields when using the Directory option.
- In the **Job Name** field when using the Windows Printer option.

> **Note:** You can name the output file or folder through the Workflow configuration. This is done by setting a Job Info variable in the Workflow configuration (see Job Info variables in the Workflow Online Help) and then referencing it in the "Print options" on page 1380 fields via the variable @automation.JobInfoX@ (where "X" is the number of the Job Info variable).

If the output is to be separated into multiple files, some Output content variables can also be used as Output file variables, depending on the level of separation (see "Content variables" on page 1662).

Template

> **Note:** The variables ${template}, ${template.base} and ${template.name} are only available when printing directly from Designer.

${template}	${template} is a shorthand for ${template.base}_${template.nr,0000}.${template.ext}. It expands to a name based on the template name. A four digit sequence number is added at the end of the base name. The file extension is determined by the selected output format. The 0000 in ${template.nr,0000} is a format pattern that takes care of formatting the number with at least four digits and leading zero's. See "Formatting date and number values" on page 1669, below. **Example:** If the template file is C:\Data\My-Invoices-EN.OL-template which gets printed to PDF, then ${template} expands to My-Invoices-EN_0001.pdf.
${template.base}	Returns the base name of the template, which is the name of the template file without its path and without the trailing file extension. **Example:** If the template file is C:\Data\My-Invoices-EN.OL-template, then ${template.base} expands to My-Invoices-EN .

${template.name}	Returns the name of the template file without the path. **Example:** If the template file is C:\Data\My-Invoices-EN.OL-template, then ${template.name} expands to My-Invoices-EN.OL-template. Note that ${template.name} still includes the extension of the template file (.OL-template in the example above).
${template.nr}	An automatic sequence number belonging to the current output file. It is automatically incremented for each new output file that gets created when Separation has been selected in the Output Preset. It is possible to format the number using a pattern and locale. See "Formatting date and number values" on page 1669, below.
${template.ext}	The extension that corresponds to the chosen output format (in lower case). For example, for PDF output, ${template.ext} would be **pdf**, for PostScript output, ${template.ext} would return **ps**. Note that ${template.ext} does not include a leading dot.

File

${file}	${file} is a shorthand for ${file.base}_${file.nr,0000}.${file.ext}. It expands to an internally generated file name with a four digit sequence number at the end. The file extension is determined by the selected output format. The 0000 in ${file.nr,0000} is a format pattern that takes care of formatting the number with at least four digits including leading zero's. See "Formatting date and number values" on page 1669, below. **Example:** When printing to PDF, ${file} could expand to **merged.5852941188491153960_0001.pdf**.
${file.base}	Expands to a unique, internally generated file name, without a trailing dot or extension.
${file.ext}	The extension that corresponds to the chosen output format, for example **pdf** or **ps**. Note that ${file.ext} does not include a leading dot.
${file.nr}	An automatic sequence number belonging to the current output file. It is automatically incremented for each output file that gets created within the same job. It is possible to format the number using a pattern and locale. See "Formatting date and number values" on page 1669, below.

${file.pageCount}	This variable was introduced for use in Printer Definitions for PostScript printers. As it entails page buffering and could easily lead to Out of Memory errors with big jobs, **usage of this variable in an Output Preset or in the Print Wizard is discouraged. It should be regarded as deprecated.**

Job

${job}	${job} is a shorthand for ${job.base}_${job.nr,0000}.${job.ext}. It expands to a name based on the name of the applied Job Preset (or 'Untitled' if no Job Preset was used). A four digit sequence number is added at the end of the base name. The file extension is determined by the selected output format. The 0000 in ${job.nr,0000} is a format pattern that takes care of formatting the number with at least four digits including leading zero's. See "Formatting date and number values" on page 1669, below.
${job.base}** or **${job.name}	Returns the name of the applied Job Preset without any extension. Expands to **Untitled** if no Job Preset was used.
${job.nr}	An automatic sequence number belonging to the current output file. It is automatically incremented for each new output file that gets created. Note, that multiple output files are created, for example, when output separation has been selected for output creation. It is possible to format the number using a pattern and locale. See "Formatting date and number values" on page 1669, below.
${job.ext}	The extension that corresponds to the chosen output format, for example **pdf** or **ps**. Note that ${job.ext} does not include a leading dot.

Content variables

The variables listed below can be used:

- In text, barcodes, and OMR and Image data in the "Additional Content" on page 1409 page (including its Condition field).
- In the Conditional field, when selectively adding inserts in the "Inserter options" on page 1464 (except the owner and name properties).

If the output is grouped and separated, Content variables (except for the owner and name properties) on the separation level and above are **also available as File name variables**. For example, if the output is grouped on the job segment and document set level, and is to be separated on the Document Set level, the **set** and **segment** variables can also be used in the Job Output Mask field.

Note: When using a variable in a condition, do not wrap it in ${ }. This notation is only necessary in the context of a text (like file names or additional page content).

Tip: If you separate a job into multiple jobs, segments, document sets or documents, you can include the file name - which will be different per part - in the additional text on banner pages using the `name` property. This property is only available at the actual separation level. (See "Banner pages" on page 1653, "Separation options" on page 1477 and "Additional Text Settings" on page 1411.)

Caution: Use `count` variables with caution. They entail higher memory usage in Weaver (the engine that creates Print output). When, for example, ${segment.count.pages} is used in additional text, the Weaver engine has to buffer all pages until it knows how many pages the segment counts in order to include that total in the additional text on each page. With big jobs this could easily lead to Out Of Memory errors.

System

${system.time}	Displays the current system data and/or time. Can be formatted using the "Formatting date and number values" on page 1669, as seen below.

Page

${page.back}	True when the page is on the back of the sheet.
${page.banner}	True when banner page has been selected
${page.cover}	True when cover page has been selected
${page.front}	Boolean indicating whether the page is on the front of the sheet.
${page.height}	The page's height (in points).
${page.landscape}	True when the page's orientation is landscape.
${page.portrait}	True when the page's orientation is portrait.
${page.nr}** or **${page.sequence.document)	Page number in the document.
${page.sequence.sheet}	Page number on the sheet.

${page.sequence.set}	Page number within the document set.
${page.sequence.segment}	Page number within the job segment.
${page.sequence.job}	Page number within the job.
${page.width}	The page's width (in points).

Sheet

${sheet.banner}	True when banner sheet has been selected
${sheet.count.pages}	Total number of pages on the sheet.
${sheet.cover}	True when cover sheet has been selected
${sheet.duplex}	True when printing on the sheet is duplex.
${sheet.height}	The sheet's height (in points).
${sheet.pageDevice}	Array specifying the sheet's Type (e.g. Plain), Weight (e.g. 83), Color (e.g. yellow), Width (in points), Height (in points) and Tumble (e.g. false).
${sheet.nr}** or **${sheet.sequence.document}	Sheet number within the document.
${sheet.sectionName}	Name of the Template Section that the Sheet belongs to.
${sheet.sequence.set}	Sheet number within the document set.
${sheet.sequence.segment}	Sheet number within the job segment.
${sheet.sequence.job}	Sheet number within the job.
${sheet.simplex}	True when printing on the sheet is simplex (one-sided).
${sheet.width}	The sheet's width (in points).

Document

${document.metadata.*propertyname*}** or **${document.metadata['*propertyname*']}	Value of a meta data property of the document. The *propertyname* must have been defined as a Tag Name on the Document Tags tab of the "Meta Data options" on page 1374 page in the *Advanced Print Wizard* or *Job Preset*. **Note:** This variable is only available if Separation based on Document has been selected on the "Separation options" on page 1477 page in the *Advanced Print Wizard* or *Output Preset*.
${document.count.pages}	Total number of pages in the document.
${document.count.sheets}	Total number of sheets in the document.
${document.nr}** or **${document.sequence.set}	Document number within the document set.
${document.sequence.segment}	Document number within the job segment.
${document.sequence.job}	Document number within the job.

The following two variables are only available when the print job is separated at the document level. (See "Separation" on page 1650.)

${document.owner}	When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job owner name specified in the Job Owner Name field on the "Print options" on page 1380 page. When printing to a **directory**, this variable contains an empty string.
${document.name}	When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job name specified in the Job Name field on the "Print options" on page 1380 page. When printing to a **directory**, this variable contains the Job Output Mask without the extension.

Set

${set.metadata.*propertyname*}** or **${set.-metadata['*propertyname*']}	Value of a meta data property of the Document Set. The *propertyname* must have been defined as a Tag Name on the Document Set Tags tab of the "Meta Data options" on page 1374 page in the *Advanced Print Wizard* or *Job Preset*. **Note:** This variable is only available if Separation based on Document Set has been selected on the "Separation options" on page 1477 page in the *Advanced Print Wizard* or *Output Preset*.
${set.count.pages}	Total number of pages in the document set.
${set.count.sheets}	Total number of sheets in the document set.
${set.count.documents}	Total number of documents in the document set.
${set.count.coverPages}	Total number of cover pages in the document set.
${set.count.coverSheets}	Total number of cover sheets in the document set.
${set.count.coverDocuments}	Total number of cover documents in the document set.
${set.nr}** or **${set.sequence.segment}	Document set number within the job segment.
${set.sequence.job}	Document set number within the job.

The following two variables are only available when the print job is separated at the document set level. (See "Separation" on page 1650.)

${set.owner}	When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job owner name specified in the Job Owner Name field on the "Print options" on page 1380 page. When printing to a **directory**, this variable contains an empty string.
${set.name}	When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job name specified in the Job Name field on the "Print options" on page 1380 page. When printing to a **directory**, this variable contains the Job Output Mask without the extension.

Segment

${segment.metadata.*propertyname*}** or **${segment.metadata['*propertyname*']}	Value of a meta data property of the job segment. The *propertyname* must have been defined as a Tag Name on the Job Segement Tags tab of the "Meta Data options" on page 1374 page in the *Advanced Print Wizard* or *Job Preset*. **Note:** This is only available if Separation based on Job Segment or *Split At Exactly n Sheets* has been selected on the "Separation options" on page 1477 page in the *Advanced Print Wizard* or *Output Preset*.
${segment.count.pages}	Total number of pages in the job segment.
${segment.count.sheets}	Total number of sheets in the job segment.
${segment.count.documents}	Total number of documents in the job segment.
${segment.count.sets}	Total number of document sets in the job segment.
${segment.count.coverPages}	Total number of cover pages in the segment.
${segment.count.coverSheets}	Total number of cover sheets in the segment.
${segment.count.coverDocuments}	Total number of cover documents in the segment.
${segment.count.coverSets}	Total number of cover documents in the segment.
${segment.nr}** or **${segment.sequence.job}	Job segment number within the job.

The following two variables are only available when the print job is separated at the segment level. (See "Separation" on page 1650.)

${segment.owner}	When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job owner name specified in the Job Owner Name field on the "Print options" on page 1380 page. When printing to a **directory**, this variable contains an empty string.
${segment.name}	When printing to a **Windows Printer** or **LPR Queue**, this variable contains the job name specified in the Job Name field on the "Print options" on page 1380 page. When printing to a **directory**, this variable contains the Job Output Mask without the extension.

Job

${job.source.name}	The Job ID or Job Set ID.

The following two variables are only available when separation is either not used or set to the job level. (See "Separation" on page 1650.)

${job.owner}	When printing to Windows Printer or LPR Queue, this variable contains the job owner name specified in the Job Owner Name field on the "Print options" on page 1380 page. When printing to a directory, this variable contains an empty string.
${job.name}	When printing to Windows Printer or LPR Queue, this variable contains the job name specified in the Job Name field on the "Print options" on page 1380 page. When printing to a directory, this variable contains the Job Output Mask without the extension.

Template

Note: The variables ${template}, ${template.base} and ${template.name} are only available when printing directly from Designer.

${template}	${template} is a shorthand for ${template.base}_${template.nr,0000}.${template.ext}. It expands to a name based on the template name. A four digit sequence number is added at the end of the base name. The file extension is determined by the selected output format. The 0000 in ${template.nr,0000} is a format pattern that takes care of formatting the number with at least four digits and leading zero's. See "Formatting date and number values" on the next page, below. **Example:** If the template file is C:\Data\My-Invoices-EN.OL-template which gets printed to PDF, then ${template} expands to My-Invoices-EN_0001.pdf.
${template.base}	Returns the base name of the template, which is the name of the template file without its path and without the trailing file extension. **Example:** If the template file is C:\Data\My-Invoices-EN.OL-template, then ${template.base} expands to My-Invoices-EN .
${template.name}	Returns the name of the template file without the path. **Example:** If the template file is C:\Data\My-Invoices-EN.OL-template, then ${template.name} expands to My-Invoices-EN.OL-template. Note that ${template.name} still includes the extension of the template file (.OL-template in the example above).

${template.nr}	An automatic sequence number belonging to the current output file. It is automatically incremented for each new output file that gets created when Separation has been selected in the Output Preset. It is possible to format the number using a pattern and locale. See "Formatting date and number values" below, below.
${template.ext}	The extension that corresponds to the chosen output format (in lower case). For example, for PDF output, ${template.ext} would be **pdf**, for PostScript output, ${template.ext} would return **ps**. Note that ${template.ext} does not include a leading dot.

File

${file}	${file} is a shorthand for ${file.base}_${file.nr,0000}.${file.ext}. It expands to an internally generated file name with a four digit sequence number at the end. The file extension is determined by the selected output format. The 0000 in ${file.nr,0000} is a format pattern that takes care of formatting the number with at least four digits including leading zero's. See "Formatting date and number values" below, below. **Example:** When printing to PDF, ${file} could expand to **merged.5852941188491153960_0001.pdf**.
${file.base}	Expands to a unique, internally generated file name, without a trailing dot or extension.
${file.ext}	The extension that corresponds to the chosen output format, for example **pdf** or **ps**. Note that ${file.ext} does not include a leading dot.
${file.nr}	An automatic sequence number belonging to the current output file. It is automatically incremented for each output file that gets created within the same job. It is possible to format the number using a pattern and locale. See "Formatting date and number values" below, below.
${file.pageCount}	**Deprecated.** This variable was introduced for use in Printer Definitions for PostScript printers. As it entails page buffering and could easily lead to Out of Memory errors with big jobs, **usage of this variable in an Output Preset or in the Print Wizard is discouraged.**

Formatting date and number values

Date and number values can be formatted using an optional pattern and/or locale.

${*expression*}	Do not format.	`${system.time}`	July 4, 2009 12:30:55 PM
${*expression,pattern*}	Apply pattern with system locale	`${system.time, yyyyMMdd-HH:mm:ss}`	20090704-12:30:55
${*expression,pattern,locale*}	Apply pattern with the specified country locale	`${system.time, "dd MMMM yyyy", fr}`	19 décembre 2017
${*expression,,locale*}	Apply a default format with the specified country locale	`${system.time,,fr}>`	19 décembre 2017 12:30:55

It is possible to enclose the values of the pattern and locale in single or double quotes. This is required for including whitespace in a pattern, or when the ${expression} would otherwise be ambiguous.

At run-time, the output engine determines the type of the value yielded by the expression. If this is a number, a number pattern is expected. For date/time-like types, a date pattern is expected. When no pattern is specified, some default format is applied. For other types, it is not possible to specify a pattern or locale.

For patterns and pattern characters see "Date and time patterns" on page 1491 and "Number patterns" on page 1511.

Generating Fax output

It is possible to generate Fax output from OL Connect through the use of PDF/VT output. Here are the details on how to implement such a process.

Required Components

The following components are required in order to output to Fax:

- A OL Connect Image license which includes OL Connect Fax.
- A Job Preset adding the appropriate metadata fields
- An Output preset generating a PDF/VT file.
- A OL Connect Workflow process outputting to the OL Connect Fax task.

Job Preset Configuration

The following metadata fields must be added to the Metadata Options page:

- **FaxNumber**: The phone number where the fax will be sent. Often part of the data.
- **FaxInfo**: The description of the fax in both the OL Connect Fax dialog box and the fax log file.

Output Preset Configuration

The following settings must be used in the Output Preset:

- In the Print Options, a PDF type should be selected, such as Generic PDF.
- In the PDF Options, the PDF Type should be set to PDF/VT

OL Connect Workflow Process

Using a Print template

The following Workflow process will produce output from a Print template and fax it.

- The four regular OL Connect tasks to generate print output:
  - **Execute Data Mapping**
  - **Create Print Content**
  - **Create Job** using the above *Job Preset*.
  - **Create Output** using the above *Output Preset*. The task's **Output Management** must be set to be *Through Workflow*.
- The **OL Connect Fax** connector task set to Passthrough (the first "Document" on the list).

Faxing a PDF (as-is)

If the original data file is a PDF that should be faxed as-is (without any changes to its content), the following Workflow process will do that.

- **Execute Data Mapping**.
  - Make sure your data mapping configuration extracts the FaxNumber and Faxinfo values for the document.
  - Set the **Output Type** to **Output IDs in Metadata**.
  - Tick the option to "Bypass content creation". In this case, the Create Print Content task is not needed.
- **Create Job**.

    - In the *Job Preset*, make sure to create both FaxNumber and FaxOption metadata tags at the **Document** level, and set them to the corresponding fields from your data mapping configuration.
- **Create Output**.
    - Make sure your *Output Preset* is set to output **PDF/VT**.
    - The task's **Output Management** must be set to be *Through Workflow*.
- The **OL Connect Fax** connector task.
    - Set the document to None (the first "Document" on the list) to run in Passthrough mode.

Generating Tags for Image output

It is possible, even easy, to generate specific tags and indexes for OL Connect Image. This can be used to send email, archive with Search or output to image formats.

Required components

The following components are required in order to output to Image:

- A OL Connect Imaging license.
- A Job Preset adding the appropriate meta data fields (see "Job Presets" on page 1646).
- An Output Preset generating a PDF/VT file (see "Output Presets" on page 1648).
- A OL Connect Workflow process outputting to the OL Connect Image task.

Job Preset configuration

For email output, the following meta data fields must be added to the "Meta Data options" on page 1374 page in the Job Preset:

- **ImageSendTo**: Add to have OL Connect Image use the specified field as the E-mail address to which to send the PDF file.
- **ImageSendCc**: Add to have OL Connect Image use the specified field as the E-mail address to which to send the PDF file as a carbon copy (CC).
- **ImageSendBcc**: Add to have OL Connect Image use the specified field as the E-mail address to which to send the PDF file as a blind carbon copy (BCC).
- **ImageSubject**: Add to have OL Connect Image use the specified field as the subject of the email that is sent.

- **ImageBody**: Add to have OL Connect Image use the specified field as the body of the email that is sent.

Note that the PDF file generated by the Print context is sent as an attachment to the email sent using the information above.

OL Connect Search Indexing

For OL Connect Search indexing, you can add your own custom fields. Each field that is not included in the above or in "Generating Fax output" on page 1670 is added as an index for OL Connect Search. For example you could add CustomerID and this would appear as the CustomerID index in Search. Yes, it's that easy!

Output Preset configuration

The following settings must be used in the Output Preset:

- In the "Print options" on page 1380, a PDF type should be selected, such as Generic PDF.
- In the "PDF Options" on page 1390, the PDF Type should be set to PDF/VT

OL Connect Workflow process

The following Workflow process will produce Image output:

- The four regular Connect tasks to generate print output:
  - **Execute Data Mapping**
  - **Create Print Content**
  - **Create Job** using the above *Job Preset*
  - **Create Output** using the above *Output Preset*. The task's **Output Management** must be set to be *Through Workflow*.

  For more information, see "Print processes with OL Connect tasks" on page 306.
- The **OL Connect Image** connector task set to *Passthrough* (the first "Document" on the list). If sending Email, choose the *Send Email* option of the **Output** group. Otherwise, choose *Archive Output*, ensure the output type is PDF, and optionally fill in the OL Connect Search Database tab appropriately.

Generating Email output

The Email context outputs HTML email with embedded formatting to an email client through the use of an email server. The HTML generated by this context is meant to be compatible with as many clients and as many devices as possible.

Email output can be generated in two different ways: from the Designer or via Workflow. Email is sent in a single batch for the whole record set.

Output generated from an Email template, can have the following attachments (and they will be attached in this order):

- Inline image attachments (in the order in which they are encountered in the email).
- The contents of the Print context, in the form of a single PDF attachment. (Compression options for PDF attachments can be specified in the Email context's properties; see "Compressing PDF attachments" on page 668.)
- Attachments specified in the head of the email; an image or a PDF leaflet for example. These can be dynamic. (See "Dynamic attachments: creating file names based on data fields" on page 680.)
- The output of the Web context, as a self-contained HTML file.

Attaching the Print context and/or the Web context is one of the options in the "Send (Test) Email" on page 1204 dialog.

These options are also available in the Create Email Content task in Workflow.

For more information see "Email attachments" on page 679.

Before generating Email output

- Decide on the use of an Email Service Provider; see "Using an ESP with OL Connect" on page 1678.
- Make sure that a data set is loaded, that any necessary files, such as images and attachments, are in place, and that the correct settings are selected (see below).
- If you're using style sheets to style the email, choose whether the styles must be added to the header of the email or to inline style properties as if local formatting was applied (this is also called "embedded CSS"). This setting is made per Email section. See "Email section properties" on page 1199.
- You may want to **rasterize** certain elements, to ensure that most email clients would actually see the output. Rasterizing converts the element to a JPG or PNG image. To rasterize an element, right-click it and select **Rasterize options**. For a JPG image you can set the quality of the resulting image in a percentage.

  Note: Rasterization options are only available for Boxes (elements); see "Boxes" on page 823.

> **Note:** A business graphic in an Email section is rasterized by default and output as PNG image, because email clients usually don't support SVG images.
> SVG images in an Email section give an error in the Preflight window (see "Doing a Preflight" on page 1075).

- For Email output, **PNG** is the preferred image format.
  EPS, PDF, SVG and TIFF images in an Email section are automatically converted to PNG to ensure that they can be seen in the email client.
- Test and validate the output; see "Testing Email output for different email clients" on page 1677.
- If an email server is communicating over HTTPS using a custom certificate, the root certificate may need to be added to OL Connect's certificate store. See Adding custom TLS certificates to OL Connect's Java certificate store.

Email output settings in the Email context and sections

The following settings for the Email context and Email sections have an impact on how the actual emails are sent.

- An Email To Script must be available in the template and refer to a valid email address; see "Email header settings" on page 672. If any record does not have a valid email, this record is skipped automatically when generating email output.

  > **Note:** When you send a test email, the Email To Script will not be used; instead, the email will be sent to the address that you specify in the Send Test Email dialog.

- The sender(s), recipient(s) and the subject can be set using Script Wizards; see "Email header settings" on page 672.
- Default SMTP settings can be set in the preferences; see "Email header settings" on page 672.
- If there are multiple Email sections, only one of them can be merged with each record. Make sure that the correct section has been set as the default; see "Setting a default Email template for output" on page 672.
  To dynamically select a section for output, use a Control Script; see "Control Scripts" on page 1098.
  When printing from the Designer, the currently open Email section will be outputted.

- A plain-text version of the HTML is added to each email if this option is checked in the Email section's properties (see "Email section properties" on page 1199). With new templates this is always the case.
- If style sheets are used to style the email, the styles can be added either to the header of the email or to inline style properties as if local formatting was applied (this is also called "embedded CSS"). See "Email section properties" on page 1199.
- PDF attachments can be compressed to make the files smaller; see "Compressing PDF attachments" on page 668.

Generating Email output from Connect Designer

To generate Email output from the Designer:

1. Open a template with an Email context.
2. Load a data file or database compatible with this template, or open a data mapping configuration. See "Loading data" on page 926.

   If you have an open data mapping configuration and open another data file, the current data mapping configuration will try to retrieve data from the file or database using its own Data Model and extraction logic.

   Note: When generating output with just an open data mapping configuration, the template is merged with the complete sample data file that is part of the data mapping configuration. The output is **not** limited to the number of records shown in the Data Model pane (which is one of the settings in the DataMapper).
3. On the **File** menu, click **Send Email** or **Send Test Email**. In the dialog that appears you can, among other things, attach the Print context or the Web context to the email. See "Send (Test) Email" on page 1204 for a description of all the options. Finally, click OK.

Note:
Send Email sends email through the OL Connect Server. The default OL Connect Server and (if it is secured) an authenticated user must be configured in the Preferences (see "Connect Servers preferences" on page 1061).
If these settings have not been applied, the "Enter Credentials dialog" on page 1148 is launched to allow selection of Server credentials.
Note that even if the software is licensed, a warning will be displayed stating that sending email is not covered by your license. This is a known issue. When you press Finish, the emails will be sent.

Send Test Email generates output directly from the Designer, without using the OL Connect Server. A test email run won't impact upon production emailing.
When you send a test email, the Email To Script will not be used; instead, the email will be sent to the address that you specify in the Send Test Email dialog. If you have a Litmus account, you can enter your Litmus test address. To make the test address appear by default, you can set the default test address in the Email Preferences: select **Window > Preferences**, click the arrow next to **Email**, click **General** and type the test address next to **Email Test address**.
See also: "Testing Email output for different email clients" below.

Generating Email output from Workflow

Here's how to generate Email output from Workflow:

1. Send the required files to OL Connect Workflow: the template that contains the Email context, and any data mapping configuration that you want to use to extract data from a data file. See: "Sending files to Workflow" on page 593.

   Note: If the input data is JSON, a data mapping configuration isn't needed.

2. Create a process in OL Connect Workflow. See: "Email processes with OL Connect tasks" on page 305.

Testing Email output for different email clients

It can't be guaranteed that content will look the same in all email clients, particularly when it comes to older client versions, so it is recommended to always test and validate the output by sending a test email.
An example of a tool that can facilitate testing to ensure a consistent look and feel is **Litmus**: https://litmus.com/email-testing.
If you have a Litmus account, you can enter your Litmus test address when sending a test email (see "Generating Email output from Connect Designer" on the previous page).
To narrow down the field of potential email clients by finding out which email clients are most commonly used by your target audience you can use Litmus' analytics tool (see https://litmus.com/email-analytics).

Aborting content creation

You may want the content creation process to be aborted in certain situations; for example, when a template script fails to load remote content. To abort the content creation process,

you may raise a fatal error from within a script in the template; see "fatalError(message)" on page 1484.

When a script calls this function in Preview mode, the script that triggers it is marked with an error icon in the Scripts pane, and the given message is displayed in a hint.

When generating output from the Designer, the Designer will log the error and display an error dialog with the given message. Content creation is aborted.

When generating output from Workflow, the entire job fails. Workflow will log the error and execute any follow-up actions that are defined in the On Error tab of the respective OL Connect Content Creation task (All in One, Create Email Content, Create Print Content, Create Preview PDF, Create Web Content and Render Email Content). For more information about how to set up follow-up actions, see Using the On Error tab in the Workflow Help.

Using an ESP with OL Connect

An email service provider (ESP) is a company that offers email marketing or bulk email services.

This topic explains why and how to use an ESP with OL Connect.

Reasons to use an ESP

These are a number of reasons why you would need an ESP:

- ESPs ensure a high deliverability, as most ESPs are whitelisted or approved by ISPs (Internet Service Providers) as legitimate email delivery service. So they help you to avoid having mail detected as spam.
- ESPs provide comprehensive tracking options to measure open rates and they log which links were clicked and by who. Typically this information is available via an online dashboard.
- Most ESPs provide Bounce Management options. They will stop sending messages to addresses that return a hard bounce and retry for soft bounces before removing that address.
- EPSs can handle unsubscribes and prevent accidental sends in the future.

Choosing an ESP

The first thing to do to use an ESP with OL Connect is to choose an ESP and create an account.

Mandrillapp.com, a popular ESP, used to have a free account but now requires a paid MailChimp account. Luckily there are plenty of alternatives that provide free accounts

(often capped to a max number of emails per month and sometimes having throttled output).

OL Connect has been tested with: Mandrillap.com, SendGrid (easy user management), MailGun (nearly instant statistics) and MailJet (shows best performance on the free account).

Adding an SMTP Preset for an ESP

After creating an account, add a SMTP settings preset in OL Connect for the chosen ESP, via the Preferences dialog of the Designer (see "Email SMTP settings" on page 674).

Make sure **Use authentication** is checked, and put in your SMTP Username in the box below.

> **Note:** Presets for different ESPs are already available in the list of default presets.

Sending an email with an ESP

Sending email from Designer

To send an email or test email with the use of an ESP, start generating the email as usual (see "Generating Email output" on page 1673). In the Send (Test) Email dialog, pay attention to the following settings:

- In the **Outgoing mail settings** area, select the preset for your ESP in the Presets drop-down.
- In the **Password** box, type the password provided by the ESP.

> **Note:** The ESP might also have a test function you can use. Check the options of your ESP.

Sending email from Workflow

In a Workflow configuration, the OL Connect Render Email Content task provides a way to generate email output without sending it. The email output can be sent using one of the email output tasks, in the same process or in another process. See OL Connect Render Email Content task in the online help of Workflow.

Adding custom ESP handling instructions

Most ESPs allow you to provide custom handling instructions as part of the email message, via custom headers. Typically these include instructions to enable open rate tracking, click through rate tracking and assign tags/categories to messages. Assigning a tag/category allows you to view statistics per email type in the dashboard of the ESP. Note that each ESP has its own notation and instructions.

In a Connect template, custom headers may be added through a Control Script (see "Control Scripts" on page 1098, "Control Script API" on page 1591 and "section" on page 1627).

Custom handling instructions for email that is going to be sent by the OL Connect Mailjet or OL Connect SendGrid Workflow plugin must be specified in the plugin's settings.

The following samples show how to assign a **tag** or **category** to a message for various ESPs.

SendGrid

Dashboard: https://app.sendgrid.com/

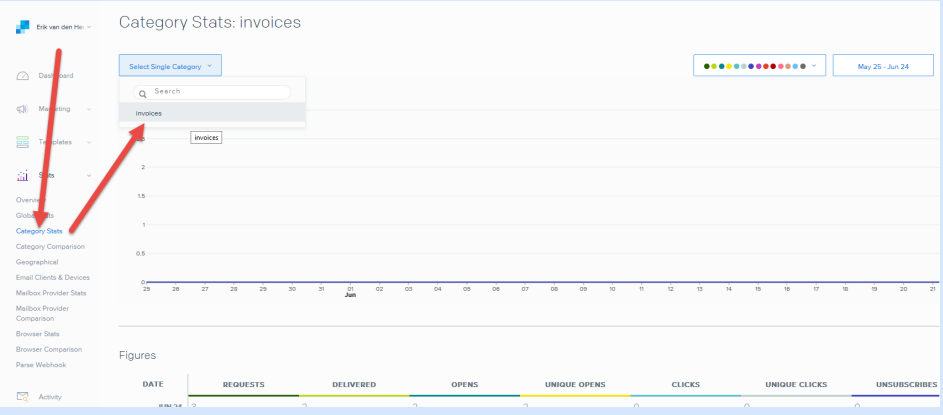
Documentation: https://sendgrid.com/docs/API_Reference/SMTP_API/using_the_smtp_api.html.

Write a Control Script that adds a header with the name *X-SMTPAPI* and sets its content to the required JSON string (for example: *"' category': ['invoices']"*).
For example:

```
var headerObj = {
    "category": ["invoices"]
};
merge.context.sections["Content"].addHeader("X-SMTPAPI", JSON.stringify(headerObj));
```

If the email is going to be sent by the OL Connect SendGrid plugin, the X-SMTPAPI header will not be used. Enter the category or categories in the plugin's settings instead.

Note: Sendgrid strips out their own mail headers. The results need to be verified via their Dashboards (e.g. the Stats section lets you verify the stats for specific categories). Alternatively one can use their Web API to retrieve stats in JSON format. To view the category stats, log in to Sendgrid and choose: Stats > Category Stats > your category name.



MailGun

Dashboard: https://mailgun.com/cp/stats

Documentation: https://documentation.mailgun.com/api-sending.html#sending

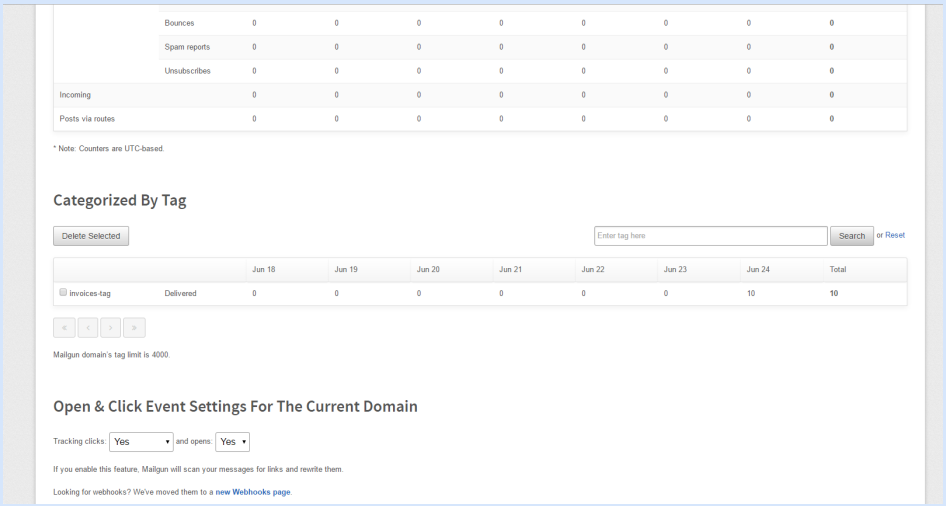
Write a Control Script that adds a header with the name *X-Mailgun-Tag* and sets its content (for example: *invoices*).
For example:

```
merge.context.sections["Content"].addHeader("X-Mailgun-Tag", "invoices");
```

You may specify multiple categories, as follows:

```
var headerObj = {
"category": [
"dogs",
"animals",
"pets",
"mammals"
]
};
var section = merge.context.sections["Content"];
section.addHeader("X-Mailgun-Tag", JSON.stringify(headerObj));
```

Note: The Mailgun tag allows you to view the stats per tag. Mailgun has a quick refresh and stats are available almost instantly.



MailJet

Dashboard: https://app.mailjet.com/dashboard

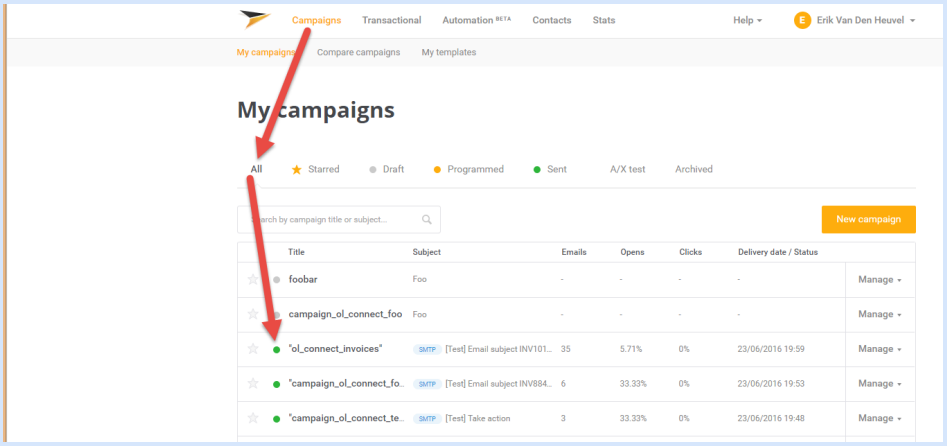
Documentation: https://app.mailjet.com/docs/emails_headers

Write a Control Script that adds a header with the name *X-Mailjet-Campaign* and sets its content to the name of the campaign (for example: *invoices*).
For example:

```
merge.context.sections["Content"].addHeader("X-Mailjet-Campaign", "invoices");
```

If the email is going to be sent by the OL Connect Mailjet plugin, the *X-Mailjet-Campaign* header will not be used. Enter the campaign in the plugin's settings instead.

Note: Mailjet strips out their own mailheaders like X-Mailjet-Campaign. The results can only be verified via the respective campaign stats page in the Mailjet dashboard. There is no need to pre-create the campaign: adding it to the email header via a Control Script auto-generates the campaign. To view the campaign, login to Mailjet and choose: Campaigns > All.



Generating Web output

The Web context outputs one HTML web page. In addition to the HTML text it contains either the resources or references to the resources necessary to display it.
JavaScript files are added to the <head> in the generated HTML file. JavaScript toolboxes like jQuery and its plugins, or MooTools, are useful when you want to implement special features in the web page. (See "Using JavaScript" on page 703)
Style sheets are also added to the <head> and are used just as they would be used in a regular web page. (Also see: "Styling templates with CSS files" on page 881)

Web output can be generated in two different ways: it can be attached to an Email template when generating Email output, or it can be generated using Workflow.

Web output can be generated from the Designer when a data set is available. The data can be retrieved from a database or data file, or from a data mapping configuration.

If you have an open data mapping configuration and open another data file, the current data mapping configuration will try to retrieve data from the file or database using its own Data Model and extraction logic.

Note: When generating output with just an open data mapping configuration, the template is merged with the complete sample data file that is part of the data mapping configuration. The output is **not** limited to the number of records shown in the Data Model pane (which is one of the settings in the DataMapper).

Before generating Web output

- Before actually generating the Web output, you may want to **rasterize** certain elements, such as business graphics. Rasterizing converts the element to a JPG or PNG image. This is very useful to support as many clients as possible. For example, when a heading uses a font type that is not a Web font, converting the heading to JPG instead would ensure that the heading looks the same in all browsers.
  To rasterize an element, right-click it and select **Rasterize options**. For a JPG image you can set the quality of the resulting image in a percentage.

  Note: Rasterization options are only available for Boxes (elements); see "Boxes" on page 823.

- For Web output, **PNG** is the preferred image format.
  EPS, PDF, SVG and TIFF images in a Web section are automatically converted to PNG to ensure that they can be seen in the browser.

Web output settings in the Web context and sections

There are a few settings for the Web context and Web sections that have an impact on the actual web page that is generated.

These settings are:

- Which Web section is the 'default'; see "Setting a default Web page for output" on page 691. When generating Web output, if there are multiple Web sections, only one of them can be merged with each record.
- The title, shortcut icon and meta tags appearing in the web page's header. See "Setting the title, meta data and a shortcut icon" on page 692

Attaching Web output to an Email template

To attach the Web context to Email output:

1. Open a template with a Web context and an Email context.
2. Check the output settings for both contexts; see "Email output settings in the Email context and sections" on page 1675 and "Web output settings in the Web context and sections" on the previous page.

   Note: When adding the Web context to an email, only the default Web section is generated and added to the email as an HTML file. To attach multiple Web sections as separate attachments, you need to create a Control Script that specifies **parts**; see "Control Scripts" on page 1098 and "Control Script API" on page 1591.
3. Load a data file or database compatible with this template. See "Loading data" on page 926.
4. On the **File** menu, click **Send Email** or **Send Test Email**. In the dialog that appears, check the option to attach the Web context to the email. See "Send (Test) Email" on page 1204 for a description of all options.

   Note: When you send a test email, the Email To Script will not be used; instead, the email will be sent to the address that you specify in the Send Test Email dialog.
5. Fill in the dialog and send the emails.

Generating Web output from Workflow

Generating Web output from Workflow requires you to send a template with a Web context to Workflow; see "Sending files to Workflow" on page 593.
Next, you will have to create a process in Workflow that serves one or more web pages; see "Web processes with OL Connect tasks" on page 309.

Note: Athough Workflow can serve both static and dynamic resources to a web browser, it is not meant to be used as a fully featured web server as it is not built for responsiveness nor guaranteed uptime. It is recommended to use a common web server (for example, IIS or Apache) to serve your contents and to let Workflow process things only it can do.
For more information on how to serve HTML and PDF generated by Connect through IIS, watch the Connect with Evie - IIS series.

Note: With a trial or reseller license, Connect Web output is limited to the *localhost*. This means that the Connect Server and Workflow must be on the same workstation in order to create Web output.

Aborting content creation

You may want the content creation process to be aborted in certain situations; for example, when a template script fails to load remote content. To abort the content creation process, you may raise a fatal error from within a script in the template; see "fatalError(message)" on page 1484.

When a script calls this function in Preview mode, the script that triggers it is marked with an error icon in the Scripts pane, and the given message is displayed in a hint.

When generating output from the Designer, the Designer will log the error and display an error dialog with the given message. Content creation is aborted.

When generating output from Workflow, the entire job fails. Workflow will log the error and execute any follow-up actions that are defined in the On Error tab of the respective OL Connect Content Creation task (All in One, Create Email Content, Create Print Content, Create Preview PDF, Create Web Content and Render Email Content). For more information about how to set up follow-up actions, see Using the On Error tab in the Workflow Help.

Optimizing a template

This topic describes some ways to optimize a template in order to speed up the output process. However, the template itself is not the only factor to be taken into account. The server configuration and hardware configuration have an influence on the output speed as well. For advice on server configuration and hardware configuration, see "Performance considerations" on page 33.

Scripts

In the process of output generation, the execution of scripts may take up more time than necessary. Here's what you can do to shorten that time.

- **Use efficient selectors**. Using very precise selectors in scripts will be much faster than using a text selector, especially on large documents. See "Use an ID or class as selector" on page 1078.

- **Optimize your scripts**. Custom scripts with non-optimized loops and unnecessary DOM manipulations can slow down Content Creation. Use the Designer's test facilities to find out which scripts can be improved (see "Testing scripts" on page 1074 and "Optimizing scripts" on page 1078).
- **Only run the necessary scripts**. Normally the Designer will run all scripts for each and every record and section. You can save time in the process of Content Creation by organizing scripts in folders and setting their execution scope or even disabling them (see "Managing scripts" on page 1071). Note that loading a JavaScript library is generally very fast and is only done once for the record set.
- **Use a fast network and internet connection** or avoid loading external or internet resources. Using images, JavaScript or CSS resources located on a slow network or on a slow internet connection will obviously lead to a loss of speed. While we do our best for caching, a document with 5,000 records which queries a page that takes 1 second to return a different image each time will, naturally, slow output generation down by up to 83 minutes.

Images

- Make sure to use **optimized graphic resources**.
  - Avoid using images with transparency where no transparency is needed.
  - Make sure the image resolution is not higher than the printer's print resolution. For example, don't use a 600 dpi picture on a 300 dpi printer. If there is no visible difference in the output, you could use an image with an image resolution that is lower than the print resolution of the printer. This can save a lot of memory usage and processing time. For example, a 150 dpi image is 16 times smaller than a 600 dpi image.
  - Don't use a full color bitmap on a black-and-white printer.
- When a template that contains lots of images is merged with a large record set, the many file requests may slow down the process of output generation. One solution is to combine the images into a single image file (an 'image sprite') and display the part that holds the image. This reduces the number of file requests and can improve the output speed significantly.

Creating and using image sprites

Step 1. Create a file that contains a collection of images.

Static images may go in any type of image file. Store images that need be added dynamically to the template, in one PDF file, one image per page.

There are several tools to combine image files into a singe PDF. **ImageMagick** is one of them. You could use the convert command of the ImageMagick library:

```
convert C:/myimages/*.jpg C:/myimages/image-collection.pdf
```

You could also use **Connect Designer** itself: create a print template with the size of your images and set the page margins to 0. Create a script that loops over your images and adds them to the text flow of the template. Subsequently generate PDF output and use the resulting file as your collection file.

Step 2. Add the file that contains the collection of images to the template's Resources (see "Adding images" on page 854).

Step 3. Display part of the collection file as an image in the template.

- **Static images** that are part of an image file can be displayed via Cascading Style Sheets (CSS). This technique is much used in web design. In this technique, the file that contains a collection of images is called an **image sprite**. The trick is to create a Box (or Div) for each image and give that box an ID (see "Boxes" on page 823). Then use the ID in a style sheet to select the Box and write a style rule (see "Styling templates with CSS files" on page 881) that sets its background image to the image sprite and positions the image.

  For an explanation and examples of this style rule, see MDN Web Docs - Implementing image sprites.

- **Dynamically added images** are loaded in a script. To retrieve one page from a PDF file in a script, add the page parameter to the file path and set that as the source of the image. Here is an example (assuming that the page number is stored in a variable `pageNumber`):

  ```
  var imageStr = "";
  var imagePath = "file:///C:/image-collection.pdf?page=" + pageNumber;
  imageStr += '<img src="' + imagePath + '">';
  results.after(imageStr);
  ```

Runtime parameters

Runtime parameters can pass values from an automation tool - OL Connect Workflow, usually - to a template, data mapping configuration or Job Preset. The actual values of the parameters may be different from one time that a process runs to the next, which means the output could be different even when the same template, data mapping configuration and/or Job Presets are used.

Runtime parameters first have to be defined in the template, data mapping configuration or Job Preset, depending on where they are needed.
Next, the **source** of each runtime parameter value must be set in the automation tool used. E.g in OL Connect Workflow, it should be set in the Runtime Parameters section of the OL Connect task that uses the template, data mapping configuration or Job Preset.

For more information about defining and accessing runtime parameters:

- In a template, see: "Runtime parameters" on page 608.
- In a data mapping configuration, see: "Properties and runtime parameters" on page 387.
- In a Job Preset, see: Runtime Parameter Selection.

Print Manager

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The **Print Manager** provides a means of managing print jobs in a print production environment. It provides the ability to queue jobs, to set hot-folders to allow jobs to run the instant they arrive in a live hot-folder, to re-print jobs, to preview job output, to halt job printing, to organise or re-organise the sequence of production printing, to monitor the current state of printers, to restart paused printers, and such like.

The **Print Manager** is primarily designed to run in conjunction with IPDS output, as it *must* be used to send IPDS output for IPDS printers. It can also be used in conjunction with PCL and PostScript output, but with less options available.

IPDS printing requires a *print stream* and OL Connect only produce *files*, thus the intermediary **Print Manager** is *required* to bridge the communication divide to IPDS printers. The **Print Manager** feeds IPDS output directly to the printer as a print stream, rather than as a file.

Neither PCL nor PostScript printers require a *print stream*, so both those output types can be sent directly to a printer from within Connect. However, you could opt to generate spool files from Connect, and then use the **Print Manager** application to provide some extra print management functionality, such as organizing job priorities, printer queues and job reprints.

The **Print Manager** can be installed standalone (so long as the **MariaDB component** is included in the installation) and can be installed on more than one machine.

Print Manager Introduction Video

Print Manager usage

For a full description of how to use **Print Manager**, see the "Print Manager Interface" below.

Print Manager ports

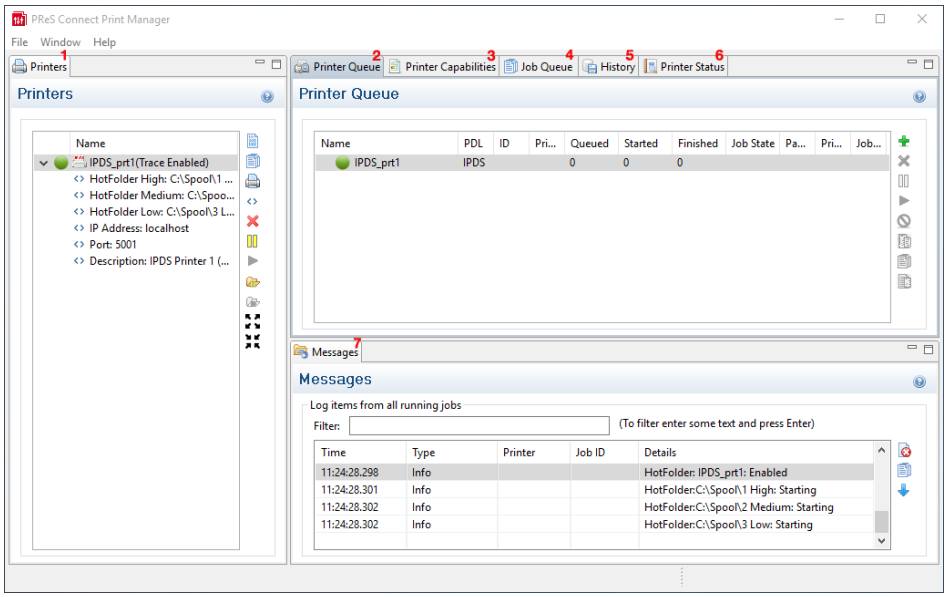
For a list what network ports **Print Manager** requires, see "Network considerations" on page 31.

Print Manager Interface

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The OL Connect **Print Manager** Interface is broken up into multiple tabs. These can be re-organised and moved around and re-sized as desired. Double mouse clicking on a tab will make the tab expand full size, to take up the entirety of the application window, or contract the tab back to original size, if already expanded.

The **Print Manager** display initially appears as in the following image (and can be re-set to this view at any time, via the **Windows > Reset Perspective** menu option):



1. "Printers pane" below
2. "Printer Queue pane" on page 1698
3. "Printer Capabilities pane" on page 1711
4. "Job Queue pane" on page 1707
5. "Job History pane" on page 1710
6. "Printer Status pane" on page 1712
7. "Messages pane" on page 1709

Menu options

- **Windows > Preferences ...**: A variety of **Print Manager** settings can be changed through this menu option. Information about the available preferences can be found here: "Print Manager Preferences" on page 1713.
- **Windows > Reset Perspective**: Reset the **Print Manager** perspective. This returns all the **Print Manager** panes (which can be moved around or re-sized as desired) to their original default positions and sizes.

Printers pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Printers pane allows you to add or remove printers to the **Print Manager**. Once a printer is added here it will be available for use within the "Printer Queue pane" on page 1698.

The following information (all editable via the "Printer Settings Dialog" on the next page) is displayed for each printer:

- **Printer Name**: Displays the name of the printer.
  The background color and the icon to the left of the printer indicates the current status of the queue. The status options are:
  - **Green**: The queue is active and any job sent to it is immediately streamed to the printer.
  - **Yellow**: The queue is paused and jobs will wait before being streamed to the printer.
  - **Red**: The printer is stopped due to a physical error (such as running out of toner) on the printer.
- **<> HotFolder** : Displays the paths of any hot folders that are set.
- **<> IP Address**: Displays either the IP Address or machine name of the IPDS printer.
- **<> Port (IPDS Only)**: Displays the port the printer communicates on.
- **<> Queue (PCL and PostScript Only)**: Displays the printing file type.
- **<> Description**: Displays printer description, if one exists for the printer..

The following controls are available to the right of the printer list, as well as via right-mouse click context menus:

- **Add job to selected printer**: Click to add file(s) to the printer queue for the selected printer. These jobs will then appear in the queue for that printer in the "Printer Queue pane" on page 1698.
- **Add job with Page Range**: Click to load a file directly to the printer queue while specifying a page range to print.
- **Add Printer**: Click to add a new printer using the "Printer Settings Dialog" on the next page.
- **Edit Printer**: Click to edit the currently selected printer using the "Printer Settings Dialog" on the next page.
- **Import Printer**: Click to import a Printer definition from a previously saved external file.
- **Export Printer**: Click to save current Printer definition to an external file.

- **Delete**: Click to delete the currently selected printer.
- **Stop Printer Queue**: Click to stop the currently selected printer(s) queue, which prevents any new jobs from being printed on the stopped printer(s). The printer status icon in the "Printer Queue pane" on page 1698 turns yellow, indicating that the printer is not active.
- **Start Printer Queue**: Click to resume the currently selected printer(s) queue. Any job within the queue assigned to the printer(s) will then start being printed. The printer status icon in the "Printer Queue pane" on page 1698 will turn green, to indicate that the printer is active.
- **Pause HotFolder**: Click to pause the hot folder for the currently selected printer. This does not prevent jobs in the printer queue from generating output - only new jobs will stop being captured in the Hot Folder. This button is only activated when the Hot Folder is currently active.
- **Resume HotFolder**: Click to resume capturing new jobs in the Hot Folder for the currently selected printer. This button is only available when the Hot Folder is paused.
- **Expand**: Click to expand the current printer details within the Printer pane, if not already expanded.
- **Collapse**: Click to hide the current printer details within the Printer pane, if they were currently expanded.
- **Disable Printer**: Click to disable the selected printer(s). This either removes the selected printer(s) from view in the Printer Pane (if the **Hide disabled printers** option is active in the General tab of the Preferences) or highlights the printer(s) in red, to visually indicate that they are disabled.

  > **Note:** If the **Hide disabled printers** option is active in the General tab of the Preferences, any disabled printers will disappear from the Printers Pane.
  > To return them to view, the **Hide disabled printers** option must be de-selected in the Preferences.

- **Enable Printer**: Click to re-enable the selected disabled printer(s).

Printer Settings Dialog

> **Note:** The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

This dialog allows you to add printers to the **Print Manager**, or to modify settings for those already added. The settings differ between printer types, with IPDS printers having more

options than other printers.

Each logical grouping of printer setting appears in its own subsection within the dialog. These can be folded up or down (to make them visible, or hidden) by clicking on the blue title bar atop each subsection.

Once the settings are complete, press the Finish button to apply the new settings.

In the case of adding a printer (via the **Add Printer** option), this adds the newly entered printer to the **Print Manager** and automatically makes it available in the "Printers pane" on page 1691 and the "Printer Queue pane" on page 1698.

If modifying an existing printer (via the **Edit Printer** option) the printer settings are then updated throughout the **Print Manager**.

To test the printer settings, one can run **Check Printer Capabilities** in the "Printer Capabilities pane" on page 1711. The printer options that can be set include:

Printer Details

- **Enable Printer**: Check to enable the printer settings.
  Printers can be Enabled or Disabled using this check box, or via the Enable / Disable options in the "Printers pane" on page 1691.
  - **Printer Type**: Select the type of printer to add, from the drop down list.
  - **Name**: Enter a name that describes the printer in the list.
  - **IP Address**: Enter the IP Address or machine name of the printer.
  - **Port (IPDS Only)**: Enter the port on which the printer communicates.
  - **LPR (PCL and PostScript Only)**: Enter the printer queue.

    Tip: RAW will work as a generic queue option for most PCL and PostScript printers.

    Those PCL and PostScript printers that support multiple queues can have multiple entries in the Print Manager. One for each Queue.
  - **Hot Folder** subsection: Defines folders from which print output files are *automatically* captured and sent to the appropriate printer queue.

- **Enable Hot Folders**: Check to enable hot folders. Browse to set the folders via the Browse button.

  Up to three Hot Folders can be set. One each for **High**, **Medium** and **Low** priority queues.
  An additional "**Set Job on Hold**" option is available which assigns jobs to the target printer but puts them 'On Hold'. They will not be taken from the queue and processed until manually released.

- **IPDS Options** subsection: IPDS specific options contained in 3 tabs. The tabs are:
  - **Extended Options** tab:
    Defines logging of printer activities. The options are:
    - **Page Counters:** IPDS Printers return counters for various categories, from which you can chose.

      Note: The Print Manager will use whichever is selected in this list box to determine when a job is Finished. Once the Counter for this selection equals the total number of pages in the job, the job will be considered Finished and it will display as such within the "Printer Queue pane" on page 1698.

      What choice you make here is dependent upon the capabilities of the printer. Not all printers will provide all options. For example, some printers will only return when the job is Committed, rather than when it has been Received or Stacked.
      The full range of choices available to the Print Manager (but not necessarily the IPDS printer) are:
      - **Received**: The amount of pages received by the printer.
      - **Stacked**: The amount of pages stacked by the printer.
      - **Jam Recovery**: If a paper jam is encountered, the page number at which the printer jam occurred.
      - **Committed**: The number of pages committed to the printer.
      - **Operator View**: This counter contains the number of pages that have passed the view of the printer operator. It increments by

the number of pages on a sheet when the last copy of the sheet passes the view of the printer operator.

- **Ack Pages:** The number of pages to send to the printer for printing before sending an acknowledgement request.

  Acknowledgement requests slow the printing process down, as the printer needs to process the request and return the information.

  The default Ack Pages number is now 10 pages which means the Print Manager will send an acknowledgement request to the printer in increments of 10 pages, until the end of the job. Given the overhead in processing such requests, the smaller the Page number, the slower the printing becomes, but the higher the granularity of the reporting and error correction. If speed is of the essence you might like to increase the number to 100.

  We recommend retaining the default value, due to the granularity of reporting and the error correction this allows.

- **SNMP:** Select this option to allow IPDS printer(s) and Print Manager to communicate back and forth.

  > **Note:** Not all IPDS printers will support SNMP, but most contemporary printers are likely to.

  This option is enabled by default, as it provides the greatest functionality.

- **Extended Options:** Select this to allow access to further Extended Options.
  The Extended Options consist of:

  - **Create Trace File:** Select this to log all communications to a log file.
    The trace file will be sent to the following folder:`C:\Users\[UserName]\Connect\trace\PrintManager`
    The `[UserName]` would be the currently active user's Windows login name.

- **Legacy Mode**: As of Connect 2021.1 outputting *Multipage PDF containers* is the default for IPDS printers.
  To disable Multipage PDF containers, select the **Legacy Mode** option.

  > **Note:** When printing on high volume IPDS printers, it is best if the printer does not pause often. On continuous feed printers in particular, pauses can lead to malfunctions as well as cause delays. It's not just the pause itself, but also the additional time spent whilst the printer slows down and then speeds up again after the pause. The faster the printer, the bigger an issue this becomes.
  >
  > Most modern IPDS printers support Multipage PDF containers, which can be used in such a way to as remove pauses between print jobs.
  >
  > Outputting jobs using Multipage PDF containers means high-end printers can be largely prevented from pausing. So Upland OL recommends using **Legacy Mode** only if your printer does not support Multipage PDF containers, or you run into issues.

- **Twin Printer Mode**: Select this mode when using a twin printer.

- **End of Job Preferences (IPDS only)** tab:
  Select from these options to determine job end processing. More than one option can be selected simultaneously.
  - **Edge Marks:** Select this to add an Edge Mark to the final page of a job. This is commonly used to indicate job separation in Continuous Form printing.
  - **Stack Received Pages:** Add blank pages to the end of the job in order to push the completed job to the stacker.
  - **Print Buffered Pages:** Print any/all buffered Pages. This selection causes the printer to schedule all buffered data for printing prior to sending an Acknowledge Reply.
    Buffered data is page data not reflected by the committed page and

copy counters.

- **Eject to Front Facing:** For cut-sheet media, this selection causes the next received page to be printed as the first page of the next sheet.

- **Error Handling Preferences** tab:
  Select these to **STOP** printing upon encountering any of the following errors (more than one option can be selected):
  - **Undefined Character Check:** If an undefined character is encountered in the job. For example, the Euro symbol might be present in the job, but missing from the printer.
  - **Report Page Position Check:** When printed beyond the edge of the page boundaries. For example, an image that extends beyond the edge of the page.
  - **Handle Errors:** Select for how to handle any/all other optional errors. The choices are:
    - **None** - ignore all other errors.
    - **Auto** - allow default error handling.
    - **Prompt** - prompt for operator action on error.

Description

An optional printer description can be entered here, if desired.

Buttons

- **Finish** button: Click to apply the selected settings to the printer. This applies to both new printers and to update existing printers.
  New printers will appear in the "Printers pane" on page 1691 and the "Job Queue pane" on page 1707.
- **Cancel** button: Click to cancel the selection of printer settings. This exits the dialog, without any changes being made

Printer Queue pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Printer Queue pane displays all the printers available for printing (as set through the "Printers pane" on page 1691), their status and the job queues associated with each printer.

This pane allows you to add or remove print jobs to the printer's queue; to pause and then restart printing (at either the individual job level, or at the printer level); to reprint jobs: to print or reprint selected page ranges and other common print production tasks.

This pane would be the main screen used by print operators to control production printing.

> **Note:** Not all of the functionality on this pane will be available to all printers. Much of the Printer Queue functionality has been specifically designed to work with IPDS printers and is currently only available for IPDS printers and output files.
> Where such restrictions apply will be noted on this page.

Job Status

When a print job is being processed by the **Print Manager** the job is renamed to reflect the current processing step. A new file extension is added to the job, and updated in real time, as described in the following table:

filename.ipds	A print job that is in the queue but which has not yet begun processing.
filename.ipds.**Started**	A print job that is now active. It is currently being processed.
filename.ipds.**Committed**	The print job has successfully been sent to the printer and is now being Committed.
filename.ipds.**Finished**	This print job has been processed.
filename.ipds.**Queued**	This print job has been resubmitted and is queued for processing.

These file extension entries generally match those found in the Printer Queue Table "*Job State*" field.

Printer Queue Table

This table displays information about both the available printers (which can be added or removed via the "Printers pane" on page 1691) and their associated job queues.

Clicking on selected column titles (such as *Name, PDL* and *Priority* columns) will re-sort the Queue display based upon the column selected. Clicking on the column title a second time will reverse the order.

The fields the table contains are broken down as follows:

- **Name**: Displays the name of the printer and beneath it are the names of all the print jobs currently residing in the printer's job queue.

  Print jobs can be added or removed from the printer's queue by a variety of means. Such as dragging and dropping jobs between the "Job Queue pane" on page 1707 and the Printer Queue panes. When a job is added from the Job Queue pane thus, it is then removed from the Job Queue pane thereafter.

  Jobs can also be added by right-click context menu options available at the printer level.
  The options include adding complete jobs via **Add Job**, or partial jobs (IPDS only) via the **Add Job with Page Range** option.

  Jobs can also be dragged from one printer to another within the Printer Queue pane itself.
- **PDL**: Displays the printer language (for both the printer and print jobs).
- **ID**: An unique identity number that is automatically generated for every job.
  The first few letters of the ID are coded to indicate how the job was added to the queue. The relevant codes are as follows:
  - **HTF**: Hot Folder capture. Any job that lands in the printers **Hot Folder** is automatically added to the Printer Queue (at the appropriate priority) and given this code.
  - **DRP**: Jobs "dragged and dropped" directly into the Print Manager (either from Windows Explorer, or from the "Job Queue pane" on page 1707) are given this code.
  - **BRW**: Code for jobs added to the queue via the **Add Job** file browse option.
  - **RSB**: Code used for jobs that have been **Resubmitted** to the printer.
- **Priority**: Displays the priority with which the job is to be printed: Low, Medium or High. Jobs are processed in the queue in the following order: High, then Medium, then Low. Job priorities can be changed on the fly, through the **Set to xxxx Priority** context menu options.
- **Queued**: At the Printer level this displays the number of jobs currently queued to the printer. At the print job level it displays the date and time that the job was added to the queue.
  This counter updates in real time.

- **Started**: At the Printer level this displays the total number of jobs that were started on this printer. At the print job level it displays the time an individual job started printing. Please note, this counter updates in real time, but always adds to the existing Printer total. Use the **Reset Printer Counters** option to re-set the current Printer total to zero.
- **Finished**: At the Printer level this displays the total number of jobs that have finished on this printer. At the print job level it displays the time an individual job finished printing.
  Please note, this counter updates in real time, but always adds to the existing Printer total. Use the **Reset Printer Counters** option to re-set the current Printer total to zero.

  For IPDS printers the "*Page Counters*" entry in the "*Extended Options*" section of the "Printer Settings Dialog" on page 1693 sets the criteria that determines whether a job is Finished or not.

  For example, if the "*Page Counters*" entry is set to Stacked, then the job is only considered Finished when the printer's Stacked pages counter equals the amount of pages in the job.
- **Job State**: Displays the individual print job status. These could be Queued, Finished, Started, and such like.

  Note: For IPDS printers the "*Page Counters*" entry in the "*Extended Options*" section of the "Printer Settings Dialog" on page 1693 sets the criteria that determines whether a job is Finished or not.

  For example, if the "*Page Counters*" entry is set to Stacked, then the job is only considered Finished when the printer's Stacked pages counter equals the amount of pages in the job.
- **Page Range**: Displays the range of pages to be printed. The default is to print the entire page range, but this can be cut down to a specific sub-set of pages via the **Resubmit Range of Pages** option.
- **Printed Pages**: At the print job level this displays the number of printed pages in the current job. At the Printer level this displays total number of pages printed on the printer.
  Please note, this counter updates in real time, but always adds to the existing Printer

total. Use the **Reset Printer Counters** option to re-set the current Printer total to zero.

- **Job Pages**: Displays the total number of pages in the job.

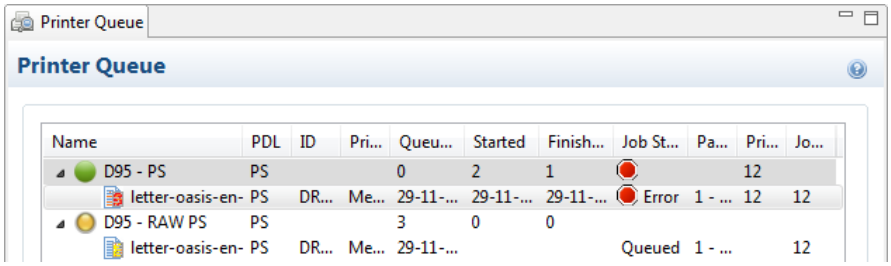
The following options are available via right-mouse click context menus. The options that display will be dependent upon what has been selected (Printer or Job) and the current state of the selected item.

Printer Queue Options

> **Note:** These options will only be made selectable when they are actually currently available.
> If an option is not available to the current selection or setting, it will not be accessible.

- **Start Printer Queue**: Select to start a printer which has been stopped, or which has been recently added.
  Stopped printers have a yellow status icon to indicate that the printer is not currently active. Started printers have a green status icon, to indicate that the print queue is currently active.

  The PostScript printer entry in the following image shows how a printer error state is represented within Print Manager when a printing or connection error occurs. The printer is still active (and thus the printer status icon remains green), whilst the job itself is stopped due to the error. This printer is, in effect, in a paused state whilst it waits upon the error being fixed, or the job being removed from the queue.



- **Stop Printer Queue**: Select to stop an active printer in the printer queue.

  Once selected, that printer's queue is immediately stopped and the printer status icon turns yellow to indicate that the printer is not active.

- **Refresh**: Select to refresh the connection to a printer. This is handy when the communication between printer and Print Manager has encountered issues.

- **Add Job**: Select to add new job(s) to the selected printer. Several jobs can be added at once.
- **Add Job with Page Range** Select to add a specified sub-set of an IPDS job. This launches the "Add Job with Page Range Dialog" on page 1705 in which the job and it's required page range can be set.
- **Remove Finished Jobs**: Select to remove any Finished Jobs from this printers Printer Queue display, making the visual monitoring of active and scheduled jobs easier.
- **Reset Printer Counters**: Select to update the printer information job counters. This resets the number of jobs Queued, Started or Finished.
- **Stack Received Pages**: Select to push out the last page committed into the printer stacker.

Printer Queue Job Options

Note: These options will only be made selectable when they are actually currently available.
If an option is not available to the current selection or setting, it will not be accessible.

- **Preview** button / context menu option: This launches a "Preview pane" on page 1706 that displays what the print job looks like.
- **Pause Job**: Select to temporarily pause the current IPDS job on the printer queue. This option only works for IPDS print jobs and the option is only available when the printer queue is active.
  Once a job is paused, the job queue for this printer is in effect paused as well, until the paused job is either restarted or cancelled.
- **Resume Job**: Select to resume a currently paused IPDS job. This option is only ever active when an IPDS print job is paused in the printer job queue.
- **Cancel Job**: Select to cancel the currently active job (even if paused). This option only works for IPDS print jobs and the option is only available when the printer queue is active.
  Cancelling the active job will also stop the printer, rather than have it continue onto the next job in the print queue.
  A cancelled job can be restated at a late time. It can either print in full, via **Resubmit**, or continue from where the job was cancelled, via **Resubmit Range of Pages** and specifying the starting page as the page the job was cancelled on,

- **Refresh**: Select to refresh the job information. This allows close monitoring of individual jobs.
- **Remove Job**: Select to delete the currently selected job(s) from the Printer Queue.
- **Resubmit Job**: Select to restart the selected job(s) from the beginning. If the original job had been processed using a page range selection, it will be resubmitted using the same page range.
- **Resubmit Range of Pages**: Select to resubmit an IPDS job, but with a specified page range selection. This launches the "Add Job with Page Range Dialog" on the next page in which the job and it's required page range can be set.
  This option is very useful when recovering from a print job failure, when a print job has only been partially printed, and you wish to complete the job. The dialog box will present you with a range of options as to where to re-start the job from.
- **Re-assign to Job Queue**: Select to take the selected job(s) out of the Printer Queue and place it back in the generic Job Queue. The same can be achieved by dragging and dropping the job from the Printer Queue to the Job Queue.
  This option is available when the print Job has yet to be processed. Either when the Printer is in a paused state, or if the Printer is active but has not yet reached this Job in the Printer Queue.
- **Set Job on Hold**: Select to hold this job on the printer queue. When held a job will not print, until it is once again activated by assigning a new job Priority. At which point it becomes active again, and will be process at the selected priority.
- **Set to Low Priority**: Select this to set the job priority to Low. This will immediately drop the job below any/all Medium or High priority jobs in the queue, as well as taking a job off Hold, if it had been held.
- **Set to Medium Priority**: Select this to set the job priority to Medium. This will immediately drop the job below any/all High priority jobs in the queue, but above all Low priority jobs, as well as taking a job off Hold, if it had been held.
- **Set to High Priority**: Select this to set the job priority as High. This will immediately push the job above any/all Medium or Low priority jobs in the queue, as well as taking a job off Hold, if it had been held.
- **Directory**: Select to open a Windows Explorer window set to the directory containing the job file.

- **Set Finished**: Select this to change a Committed job status to Finished. This option is only available to Committed jobs. It is a way to manually set a completed job to Finished, when the Print Manager has not automatically done so (as can happen, in some scenarios).

Add Job with Page Range Dialog

> **Note:** The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The **Add Job with Page Range** dialog box works with IPDS output only. It allows the selection of a page range to print, from IPDS spool files.

It can be used to print a section from within the middle of the file, or it could be used to select a starting point and then print all the pages thereafter.

Job Name details section:

- **File Name**: Paste the print job filename and folder into the entry box, or use the Browse button to browse for the print job.
- **Start Page**: Select the page number at which to start printing.
- **Last Page**: Select the ending page number.
  This defaults to the last page in the print job, but can be set to any page that follows the **Start Page** selection.
- Preview **Preview** button : Use this button to launch the "Preview pane" on the facing page.
  The preview will display the current job, starting at the page selected as the **Start Page** in the page range.
- **Priority**: Select the job printing priority.

Select Start Page details section: ***This section only appears when resubmitting a previously processed IPDS job.*** It allows you to set the starting page based upon the following criteria:

- **Start Page**: The first page of the job. If the job was selected with a page range, this would be set to the beginning of that previously selected page range. Otherwise it will always be page 1.
- **Pages Committed**: Select this to start from the last page Committed to the Printer.

- **Page Jammed Recovery**: Select this to start from the Paper Jam Recovery page counter entry.
- **Pages Received**: Select this to start from the Pages Received counter entry.
- **Operator View**: Select this to start from the Operator View counter entry.
- **Pages Stacked**: Select this to start from the Pages Stacked counter entry.

Tip: When an issue arises with the printer and you wish to restart the job from a recovery point, it would be best to first "Preview pane" below the page at the selected page counter, to confirm that the selected page counter entry is accurate.

In reality, you might need to move the starting page entry backwards or forwards some few pages in order to cater for pages which have been printed, but destroyed in the process.

Preview pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Preview pane displays previews of print jobs from with the "Printer Queue pane" on page 1698, the "Job Queue pane" on the next page, the "Job History pane" on page 1710 and the "Add Job with Page Range Dialog" on the previous page.

Note: The Preview will work "out of the box" for both PCL files and standard IPDS files but not for PostScript files, nor continuous IPDS output created as Multipage PDF containers. Previewing either of these types of files requires the **Ghostscript** software be installed first.

Ghostscript is a free industry standard PostScript interpreter/renderer that can be downloaded from here: https://www.ghostscript.com/download/

Select the **Ghostscript AGPL Release** version that best matches your Operating System, then download that installer and run. PostScript and continuous IPDS output Previews will then be available, the next time you launch the Print Manager.

The Preview pane allows you to view all the pages in a print job, via the Preview toolbar:

You can step through print jobs using the page step buttons, or move straight to a specific page by entering a number in the page number box. You can also customize the preview viewing settings as you wish.

Note: Print jobs featuring associated Connect "Output Presets" on page 1648 which have "Imposition options" on page 1456 set to Landscape will not **Preview** in Landscape, but rather in Portrait mode. The printout is unaffected, however.

Job Queue pane

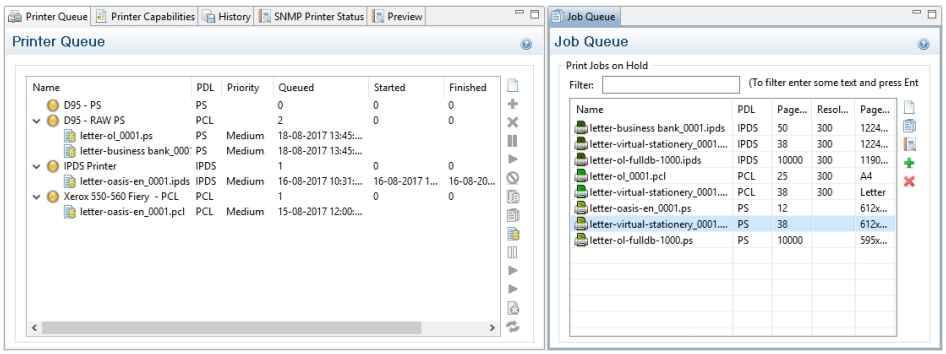
Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Job Queue pane is a centralized location designed to allow the consolidation and management of print Jobs.

Jobs are not printed from here. The Job Queue could best be considered a staging area, where one can consolidate jobs in preparation for production printing.

You can add Jobs to the queue by "dragging and dropping" them from a Windows Explorer window, or from the "Printer Queue pane" on page 1698, or by using the **Add Job** option.

Once in the Job queue, jobs can be previewed to confirm the contents are correct, and selected for printing by "dragging and dropping" the print Job into the "Printer Queue pane" on page 1698.
To do this, you must have both screens visible at once, as seen here:



Note: Print Jobs ***are not actually scheduled for printing*** in the Job Queue. That is done in the "Printer Queue pane" on page 1698.
The Job Queue is a job holding/staging queue, rather than a production printing queue.

- **Filter**: Use this search functionality to reduce the data-set displayed in the table. Type in a search word and then hit Enter to filter the table using the entered search

word. For example, select "business" to display only those entries that feature the text "business".

To clear the filtering, delete the contents of the Filter box and press Enter.

Job Queue Table

- **Name**: The name of the job filename, as captured or added to the Print Manager.
- **ID**: An unique identity number that is automatically generated for every job.
  The first few letters of the ID are coded to indicate how the job was added to the queue. The relevant codes are as follows:
  - **HTF**: Hot Folder capture. Any job that lands in the printers **Hot Folder** is automatically added to the Printer Queue (at the appropriate priority) and given this code.
  - **DRP**: Jobs "dragged and dropped" directly into the Print Manager (either from Windows Explorer, or from the "Job Queue pane" on the previous page) are given this code.
  - **BRW**: Code for jobs added to the queue via the **Add Job** file browse option.
  - **RSB**: Code used for jobs that have been **Resubmitted** to the printer.
- **PDL**: Displays the job printer language.
- **Page Count**: Displays the total number of pages in the job.
- **Resolution**: Displays the resolution, in DPI, of the job.
- **Page Size**: Displays the dimensions of the job.

To the right of the table and also via right-mouse click context menu within the table, the following options are available:

- **Preview** button / context menu option: This launches a "Preview pane" on page 1706 that displays what the print job looks like.
- **Copy selected items to clipboard** button / **Copy** context menu option: This copies job information (job name and folder path) to the clipboard.
- **Properties** button / context menu option: This launches a job Details Window that displays detailed information about the makeup of the print job. Once the Details Window is open it remains atop the Print Manager thereafter, to allow simple comparisons between jobs.
  To close the Details Window, put the focus upon the Details Window by clicking

anywhere within it, and then press the Escape key ("Esc").

- **Add Job** button / context menu option: Click this to add a print Job to the Job Queue.
- **Remove Job** button / context menu option: Click this to remove a print Job from the Job Queue.
- **Refresh** button / context menu option: Refreshes the job(s) to update current page count and the like.
- **Directory** button / context menu option: Opens the folder containing the selected file.

  Note: If multiple files are selected, only the folder of the first file selected will be opened.

Messages pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Messages pane displays any notice, warning or error message that is generated by the **Print Manager**. This may include indications that jobs have finished or started, or that errors have occurred.

The level of logging can be set in the Print Manager Preferences screen, under General preferences.

- **Filter**: Use this search functionality to reduce the data-set displayed in the table. Type in a search word and then hit Enter to filter the table using the entered search word. For example, select "business" to display only those entries that feature the text "business".

  To clear the filtering, delete the contents of the Filter box and press Enter.
- **Messages** table: This contains the message details. Which are:
  - **Time**: The message time-stamp.
  - **Type**: The type of message.
  - **Printer**: The relevant printer, if applicable.

- **Job ID**: The relevant Job ID, if applicable.
- **Details**: The actual message itself.

The following controls are available to the right of the Messages table, or via right mouse click context menu:

- **Clear Log Items** button / **Clear Log** context menu option: Click to clear the current log. This does not delete log files, but it does delete the messages displayed onscreen.
- **Copy selected items to clipboard** button / **Copy** context menu option: Click to copy all messages to the clipboard. Useful for pasting messages into external software.
- / **Always select last log entry** button / **Tail** context menu options: Click to toggle auto-scrolling on/off. Auto-scrolling ensures the most recent message is always displayed when it is added to the log.

Job History pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Job History pane displays a list of print jobs that have been processed through the **Print Manager** (except for those which have been explicitly removed).

How many print jobs are displayed (and the order of the display) is determined by the **History** settings in the Print Manager "General Settings" on page 1716. The default is for the last 500 jobs to be displayed.

- **Filter**: Use this search functionality to reduce the data-set displayed in the table. Type in a search word and then hit Enter to filter the table using the entered search word. For example, select "business" to display only those entries that feature the text "business".

  To clear the filtering, delete the contents of the Filter box and press Enter.

  Note: The Filter applies to the entire data set, so if you were to search for the text "Med" the filter would return all those jobs that featured "Med" in the title, plus all those that in any hot folders that featured "Med", plus all jobs whose priority had been set to "Medium".

The search is case insensitive. Thus a search for "med" would return "Medium", as well as "medium" values.

- **History Log**: Displays details on jobs run through the **Print Manager**. The history can be sorted by either processing date or printer, by selecting the column title.
  - **Date**: The date and time when the job was started.
  - **Printer**: The name of the printer the job was printed to.
  - **ID**: The internal ID of the job.
  - **Job**: The name of the print spool file.
  - **Path**: The path from where the job was captured or added.
  - **Start Page**: The page from where the printing started.
  - **Pages**: The pages printed and the total number of pages in the job.
  - **Status**: The final status of the job, such as Cancelled or Finished.
  - **Priority**: The job queue priority the job was printed with.

To the right of the table and also via right-mouse click context menu within the table, the following options are available:

- **Preview** button / context menu option: This launches a "Preview pane" on page 1706 that displays what the print job looks like.
- **Copy selected items to clipboard** button / **Copy** context menu option: Click to copy all messages to the clipboard to paste them in an external software.
- **Always select last log entry** button / **Tail** context menu option: Click to toggle auto-scrolling so that the last message is always displayed when it appears.
- **Remove Job** button / context menu option: Click to delete job from history.

Printer Capabilities pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The Printer Capabilities pane displays the technical capabilities of any printer that has been added to the "Printers pane" on page 1691.

- **Filter**: Type in a filter for a specific capability name and press Enter to filter the results below.
  To clear the filter, delete the contents of Filter edit box, and press Enter.
- **Name**: Displays the list of printers and their capabilities. By default, capabilities are not displayed for performance reason. To display the capabilities, right-click on the printer and select **"Check Printer Capabilities"** or click on the button to the right.

  Note: Requesting a printer's capabilities can take some time, and not all printers will support it.

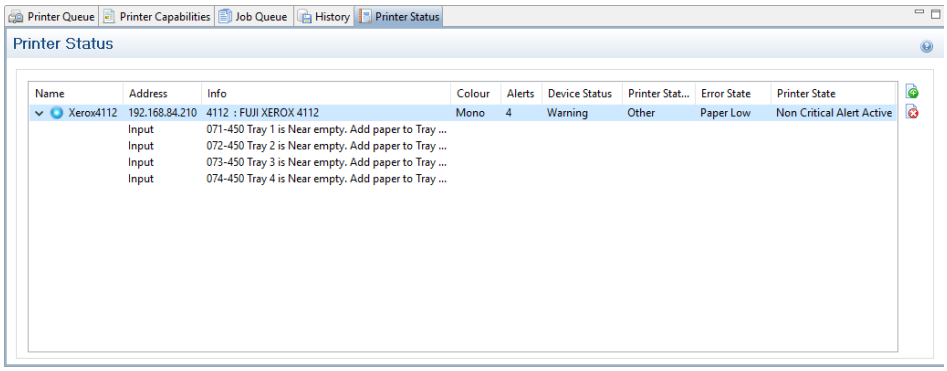
  For PostScript and PCL printers you should first set up monitoring of the printer, in the "Printer Status pane" below.

  - **Check Printer Capabilities** button / context menu option: Each printer capability is listed under the Printer, in separate divisions. Capabilities are read directly from the printer.
  - **Export Printer Capabilities** button / context menu option: Export the printer capabilities to JSON file. This contains all the relevant printer capabilities, in a text file format.
  - / **Expand / Collapse** button / context menu option: Expand or contract the printer details.

Printer Status pane

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

This pane can be used to monitor the state of PostScript and PCL printers. It contains general printer information, such as the printer's B&W or color settings, as well as the quantity of paper in trays, any outstanding alerts or errors and whether the printer is currently in use, or awaiting a job. An example of this can be seen in the following image:

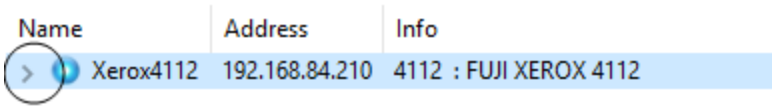


To **Start monitoring a printer** select the Start icon to the right of the status display.

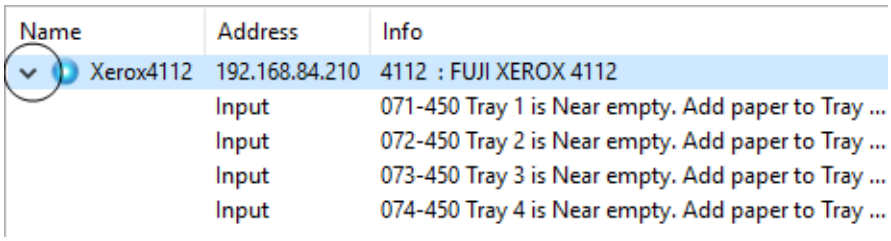
To **Stop monitoring a printer** select the Stop icon to the right of the status display.

Once a printer is being monitored, the details can expanded and the individual printer trays displayed. These can be done as follows:

- **To expand the printer details**, press the right arrow besides the printer Name, as seen in the following image.



- **To reduce the printer details**, press the down arrow besides the printer Name, as seen in the following image.



Note: If SNMP monitoring is enabled on a printer, then printer issues will pause the "Job Queue pane" on page 1707 for that printer.

Print Manager Preferences

Note: The **Print Manager** is only available in the **Enterprise** edition of OL Connect.

The **Print Manager** *Windows > Preferences ...* menu option allows adjustment of various **Print Manager** preferences.

There are several groups of preferences which can be changed, or reset to default settings at any time. These groups are:

1. "Database Connection preferences" below
2. "General Settings" on page 1716
3. "Language Setting" on page 1719
4. "Log Settings" on page 1719
5. "SNMP Connection Settings" on page 1721

Database Connection preferences

Dialog used to change the OL Connect back-end Database.
This dialog supports the swapping of the back-end database between various vendor databases. Note, however, that the alternate vendor database(s) *must already be installed and available* in order to swap to them.

This is not a migration tool. It is a simple connection tool, that enables shifting to a different back-end database. Any existing data will **not** be transferred/ migrated between the databases, and any existing File Stores will be cleansed by the Clean-up Service after the swap.

When shifting to a different back-end database, the changes won't be applied until OL Connect is restarted. Including the Connect services. A full machine restart is recommended, as this provides the cleanest restart of all the services.

- **Basic Connection Settings** selections:
  - **Database vendor**: Select the database type from the drop down list.

    Note: Moving from one vendor database to another will reset all screen selections to defaults, regardless of what may have been previously selected.

  - **Database URL**: This is a read-only summation of the current database connection settings.

    Tip: If the **Test Connection** button shows that the database cannot be successfully connected to using the selected settings, then the contents of this field could be used to try to connect to the database outside of OL Connect. This should help in determining and refining the acceptable connection options.

- **Hostname**: Enter the IP Address or alias of the server where database resides.
- **Use default port** checkbox: Select to force the use of the default port for the selected database type. This will enable the **Database Instance Name** selection, when used with a Microsoft SQL Server.
- **Database Instance Name**: Enter an existing Microsoft SQL Server's instance name.
  This option only applies to existing Microsoft SQL Server instances, and not for MariaDB or MySQL.
  Note that the Microsoft SQL Server instance name must meet Microsoft's require-ments.
- **Port**: Enter Port number. The initial selections are those which the vendors use by default.
- **Schema**: The individual database schema, within the vendor database. The default is **olconnect**.

  > **Note:** If a previously non-existent schema were chosen here, then a new schema of that name will be created within the database when the back-end database swap is applied. The tables within that schema, though, will not be created until Connect is restarted.

- **Use Windows Authentication**: Select to use Windows authenticate for database login. This option is only available for Microsoft SQL Server back-end databases. Selecting this option disables the Username/Password fields.
- **Username**: Enter the database login username. This was initially set during the installation, and the default is **olconnect**.

  > **Tip:** It is considered best practice for this user to have root privileges.

- **Password**: Enter the password associated with selected username.

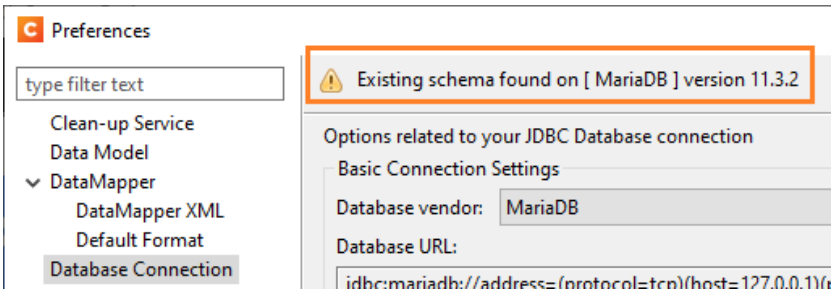
- **Advanced Connection Settings** selections:
  - **Maximum concurrent threads**: This option sets the maximum database threads. The maximum setting is determined by the specific capabilities of the machine Connect is installed upon (CPU speed and the amount of cores being the major determinants).

    > **Tip:** Leaving this value set to the default maximum *should* be the best option in most circumstances.
    > We recommended this entry be left at the default value.

- **Custom database parameters** table: These are extra parameters which are appended to the database connection URL. The default values are those which have been determined to be useful in connecting to specific vendor databases. For example, in order to use Integrated Authentication with Windows, specify the property `integratedAuthentication` with the value `true`, which adds `integratedAuthentication=true` to the URL.
    - **Property**: These are free field text fields.

      Note: These fields and their associated values get appended to the JDBC connection and therefore must follow all rules regarding acceptable URL addresses for such.

    - **Value**: The value applied to the associated Property.
    - **Add**: Used to add extra Property values to the table.
    - **Delete**: Used to remove existing Property values from the table.
- **Test Connection**: Use to test if current connection settings will connect to the specified database.
  If successful, this will display the version of the database at the top of the Preferences page.



- **Restore Defaults**: Will restore the settings to OL Connect HyperSQL standard defaults.
- **Apply**: When a database connection is confirmed as correct this button becomes active, and is used to actually apply the database swap.

General Settings

- **Hot Folder** group: Define a global hot folder that captures any jobs placed within that folder and places them in the Job Queue pane (which can be considered a holding queue).
  This does not automatically print the jobs, but it does allow them to then be dragged

across to the appropriate production printer in the "Printer Queue pane" on page 1698.

- **Enable Hot Folder** checkbox: Check to enable the global Hot Folder, to have jobs added to the Job Queue pane.
- **Monitored Folder** entry: This entry is only available if the **Enable Hot Folder** checkbox selected.
  Enter the path for the global Hot Folder, or click the Browse button to browse for the folder.
- **Add Time Stamp** group: Section used to add a time stamp to any files that arrive in the Hot Folder, in order to ensure file uniqueness.
  - **Add Time Stamp** checkbox: Select to have a time stamp added to any files as they arrive in the Hot Folder.
  - **Format** selection: Select the format of the time stamp to add to files in the Hot Folder.

- **Session Retries**: The number of times Print Manager will try to reconnect to the printer before it stops trying and instead produce a timeout error in the messages.
- **Printer Timeout (sec)**: The startup time (in seconds) allowed to the printer before a timeout error is reported in the messages.
- **Printer Ack Timeout (sec)**: The time (in seconds) that the Print Manager waits for a response (an **Ack**nowledgement) from the printer after sending a response request.
- **Retry Delay**: The time (in seconds) that the Print Manager waits before retrying a job after a timeout.
- **Last print job idle delay**: The time the Print Manager will sit idle after a job, waiting for another job to arrive.
  To avoid having to wait after the last job, you could right-click the printer in the Printer Queue and select "Stack Received Pages" to push out the last page committed into the printer stacker. (See: "Printer Queue Options" on page 1702)
- **IPDS Streamed Reader** checkbox: Select this option if difficulties are experienced with IPDS printers periodically freezing whilst **Print Manager** reads responses from the printer.
- **Log Level**: Use the drop-down to select which Print Manager logging level to use.

  Note: This logging relates to Print Manager job processing, and not to internal Print Manger debugging. Settings for the internal logs are catered for in the generic "Log Settings" on page 1719 pages.

The selection of logging chosen here will determine what is displayed in the Print Wizard Message Pane.
Higher logging levels such as "Debug" and "Status" provide more details but will have an impact in terms of processing time and resources required.
You might need to adjust this setting based upon your production facilities, production speeds and logging requirements. Some sites might decide that "Error" logging levels (which only records Errors, and thus should be the minimal logging option) are all they need.

> **Tip:** We recommend leaving the logging level set at the default level of "Info", unless otherwise needed.

- **Printer Preferences** group:
  - **Hide disabled printers**: Select to have any printers disabled in the "Printer Settings Dialog" on page 1693 not appear in the "Printers pane" on page 1691.

    > **Note:** Not being able to see a printer that exists in the "Printer Settings Dialog" on page 1693 within the "Printers pane" on page 1691 can be confusing for new users.
    > We would recommend not selecting this option until you are fully comfortable with the Print Manager.

- **Cleanup** group:
  - **Automatically remove finished jobs** checkbox: Select to have jobs in the Print Manager automatically deleted, after a set period of time.
  - **Remove finished jobs after** setting: This option only becomes available if the **Automatically remove finished jobs** checkbox was selected.
    Select the time period (measured in either *Minutes*, *Hours* or *Days*) for which finished jobs are to be retained. When this retention period is completed the job (s) are deleted. The deletion both removes the job from the Printer Queue, as well as deleting the file on disk.
    The job will remain listed in the job history, but can no longer be viewed.

- **History** group:
  - **Limit History view at startup to** selection: Use this to limit the number of jobs that will be loaded in the Job History table. The Job History table becoming very large was found to have adverse impacts upon Print Manager performance. This option allows you to limit the Job History to more manageable levels.
    When limiting the number of jobs shown, it is the newer jobs that are retained.

- Select how the Job History table should be ordered. Either **Descending** or **Ascending**..

Language Setting

- **Display language**: Select a language from the drop-down list to be used as the language of the User Interface (after the software is restarted).

Log Settings

OL Connect logs the activities it undertakes whilst running. New Connect logs are created daily and are held for a period before they are automatically deleted.

The settings on this page determine the level of logging and how long the log files should be retained.

These log files can be an essential resource when diagnosing issues with OL Support. The logging settings are global to ***all*** Connect applications and the Logging Preferences can be adjusted from within any of the Designer, Print Manager and Server Configuration Preference dialogs.

The settings are:

- **Overall Logging Level** selection: Select the overall Connect Logging level. This controls how much logging Connect will do.
  By default the logging level is set at the midpoint of *Info*, but it can be set higher to include more logging (*All*, *Trace*, *Debug*), or lower to reduce the amount of logging (*Warning*, *Error*).
  The logging is hierarchical, with each Logging Level selection containing all of the Logging Levels below it. The default logging *Info* Level contains all *Error* and *Warning* entries plus informational log entries.

Caution: Higher logging settings will have an impact upon Connect production speeds, as well as leading to substantially larger log files.

We recommended leaving the logging level to *Info* and only using the higher levels of logging in conjunction with advice from OL support.

- **Rollover policy** selection: Chose whether to retain Connect log files for a certain number of days (**Daily logs**) or based upon some predetermined hard disk usage limitations (**Size-based logs**). The selection are as follows:

- **Daily Logs**: Use this setting to determine how many days Connect Logs are to be kept, before they are deleted.
    - **Number of days to retain logs**: This value only impacts upon historic (closed) logs.
      Chose between 1 and 99,999 days.
      The default value is set to 10 days for a new installation and 99,999 days on existing installations (to preserve backward compatibility).
- **Size-based logs**: Use this setting to restrict log file size, and to keep only a specified number of them.
  By combining the maximize individual log file size with the amount of log files to retain, this effectively allows a hard disk space usage limitation to be placed upon the logging process.
    - **Maximum size for log file**: This sets the maximum size a log file can reach before the logging system creates a new file.
    - **Number of files to keep**:This sets the maximum number of log files kept in the log folder. The default value is set to 50 for a new Connect installation and 99, 999 for an existing installation (to preserve backward compatibility).

- **Logging pattern** edit box: This edit box determines the formatting of the individual log entries.
  By default it is set to a date/time value which allows for simple searching within log files.

  Caution: We recommend leaving the Logging pattern to the default value.

  If you do need to change the Logging pattern, please see the Pattern Formatting guide (https://logback.qos.ch/manual/layouts.html#conversionWord) for help in doing so.

- **Log message preview** display: This displays a real time example of the format and content of individual log file entries, based upon the **Logging pattern** setting.

Advanced Log Settings

The Advanced settings over-ride the Overall logging settings, and provide a greater level of logging granularity.

Caution: Higher logging settings will have an impact upon Connect production speeds, as well as leading to substantially larger log files.

The Advanced Log Settings should only be set in conjunction with advice from OL support, to ensure that only the most relevant settings are set to the higher logging levels.

This Preferences page allows you to add ( ) or remove ( ) individual Connect Packages, or change their logging settings ( ).

SNMP Connection Settings

Options for printers that support Simple Network Management Protocol (SNMP) connection. Not all IPDS printers do, but most PostScript and PCL printers will. This protocol allows the printers to communicate their status to the Print Manager.

All the SNMP settings relate to SNMP connection and re-connection attempts. They are:

- **Connection attempts** entry: Defines the number of times the Print Manager will attempt to connect to a printer before abandoning the attempt. This setting applies only when the Print Manager is started.
- **Polling interval (secs)** entry: Defines the polling interval in seconds.
  This determines how often the Print Manager attempts to communicate with the printer, and applies to all communications with the printer. It applies both to the initial **Connect attempts** (setting the interval between those connection attempts) as well as how often the printer is polled after a connection has been established.
  Each time the printer is polled the printer status is updated in the "Printer Status pane" on page 1712.
- **SNMP keep retrying after timeout** checkbox. Tick this if you would like to attempt to reconnect to the printer if a connection fails (or times out). Once ticked, you can set the retry interval amount.
  - **Retry interval on Timeout (secs)** entry: Enter the length of time (in seconds) between connection retries. This interval period applies only to attempts at reconnecting to printers.

OL Connect Release Notes

This chapter contains the current OL Connect 2025.1.x Release Notes, as well as the Release Notes from previous versions.

- "OL Connect Release Notes 2025.1.1" below

Previous releases:

- OL Connect Release Notes 2024.2
- "OL Connect Release Notes 2024.1" on page 1743
- "OL Connect Release Notes 2023.2" on page 1759
- "OL Connect Release Notes 2023.1" on page 1775
- "OL Connect Release Notes 2022.2" on page 1794
- "OL Connect Release Notes 2022.1" on page 1808
- "OL Connect Release Notes 2021.2" on page 1823
- "OL Connect Release Notes 2021.1" on page 1833
- "OL ConnectRelease Notes 2020.2" on page 1843
- "OL Connect Release Notes 2020.1" on page 1855
- "OL Connect Release Notes 2019.2" on page 1867
- "OL Connect Release Notes 2019.1" on page 1881
- "OL Connect Release Notes 2018.2" on page 1893
- "OL Connect Release Notes 2018.1" on page 1913
- "OL Connect Release Notes 1.8" on page 1931
- "OL Connect Release Notes 1.7.1" on page 1950
- "OL Connect Release Notes 1.6.1" on page 1971
- "OL Connect Release Notes 1.5" on page 1984
- "OL Connect Release Notes 1.4" on page 1995

OL Connect Release Notes 2025.1.1

License Update Required for Upgrade to OL Connect 2025.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update OL Connect.

Every OL Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of OL Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This licensing model requires anyone upgrading from OL Connect versions earlier than 2019.1 to first update their OL Connect License. A dialog box appears as part of the OL Connect upgrade process requesting that you do this.

It is recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version.

For further details on how to upgrade the OL Update Manager to the latest version and update your OL Connect License see the Update Manager 1.9 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing OL Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2025.1 and Workflow 2025.1, as well as some important installation information.

Installing OL Connect 2025.1 and Workflow 2025.1

- OL Connect is released as a 64 bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online helps.
  OL Connect: Installation and Activation.
  OL Connect Workflow: Workflow Installation.
- Note that both OL Connect and Workflow come with a 30 day trial license by default.
- It is recommended that you install and use OL Connect Designer and Server under the same user account.

OL Connect and Workflow installers require Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

OL Connect templates used in Workflow

For improved performance we recommend re-saving any OL Connect templates used in Workflow to the current version of OL Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2025.1.1 Improvements

During the OL Connect 2025.1.0 partner preview the following issues were discovered, and fixed for the 2025.1.1 release:

- An issue was discovered where if PDF security options were used in the Output Preset, some links within the output PDF would not work. (94542)
- An issue was discovered with PlanetPress Suite documents containing PNG images with transparency. These would show some slight differences in output. (95096)

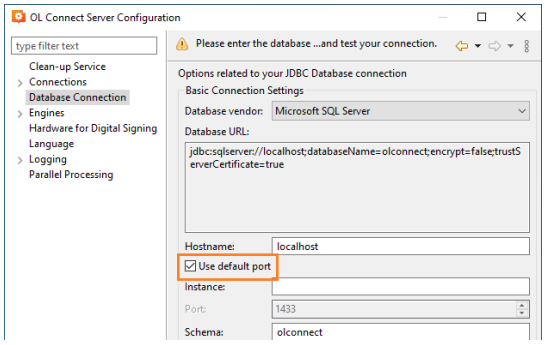
OL Connect 2025.1 Improvements

Windows Server 2025

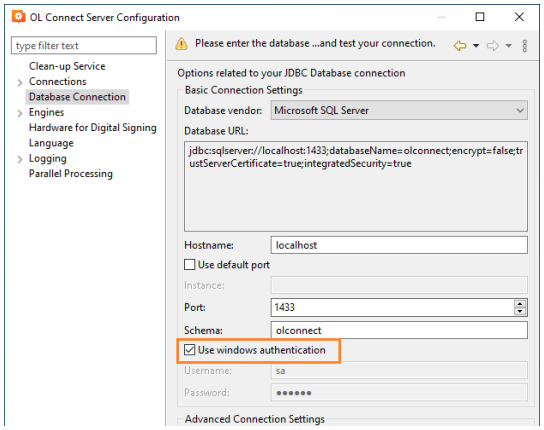
OL Connect and OL Connect Workflow are now officially supported on Windows Server 2025. (95106)

OL Connect general improvements

- Added ability to force using the default port in the database connection Preferences. This allows you to connect by **Instance** name when using a **Microsoft SQL Server** as your back-end database. (92236)



- Support added for turning **Windows authentication** on and off for **Microsoft SQL Server** back-end database connections. (92247)



- Improved stability for better detection of licensing problems. (92497)
- Fixed an issue where in certain scenarios no data would initially be available in the Designer after opening a data mapping config, unless you also refreshed (synchronized) the data model. (92652)

- Installer now does an extra validation to prevent the Connect server from being configured to run as one of the built-in Windows service accounts, which is unsupported by Connect. (92923)
- Fixed an issue with the custom database preferences set by the installer not using the proper delimiter. (93730)
- Improved Context Sensitive Help for a number of dialogs. (93847\93859)
- Several improvements made to interface localisation. (93996)
- The **Logging Preference** rollover policy was not being applied to the licence logs. This has now been fixed. (94012)
- Resolved issue with Preferences being incorrectly updated or even deleted when upgrading Connect. (94242)

Architectural Software Updates

- Removed Microsoft VBScript dependencies from OL Connect. (93724)
- Replaced older Eclipse j2v8 with Javet. (93864)

Security improvements

- Third party libraries updated to more recent versions. (94136/94221)

OL Connect REST API improvements

- If a `jobOutputFolder` parameter is specified for an **All-In-One** print operation, the pre-output test print will now use the value of the parameter specified instead of the default value. (94353)
- The Rasterize File REST API endpoint now supports the conversion of XPS files. (94597)
- Document set properties are now included in the **JSON Job Statistics** structure returned from the Job Statistics REST API endpoint, when a level mode of `documentSetOnly` or `documentSetAll` is specified. (94716)
- Creating too many REST API calls could cause a Stack Overflow, which would require restarting the Server to resolve. This has been fixed. (94723)
- Improved security by removing hard coded default credentials from some PowerShell commands used in the REST API Cookbook. (94881)
- Rate limiting has been added to REST API authentication to reduce the risk of malicious parties attempting a brute force attack. (94883)

OL Connect 2025.1 Designer Improvements

Improved support for editing non-Connect resources

Double-clicking a resource file in the resources panel for which the Designer does not have a built-in editor now pops up a dialog, which allows you to open that resource file in the default application associated with that file type, directly from within Designer. (90440)

For example, double-clicking an image resource in the Resources. If the Designer doesn't have a built-in viewer for the file type, a prompt will appear, allowing you to open it in the default application. This allows you to not only view the image but also make edits that are saved directly to the file in the template.

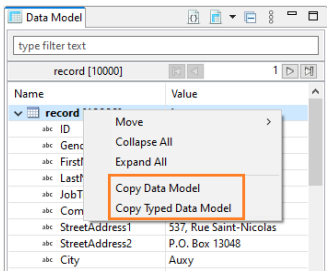
Section Cloning Wizard improvements

- Cloned sections no longer require unique names. (93973)
- The Section Cloning Wizard now lets you set section clone names based on a data field, such as an insurance policy name. This makes it easier during the design phase to quickly find and view a specific section in Preview mode, helping you validate the content presentation or check the results of certain business rules. (94772)
- The Section Cloning Wizard now inserts expressions that work with the detail record context, reducing the need to rewrite or remember the correct syntax.

  Note: Only supported in the wizard, not in raw scripts.

Copy Data Model

Options added to the context menu of the Data Model view allowing you to copy the model of the current template/data mapping configuration.

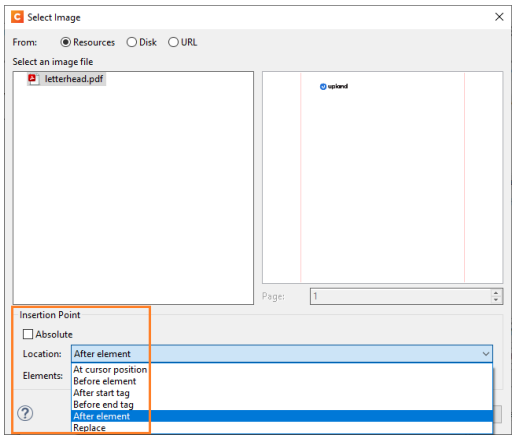


These copy a JSON representation of the data model to the clipboard. This makes it easier to generate sample data using AI tools like ChatGPT and Microsoft Copilot. Simply copy the model, then paste it into the AI and request one or more sample records. (94114)

Learn more about streamlining template design with AI:
https://olresourcecenter.uplandsoftware.com/blog/streamlining-template-design-with-ai/

Insert image through insertion point

The Image Insert dialog now includes an **Insertion Point** option, which inserts the new image at a specific position. (92004)



In previous versions, images were always added as inline elements, requiring manual steps to move them to the correct location or place them in an absolute position. This now vastly simplifies that process.

General Designer improvements

- The ability to create additional copies of records in the DataMapper is a powerful feature that was introduced in a previous version. However, until now, the results could only be viewed by generating output. We have now made the copies visible and browse-able within the Data Model so you can preview them directly in the Designer.
- Clicking on an absolute positioned image element will now trigger the box selection mode directly. (90602)
- When a Data Model field is drag and dropped outside the page margins an absolute positioned box is created. If you resized that box to be smaller than the contents within, the text would overflow outside the box. This is no longer the case. The box now behaves similar to a regular absolute positioned box that can be inserted from the toolbar. (91170)
- Fixed an error in the Data Model for the last record, when the number of records to be displayed is limited. (91302)
- Fixed issue with buttons disappearing after **Expand All** selected in Data Model. (91576)
- Remote content is no longer cached while editing a template in the Designer. (91620)
- The JSON sample data has been simplified and improved. (91788)

- Support added for "Alt+arrow up" and "Alt+arrow down" keyboard shortcuts to both script source and JSON sample data editors. (92463)
- Added support for zooming in and zooming out using the CTRL + '+' and CTRL + '-' keys in the script source and JSON sample data editors. (92479)
- Fixed an issue with the display of filtered results in the Data Model view. (92965)
- If the main editor is read-only we have made sure you can still scroll in the Attributes panel and view and copy-paste values. (93118)
- You can now browse through copied data in the Data Model view. (93204)
- You can now define the encoding used in external files imported in to the DataMapper or Template. (93372)
- When importing from Word we now substitute the font “Aptos” (a proprietary cloud font that cannot be used by a browser) with the more general “Calibri”. (93417)
- Fixed a cosmetic issue where the Data Model view would keep showing a “Copies” label even after closing the DataMapper config editor. (93652)
- Prevented excess error logging by the barcode library. These would appear in the Designer messages panel and could cause confusion. (93685)
- Inserting a barcode on a Master page will now only insert a script if the active section does not have Handlebars enabled. If the section has Handlebars enabled a Handlebars expression will be used in the barcode box. (93862)
- Any changes made in the Source tab to the inner HTML of an (empty) article for shared content are now assumed to be unintentional and will be ignored. (93978)
- Fixed an error that was encountered when switching a cloned section between Preview and Source modes. (93994)
- When dragging a data field to a cloned section that uses a detail table as the context for Handlebars, the expression is adjusted accordingly. (94001)
- When you create a package to be sent to Workflow or saved on disk and you try to add a resource with a name that is already in the package you will now get a dialog asking if you want to overwrite the file in the package with the new one. Prior to this change you could have two files with the same name but when the package file was saved an error would occur. (94095)
- Fixed some issues with nested tables that include Handlebars expressions. (94361/94362)

- Fixed an issue where master page objects where positioned incorrectly on the section page when the top and left bleed were set to different values. (94398)
- Fixed an issue with *Dynamic Image Scripts* when the *Folder on disk* location option was used with an empty path. Previously this caused the base path to use the path of the Designer application. Now it will use an empty path as the base path.
  Additionally a validator was added to that *Path* field to disallow relative paths and accept only absolute paths.
  This validator was also added to the *Dynamic Background Script*. (94549)
- Improved text font size changes implemented through Format dialogs. (94550)
- When the *Duplex* checkbox in the **Sheet configuration dialog** gets checked we now set the *Allow content* on fields to *All sides*. If it gets unchecked, we set it to *Front Only*. (94551)
- Applying text alignment within a Positioned box no longer wraps content in a new inline element. (94554)
- Fixed an issue with the script debugger that may occur if a run is aborted and Handlebars are not used. (94566)
- Improved support for Handlebars expressions in src or href attributes. We URL-encode the expression result by default, but this can now be disabled using a triple-stash. (94595)
- When selecting an Absolutely positioned table through the 'breadcrumbs' section and then choosing the *Table...* option, none of the fields within the **Table Properties** dialog were set. This has now been fixed. (94645)
- Improved the importation of new tabular data into Templates already linked to a data file. (94720)
- Fixed an error that may occur when applying styles like bold and italic to text in a positioned box. (94811)
- Fixed an issue introduced in 2024.2 that prevented *page-break-before* or *page-break-after* styles from having any effect when applied to an element inside a table. (95132)
- Fixed an issue where references to empty Handlebars partials would produce an error instead of the reference being replaced with an empty string. (95262)

OL Connect 2025.1 DataMapper Improvements

Improved handling of control characters in DataMapper configuration fields

- Improved stability of configuration scripts containing special characters. (90417)
- Prompt when saving configuration files with invalid characters. (93412)

General DataMapper Improvements

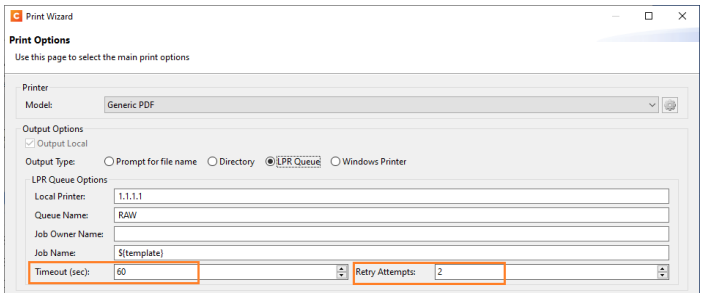
- Added support for **Unicode surrogate pairs** to ensure proper extraction of extended Unicode characters in text data file processing. (57312)
- Improved **JavaDoc** handling in the post-processor script engine of DataMapper. (90673)
- In **XML data mapping** configurations, extraction fields based on Location now have their XPath property validated to make sure they are not empty. (91393)
- Improved action to rename multiple fields in DataMapper extraction step. (91525)
- Improvement of `jsonpath` default selection when adding repeat step to JSON data. (91585)
- We now check whether font files referenced by user Fontmap files actually exist. (91802)
- Improved use of the selected field in the appropriate view of the Extraction step. When selecting any field from the Field List dropdown, the Data Model now highlights the corresponding field and scrolls it into view if it isn't already. (92262)
- The actions **skip to next record** and **stop datamapping** are now available in the DataMapper script engine as functions. This reduces the need for additional Condition and/or Action steps. (92289)
- An error will now be shown when the **Sheet field is empty** in an Excel input data settings. (92388)
- The lowest version that the DataMapper configuration can now be saved as is version 1.8. (92443)
- Fixed an issue where printing using a JSON based DataMapper configuration with a data file that used an encoding other than UTF-8 would generate an error. (92622)
- Improved property management for `automation.properties.OriginalFilename`, which is based on the filename stored in the OL Connect database. (93092)
- Improved the efficiency of text searching and implemented infinite loop detection in the **PDF skip to the next occurrence** step. (93146)

- Database ID allocation has been improved to reduce database traffic and also avoid potential ID allocation conflicts. (93459)
- Fixed the exception message when loading empty XLSX files from the DataMapper wizard with the '*First row contains field names*' option enabled. (93774)
- Fixed mistaken error message that appeared after setting ODBC boundaries in the DataMapper. (93827)
- An issue was discovered when using certain AFP files through the DataMapper, whereby Type 1 CID fonts were not being extracted properly. This has been fixed in this release. (94701)
- Improved performance for tabular input when using an "on script" trigger for boundaries. (94885)

OL Connect 2025.1 Output Improvements

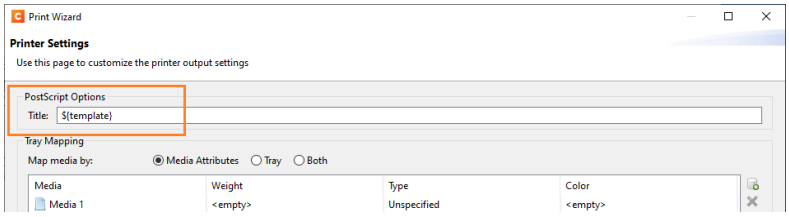
LPR print Improvements

- Multiple simultaneous LPR connections could cause failures, and then those failed jobs could mistakenly removed.
  We now move files to a **printservice** folder and add an LPR job to an internal task queue. This reduces the possibility of failures due to multiple simultaneous connections. And if the LPR fails the job will now be moved to the **printservice/error** folder. (86245)
- LPR Queue *Timeout* and *Retry Attempts* are now configurable in the Print Options page. (94461)



Controlling the Title value in PostScript output

Added capability to control the %%Title value in PostScript output.



The %%Title field is often used by printers to display as the job name in the printer controller. With this change OL Connect users no longer need to rely on automation tools (such as Connect Workflow) to modify the field. (94073)

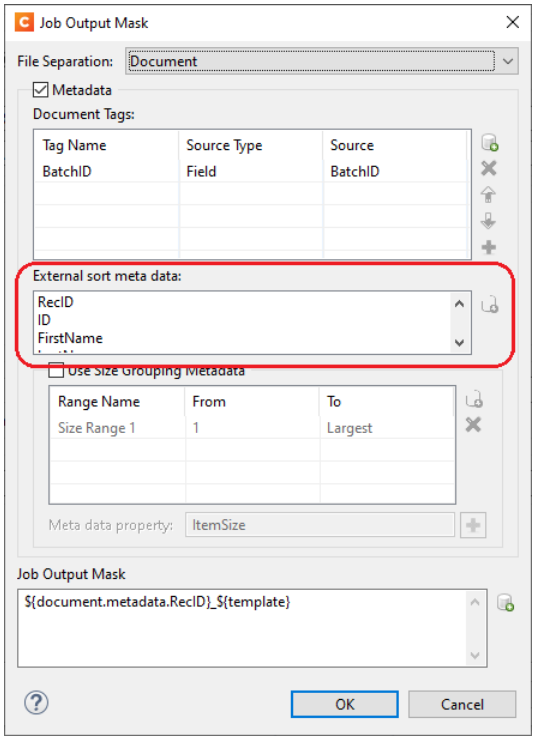
Prompt for Printer Properties in all Connect versions

For OL Connect Desktop users, the "Prompt for Printer Properties" feature allows users to submit print jobs through the printer driver and modify finishing options (such as orientation, simple or duplex, color, & etc.) within Connect Designer.

This functionality has now been extended to OL Connect Professional and Enterprise editions, enabling users to handle ad-hoc jobs more efficiently directly within Connect Designer. (94294)

Print Wizard\Preset enhancements

- External sort metadata is now catered for in the Job Output Mask in the Print Options page of the Output Preset and Print Wizards along with the job output mask dialog. (92784)



- Increased the number of fields that can be supported in job creation grouping. (93589)
- Fixed an issue that was preventing dynamic PPD based output presets from being loaded into the Output and Print Wizards. (94250)
- Fixed display issue when switching between PDF and non-PDF printer models in the Print Options Page of the Output and Print Wizards while the output type is set to windows printer. (94670)

General Output Improvements

- Fixed an issue with loading images when the user profile location contains characters outside the US-ASCII range. (93453)

- Fixed an issue with batched pagination that could cause the output to contain blank content if the output for a single record consisted of more than 15 pages; one of the top-level elements spans multiple pages; and that element happens to intersect a page range boundary for batched pagination (every 15 pages by default). (94516)
- Improved behavior when the user running OL Connect has Unicode characters within their name. (94657)

Email Output Improvements

- Fixed an issue where the "*Do not inline styles*" option would still inline CSS style rules on elements. (94637)

  > **Note:** This might affect existing jobs. Particularly if you were inadvertently using the "Do not inline styles" option, but actually wanted the "Apply CSS properties to elements" behavior.

Print Output Improvements

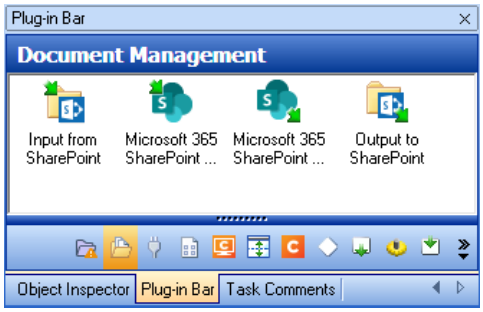
- Fixed an issue that could cause missing pages in output if a single record produces more than 15 pages of content and a post pagination script calls `merge.section.paginate`. (93714)
- Prevent entry of characters in the Job, Output and Print Wizards that are not valid according to the XML 1.0 specification. (94000)
- Fixed an issue where (in certain edge cases) repagination from a post pagination script could produce overlapping master pages. (94025)
- Additional Content now supports text patterns with hexadecimal and Unicode escaped characters. For example: ${"\x41"} (94465)
- Fixed an issue that could occur in some edge cases when "*Omit Master Page Back in case of an empty back page*" was turned on. (94478)
- Fixed masking of high byte in character codes when a TrueType (3,0) cmap table is being used. (94630)
- An issue was identified with gray color in PCL output, where specific gray values were inverted. Light gray appeared as dark gray and vice versa. This problem has now been resolved. (94658)
- Fixed a *StackOverflowException* which was being encountered during the creation of AFP output when processing a PDF input file that included a significant number of form elements (more than 32,000). (94949)

- Addressed an issue where filestore uploads from output engines running on a server extension would sporadically fail when there were large numbers of output files being uploaded. (95225)

Workflow 2025.1 Improvements

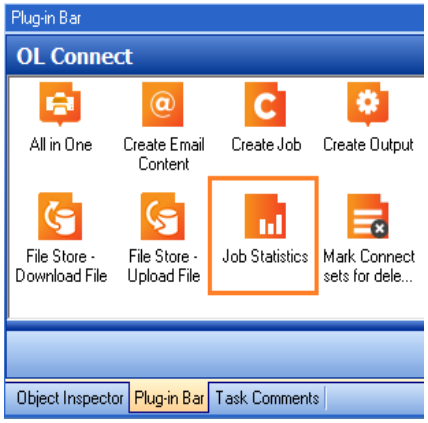
Microsoft Office 365 support

Added new Microsoft Office 365 Sharepoint **Input** and **Output** plugins, under the **Document Management** category.
These new OL Connect Workflow tasks allow you to upload and download documents from Microsoft Office 365 SharePoint environments. They simplify integration by connecting directly to Microsoft Office 365 SharePoint through the user interface, eliminating the need for custom scripts. (52936)



New Job Statistics task

A new task in OL Connect Workflow has been added to retrieve statistics for print jobs created through OL Connect Server. The data returned includes the number of documents, sheets, and pages used, as well as the media types applied. This feature is valuable for accounting purposes and helps you track costs for better budget management in a printing environment. (91376)



Additionally, the information includes custom properties assigned to documents and document sets, which can be used to enrich documents with relevant meta data when uploading them to a document management system for archiving.

Security and library improvements

- Third party libraries (including OpenSSL and NodeJS libraries) updated to more recent versions. (92720)
- We now use a 64-bit version of the AlambicEdit PDF library, allowing operations on larger files to be performed more efficiently. (92026)
- The **SOAP Client** plugin now supports up to TLS 1.3 security protocol. (93189)

NodeJS Input plugin improvements

Improved error behavior. (92055/92937)

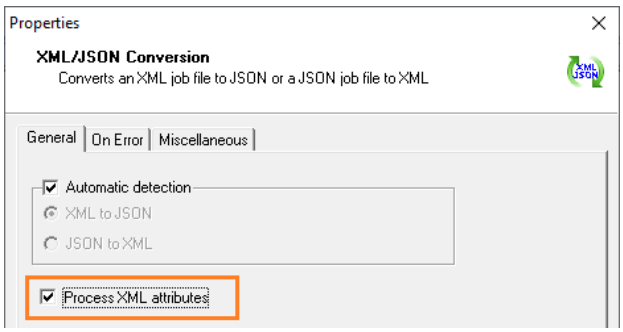
Improvements include:

- Improved logging on error.
- Improved error response feedback from NodeJS Input plugin.
- The log message indicating the SOAP server's wsdl address now contains the actual port being used.
- The module name in the Route mapping messages is now set to Workflow NodeJS Server instead of Workflow HTTP Server.

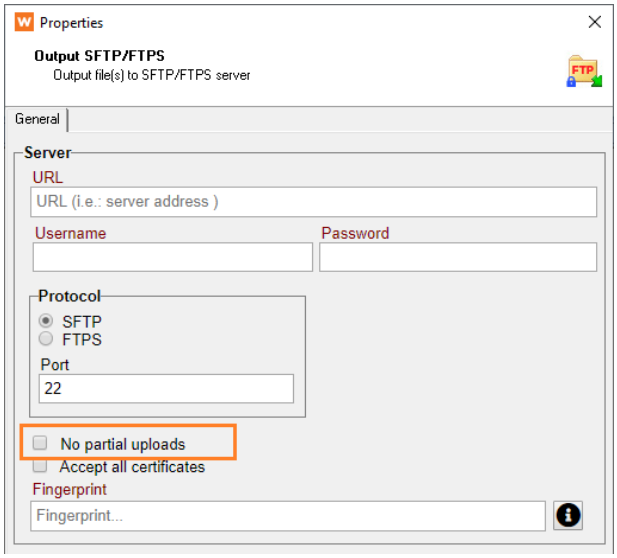
General Workflow Improvements

- Fixed out-of-memory errors that occurred when creating PDFs via the **Create PDF** plugin. (72549/82496)
- Fixed perspective reset action during Workflow start for dialogs: **Open Script**, and **Open XSLT**. (77932)

- Add support for *XML attributes* in the **XML->JSON** plugin converter, ensuring their values are stored in the job data for easy access in tasks like RunScript. (89733)



- A problem was fixed where producing an archive to a path which contained non-ASCII characters would cause conversion to PDF to crash. (90638)
- Microsoft XSLT processor was added as one of the **Open XSLT** plugins processor options. This delivers a more efficient transformation engine for converting XML content into other structures and document formats.
  The XSLT 1.0 and XSLT 2.0 options were renamed to Altova (XSLT 1.0) and Altova (XSLT 2.0), respectively. (90692)
- An option was added to the **SFTP/FTPS output** plugin to allow disabling making partial transfers when uploading (partial transfers are enabled by default). (92019)



- When dragging and dropping a data mapping configuration to the Workflow process and deciding to set the sample data file from the data mapping configuration as the current Workflow sample, the file that was most recently modified is now used. (92087)

- “Plugin help” option is now available when right clicking on a plugin in a Workflow process. It will open the system’s browser and load the context help for the highlighted plugin. (92162)
- Logging for all **Connect plugin** operations was improved to better facilitate debugging of processes. (92268)
- An extra timestamp in the Workflow logs was removed as it just duplicated another. (92269)
- Improved error handling for **Connect plugins**. (92352)
- Resolved an error that occurred when closing the Properties window of the Microsoft 365 Email Output plugin in the **French version** of the Workflow. (92449)
- Increased capacity of identifiers list used in job creation API calls. (92482)
- Fixed an issue with the **FTP Input** where files whose names contained characters which were non-ASCII but still within the current locale (example: "ö" in a Latin locale) would fail to be captured. (93011)
- Improved download speeds for the **SFTP/FTPS plugins**. (93130)
- Runtime parameters in Workflow Connect plugins now keep the proper parameter listing order from their original OL Connect Template. (93742)
- Issues with Workflow installation when Workflow Services were not running with LocalSystem have been resolved. (93883)
- Fixed %O variable in Input **SFTP/FTPS plugins**. (94143)
- Fixed **Secure Email Output** error for p7m email attachments. (94147)
- The **Job report plugin** was moved to the legacy category. (94403)
- A problem was fixed where the metadata viewer would not open on a system where the language was set to Serbian (Latin). (94564)
- Removed erroneous empty line from the Workflow includes.json file. (94572)
- When the **Optimize memory usage over performance when processing PDF files** option is turned on, the AlambicEdit library used is now 64-bits instead of 32-bits. (94581)
- **XML/JSON Conversion** plugin now supports node names starting with a numeric character. (94612)
- Memory leaks were fixed in the **Print using a Windows Driver** plugin. (94620)

- A problem was fixed where processing multiple PDF/VT files simultaneously using the **Digital Imaging** or the **OL Connect Image service** would result in some files randomly failing. (94689)
- A problem was fixed where using one of the special XML characters (&, <, >, ") in the configuration of the **Logger**, **Mathematical Operations** or **PReS Print Controls** plugins caused those characters to be replaced by their XML entity equivalent (&, <, > and ", respectively) when running or saving the configuration. (94823)
- Microsoft Visual C 2013 redistributables were removed from the Workflow installation in 2024.2. This caused errors with some Worklow plugins, so they have been reinstated. (95097)
- Memory leak fixed when Workflow was loading DataMapper configurations upon starting up the tool or refreshing the OL Connect files from the treeview. (95306)

Known Issues

The Known Issues can be found here.

Previous Releases

OL Connect Release Notes 2024.1

License Update Required for Upgrade to OL Connect 2024.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update OL Connect.

Every OL Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of OL Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This licensing model requires anyone upgrading from OL Connect versions earlier than 2019.1 to first update their OL Connect License. A dialog box appears as part of the OL Connect upgrade process requesting that you do this.

It is recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version.

For further details on how to upgrade the OL Update Manager to the latest version and update your OL Connect License see the Update Manager 1.8 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing OL Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2024.1 and Workflow 2024.1, as well as some important installation information.

Installing OL Connect 2024.1 and Workflow 2024.1

- OL Connect is released as a 64 bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online helps.
  OL Connect: Installation and Activation.
  OL Connect Workflow: Workflow Installation.
- Note that both OL Connect and Workflow come with a 30 day trial license by default.
- It is recommended that you install and use OL Connect Designer and Server under the same user account.

OL Connect and Workflow installers require Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

OL Connect templates used in Workflow

For improved performance we recommend re-saving any OL Connect templates used in Workflow to the current version of OL Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2024.1.1

During the OL Connect 2024.1.0 partner preview a regression issue was discovered with PDF text extraction using the DataMapper. This was fixed in the 2024.1.1 release. (92889)

Architectural Software Updates

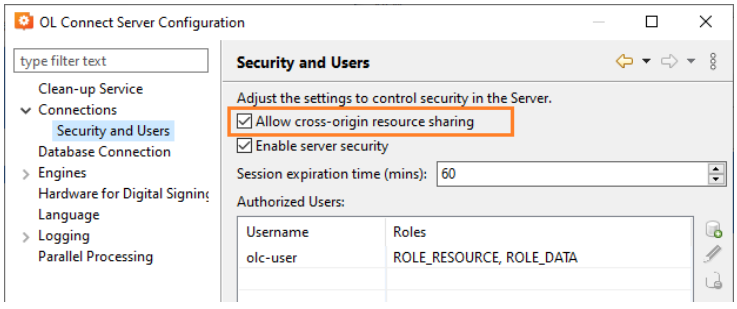
- **Java**: Updated the Java platform which OL Connect runs on to Adoptium Temurin version 17.0.10. (92408)
- **MariaDB**: Updated the database that is packaged with OL Connect to 10.11.5. (89879)

Google Cloud Platform

Google Cloud Platform is now officially supported as a cloud environment to deploy OL Connect, with AWS and Azure. (92104)

Security improvements and library updates

- The setup no longer installs Microsoft Visual C++ Runtime 10.0 (x64) as part of the prerequisites. (90229)
- Added Server configuration option to disable cross-origin resource sharing. (92861)

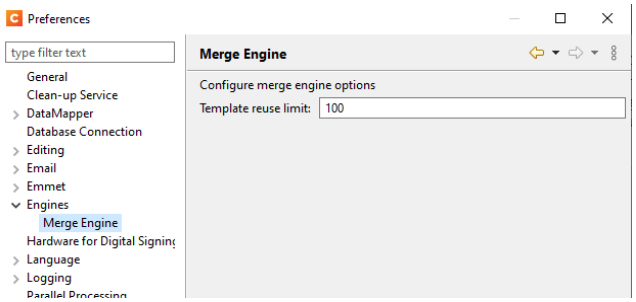


Installer Improvements

- A **Use Windows Authentication** option was added to the installer when configuring a custom Microsoft SQL Server database. (81421)
- If an error occurs when installing the backend MariaDB database and the database service is found not to be running, improved error details are now recorded in the installation log. (90627)

OL Connect backend improvements

- Added **Merge Engine** *Template reuse limit* to the Preferences. (87403)



- Fixed handling of dates under selected Microsoft SQL Server login languages, when using partitioning for the clean up service. (90679)
- Updated validation of Microsoft SQL Server instance names in the database preference pages to match Microsoft requirements. (90683)
- Fixed issue whereby the magic number was not displayed in the **Activation dialog** due to a computer firmware issue. (90914)

REST API improvements

- **Job Creation without Job Preset**
  Added support for running default job creation processes on Content Sets via new REST API endpoint `/rest/serverengine/workflow/jobcreation/default/{contentSetId}`, without requiring a Job Creation Preset. (89926)
- **Print Job Statistics**
  A new REST API endpoint has been added to OL Connect Server to retrieve statistics of print jobs:
  `/rest/serverengine/statistics/jobset/{jobSetId` (91372)

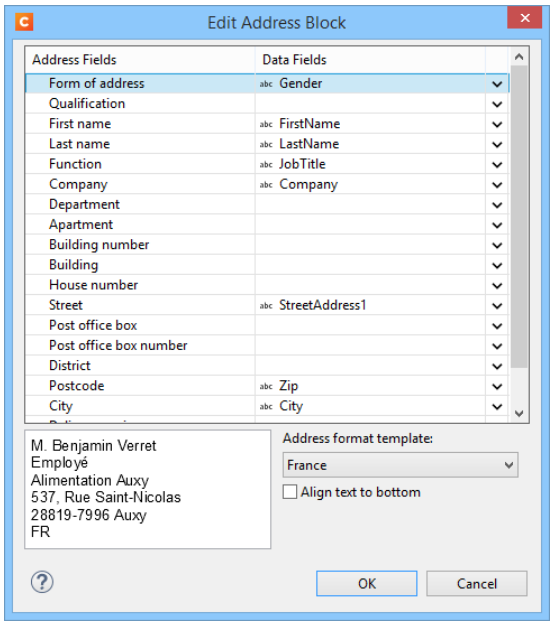
The information contains the number of documents, sheets and pages used, along with the applied media types. This is useful for accounting purposes and helps with calculating costs to manage budgets in a printing environment.

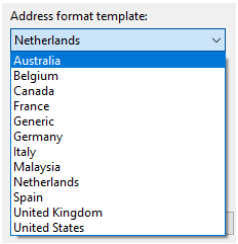
The same endpoint also returns document tags for each document (such as dataRecordId and contentItemId), which can be used to populate metadata for document management and archiving solutions. These tags are set in the Job Preset configuration.

Address Block

A new address block feature has been added to OL Connect Designer to effortlessly insert addresses formatted to the specific guidelines of the selected country. (8795)

You can link data to the elements that compose the address details via the Address Block dialog. The feature automatically collapses empty lines, and provides an option to align the address to the bottom of the address block object.





It automatically sorts the address lines as per the specified country's address specification, and will even add extra spaces between address elements, if required.

At the moment address block templates are provided for the following countries: United States, Canada, United Kingdom, Germany, France, Italy, Spain, Netherlands, Belgium, Australia, and Malaysia. Other regions and postal service specific formats will be added in future versions.

For specific countries the feature provides options to render postal barcodes, for example the Dutch KIX barcode.

The feature also allows for custom address block templates. This allows you to create your own custom address block templates, for accommodating missing regions or meeting specific postal service criteria.

Major Handlebar Improvements

- **"Else if" construct**

  The Handlebars implementation in OL Connect supports the fundamental *"if else"* statement, allowing you to conditionally display content within Handlebars snippets. This construct is now expanded with the *"else if"* statement, enabling even more sophisticated comparisons. (91943)
  For example:

```
{{#if eq policy "HI"}}
    Health Insurance
{{else if eq policy "CI"}}
    Car Insurance
{{else}}
    General Insurance
{{/if}}
```

- **Concat() helper**

  Added a 'concat' helper for Handlebars, which concatenates two or more arguments. (91843)
  It is useful when you want to combine multiple strings for example to dynamically load the contents of a snippet using the partial notation:

```
{{+ (concat 'product_' productName)}}
{{+ (concat productName '_' lang)}}
```

  This loads the contents of the *"product_toothbrush"* or *"toothbrush_EN"* Handlebars snippet respectively.

- **New data variables**

  New data variables (prefixed with "@"-sign) are introduced to access **runtime parameters** and **template properties**. (91997/92039)

```
{{@parameters.workspaceFolder}}
{{@properties.application}}
{{@properties.custom.MyProp1}
```

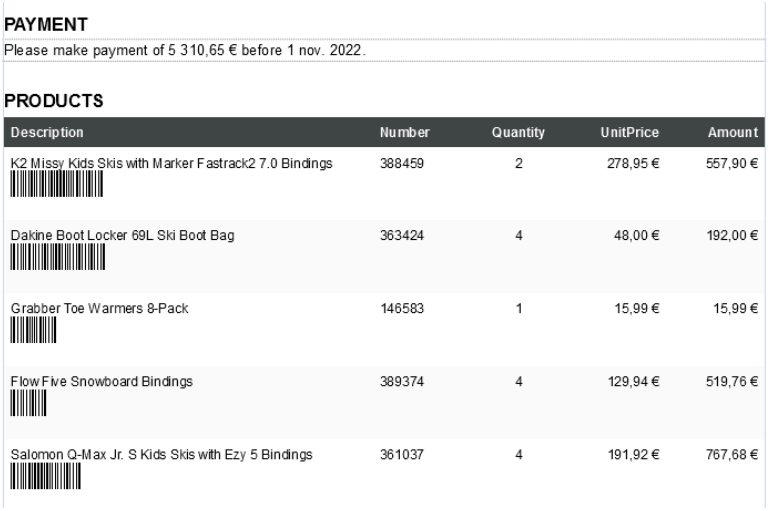
- Added support for *hash* arguments and *Partial Contexts*. (91956)
- Improved behavior for *whitespace control*. (92173)
  See Expressions | Whitespace control for more info.
- *Raw blocks* (those that start with four curly braces) are now ignored. (92343)

Translation improvements

When **po** files are missing the Plural-Forms header, OL Connect now use a default Plural-Forms definition for the language, based on Plural Forms - Localization Guide. (90209)

Dynamic Table improvements

Added support for barcodes in dynamic tables, enabling the creation of documents such as warehouse pick lists with scannable barcodes for each line item. (91196)



PAYMENT

Please make payment of 5 310,65 € before 1 nov. 2022.

PRODUCTS

Description	Number	Quantity	UnitPrice	Amount
K2 Missy Kids Skis with Marker Fastrack2 7.0 Bindings	388459	2	278,95 €	557,90 €
Dakine Boot Locker 69L Ski Boot Bag	363424	4	48,00 €	192,00 €
Grabber Toe Warmers 8-Pack	146583	1	15,99 €	15,99 €
Flow Five Snowboard Bindings	389374	4	129,94 €	519,76 €
Salomon Q-Max Jr. S Kids Skis with Ezy 5 Bindings	361037	4	191,92 €	767,68 €

General Designer improvements

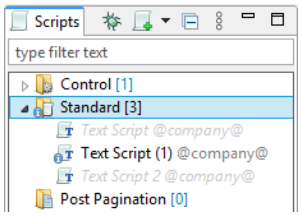
- Fixed a bug where resources added to the template (for example images) were not visible in the resources panel. (89309)
- Fixed an issue with **fr-CA** locale accounting currency formatting incorrectly. (90227)
- Fixed an issue with switching from Preview mode to Design mode, if Translations are used in combination with '*Evaluate Handlebars expressions*'.
  In order to fix this Preview mode is now read-only if '*Evaluate Handlebars expressions*' is turned on. (90420)
- Designer now makes sure to select an entire Handlebars expression, even it's only partially selected, when applying a style. For example setting a text selection to bold. (90597)

- Fixed an issue that could result in inappropriate margins after calling `merge.section.paginate` from a post pagination script. (90932)
- When inserting an image with a name that already exists in the template, you can now overwrite the existing image with the new one. (90976)
- Updated the default styling for `<hr>` elements to avoid rasterization in AFP output. (91004)
- Added **date formatting options for strings** in the Text Scripts and Text Helper wizards. These can be used to present the string as a date, but the date must be in ISO or UTC format for this to work. (91061)
- The proper value for `record.index` is now shown in Preview mode when using JSON sample data. (91108)
- Fixed display of icons in the **Package File** dialog when the windows display scaling is not set to 100%. (91226)
- In the **Word based print template** wizard the margin values are always in print units (the preferred unit) and now all the values are converted to 2 decimals. (91309)
- Fixed errors when renaming files and folders in the **Resources** panel. (91318)
- Fixed several issues related to forced breaks done through `page-break-after` and `page-break-before`. (91365)
- Downgrading OL Connect templates to 2022.2 (or earlier) now migrates dynamic tables correctly. (91476)
- A "*selector produced no results*" tooltip is no longer shown for a script if its selector matches one on one of the applied master pages. (91478)
- The *Set Type* menu item in the context menu for the Data Model view is now enabled as long as no data mapping configuration editor is open. (91513)
- **Welcome screen** improvement. The Designer is no longer unresponsive for 20 seconds on startup when working offline. (91517)
- Pressing an arrow key in an HTML source editor no longer triggers code completion. (91591)
- Improved start up time of OL Connect Designer when setting up database connection. (91857)
- The **Resources** panel now highlights the active resource. (92007)
  The text bubbles at the top of each page in the main editor now have a distinct look for sections and master pages.

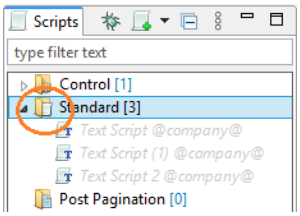
- The default scale setting for new inserted **QR Code** and **Datamatrix** barcodes has been changed to *Proportional*. (92037)
- Fixed an issue that prevented moving existing absolute positioned boxes on a master page of templates created in version 2023.1 or older. (92044)
- Improved version labels in **File > Save a Copy...** dialog. (92183)
- Fixed issue with renaming open snippets when Designer is first started. (92404)
- An error is now raised if an invalid date is passed to one of the date format functions. Previously this would fall back to January 1, 1970. (92607)
- Fixed an issue that could cause a dynamic table to be pushed to the next page unnecessarily in Preview mode. (92671)
- Improved the output of the Preview HTML, particularly for print sections. (92714)

General Scripting/Handlebar Improvements

- Handlebars expressions used in hyperlinks now work correctly in output. (90757)
- Added a visual indication as to whether the script folder contents are all disabled or not. (90759)

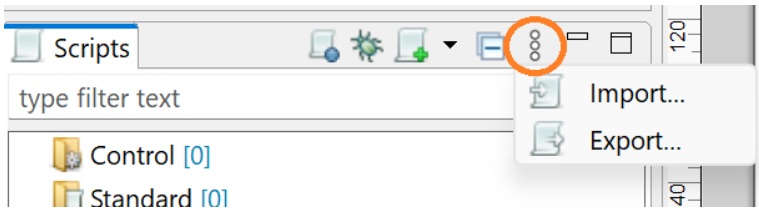


Some Enabled



All Disabled

- Dynamic media settings can be modified through one or more Control Scripts. For example: `var section = merge.template.contexts.PRINT.sections["Section 1"] section.sheetConfig.positions.all.media = "Media 1"` (90854)
- Moved the import and export scripts options to the ellipsis menu of the Scripts View. (91071)



- Fixed the annotation overview tooltip in the top-right corner of script editors. (91174)
- Fixed an issue where error suggestions for scripts that run on master pages would not always appear. (91483)
- Fixed a Handlebars issue where certain characters in strings (surrounded by quotes) or escaped content (surrounded by square brackets) would cause an error. (91567)
- Rule trees now initialize faster than before, particularly if there are many rules. Examples of rule trees can be found in the **Conditional Content** script wizard and in the **Dynamic Table** wizard. (92181)

General DataMapper Improvements

- Microsoft Excel based DataMapper configurations can now select the sheet name dynamically, based upon runtime/automation properties. (80908)
  You can therefore re-use the same data mapping configuration even when the name of the target Excel sheet changes over time. For instance, your sheet name might change each month because it includes a date, or the workbook might contain two identical sheets, except the first one is in English while the second one is in French (e.g. Customers-EN, Customers-FR).
  In addition, you can refer to a sheet by its index instead of its name. You can therefore refer to the 1st, 7th or 42nd sheet in the workbook, based on a runtime parameter or some other property
- DataMapper now retains the last selected state for the "*First row contains field names*" property for new CSV and Excel data files. (89126)
- Improved version labels in **File > Save a Copy...** dialog. (90117)
- Fixed issue with missing last row in some Microsoft Excel data files. (90330)
- Fixed error in step property of JSON emulation after a "*Validate*" action. (90369)
- The **JavaScript Expressions editor** is now resizable. (90475)
- Fixed a regression issue that impacted the value of the final statement in a script, in the script engine. (90715)
- Improved usability of the **CSV importation** dialog. (90884)
- Fixed a problem with incorrect interpretation of multibyte characters in **UTF-8 JSON** data files. (91077)
- Data Model expanded state behavior has been improved and is now retained in the configuration file. (91101)
- Improved functionality in **Order and rename fields** dialog in DataMapper perspective. (91370)
- Fixed an issue that occurred when downgrading a template to OL Connect version 1.7. (92213)
- Logging statements were added to log which resources were used when a data mapping operation is performed. (92445)
- Fixed a problem with data mapping text from scaled PDF forms. (92600)

Print Manager

Improved support for dragging and dropping files within the Print Manager. (88048)

Print Wizard\Preset enhancements

- Fixed issues with loading an existing output preset into the Advanced Print and Output Preset wizards when the imposition sheet size is *custom*. (82807)
- Fixed issues when creating output presets with a printer model created from a PPD. If the PPD does not contain maximum page size dimensions and/or PageSize properties on PageSize elements it was possible for the internal media settings in the created preset to be corrupted, resulting in an unusable preset. (90690)
- Fixed color display in the Proof Preview dialog when previewing output via the preview option in the Output Preset wizard and Advanced Print wizard. (91238)
- Changed the default Position Method setting for **Impositioning**. (91999)
    - The initial 'positioning method' setting for impositioning is now "*Center*" instead of "*Scale to fit*".
    - The initial target sheet size for impositioning is changed to *SRA3* (otherwise it would be invalid with Letter format in combination with the new default positioning method "*Center*").

General Output Improvements

- Fixed issue with images not being updated correctly when using a shared folder or local folder for Dynamic Images. (83415)
- Fixed an internal computation for cases where a negative font scale was used. (88553)
- Fix an output creation error for certain input files (for example, those with 0-scaled content) in combination with a license that produces a watermark (proof print). (90222)
- Added support for vertical writing content in passthrough PDF files. (90799)
- Use system encoding as fallback when decoding font names that are not **UTF-8** or **CIDSystemInfo**-based. (90935)

- **PDF passthrough** now retains greater accuracy for matrices, which may be important for character positioning. (90974)
- Fixed a problem with PDF word spacing (Tw) handling when used with CID fonts. (90979)
- Fixed issues with erroneous **Type 3** fonts being output, in some circumstances. (91058)
- The output engine now filters out (invisible) images that are drawn with a scale of zero. (91859)
- Fixed a problem with hyperlink being inserted at an incorrect position. (92128)
- Certain fonts may use multiple character codes for a single glyph. This was not supported correctly in the OL Connect output engine but has now been fixed. (92612)

Print Output Improvements

- Fixed an issue with content drift in certain cases where a single record would produce more than 30 pages of content. (89354)
- Fixed an error that would occur during proof printing, if a sub-directory was selected in the Job Output Mask and that sub-directory already exists. Such as when proof printing a second time. (90617)
- Improved font handling in AFP for large jobs. (90981)
- Fix "Invalid coordinates" exception that could occur when dithering very small images, in some AFP/IPDS/PCL output. (91093)
- Fixed an issue with images being written in incorrect order into the PCL output. (91782)
- Fixed an issue where under certain time sensitive circumstances a data record would be skipped and be missed in content creation. This would lead to missing pages. (92721)

Email Output Improvements

- Barcodes no longer produce unnecessary attachments that use the *.bin* extension. (90946)
- Hidden rasterized barcode elements are no longer added as attachments. (90947)
- Prevent "*TypeError: layoutData is null*" errors if a rasterized element is hidden for email output. (90953)

Security improvements

- Third party libraries updated to more recent versions. (89143)
- A critical vulnerability was fixed where processing a PostScript job might enable an attacker to perform a remote arbitrary code execution with elevated privileges. (83509)
- A number of network client plugins can now validate the server certificate when establishing a secure connection. (89499)

General Workflow Improvements

- Fixed resource error message when selecting the *View Selector* button from the *Properties* window in the **Change Emulation** plugin. (88261)
- Fixed OutOfMemory error in the **Create Output** plugin when *As defined by output preset* is selected. (90214)
- External file path and filename are now being logged at the Information level when executing **Load External File** plugin. (90618)
- Fixed error in the **Retrieve Items** plugin for data records, when output format is set to Metadata. (90701)
- Under certain conditions, extracting text from a PDF using the *AlambicEdit ExtractText2()* method could result in an error that could not be handled from the script environment.
  This has now been fixed, and exceptions can now be properly handled (e.g. using *catch{}* in JScript). (91013)
- Fixed automation variables consumption in the Output engine for the **All-in-one** plugin. (91560)
- A problem was fixed whereby opening the editor options in the **Run script** and **Open XSLT** plugins would result in an error which would prevent access to the editor options. (91565)
- *PlanetPress Capture* has been removed from the installation and is no longer available or supported. (91598)

- Fixed error with **DocuWare Download** plug-in, for situations where a multi search returned more than 50 results. (91612)
- A problem was fixed where using a Text Condition on a PDF would leave the PDF open after the condition, causing sharing violations and *I/O error 32* errors when trying to modify, delete or replace the job file. (91964)
- The Java Runtime Environment used for OLCS plugins has been updated to the current JRE 17.0.10. (92196)
- Fixed a regression issue (introduced in 2023.1), where running a PlanetPress Suite form that used TIFF images might fail with an undefined PostScript error when producing a PDF or printing the job. (92707)

Node.js improvements

- "*EMFILE: too many open files*" errors from NodeJS input were resolved by changing the file libraries used internally. (88855)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the OL Connect Online Help.

They can be found here.

OL Connect Release Notes 2023.2

License Update Required for Upgrade to OL Connect 2023.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This licensing model requires anyone upgrading from Connect versions earlier than 2019.1 to first update their OL Connect License. A dialog box appears as part of the Connect upgrade process requesting that you do this.

It is recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version.

For further details on how to upgrade the OL Update Manager to the latest version and update your OL Connect License see the Update Manager 1.8 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2023.2 and Workflow 2023.2, as well as some important installation information.

Installing OL Connect 2023.2 and Workflow 2023.2

- OL Connect is released as a 64 bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online helps.
  OL Connect: Installation and Activation.
  OL Connect Workflow: Workflow Installation.
- Note that both OL Connect and Workflow come with a 30 day trial license by default.
- It is recommended that you install and use OL Connect Designer and Server under the same user account.

Connect and Workflow installers require Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect Editions

Rather than have entirely different OL Connect products named after our Classic products (PReS, PlanetPress and PrintShop Mail) we have simplified the entire suite of OL Connect products. They will now share unified branding, with Editions separating them. (89676)

- **OL Connect Desktop Edition** is the replacement for ***PrintShop Mail Connect***.
  It has been tailored for outputting variable data print documents directly from desktop machines.
- **OL Connect Professional Edition** replaces ***PlanetPress Connect***.
  It can output all the common print streams (PCL, PDF, PPML, PostScript and variants), as well as emails and web pages.
- **OL Connect Enterprise Edition** replaces ***PReS Connect***.
  It can output AFP and IPDS in addition to all the common print streams (PCL, PDF, PPML, PostScript and variants), as well as emails and web pages.
  Enterprise is our most powerful version, supporting load distribution across an array of machines.

Security improvements and library updates

- Native binaries (EXEs and DLLs), have been recompiled using the latest available versions of libraries and MS Visual Studio 2022. (86866)

- Adobe PDF Library updated. (89810)
- Jetty updated to version 10.0.14. (82387)

Welcome screen improvements

- Recently used job and output preset files are now shown in the recently used file list in the Welcome screen. (88865)
- Fixed a bug where the welcome screen would go blank after you opened a preset. (90069)

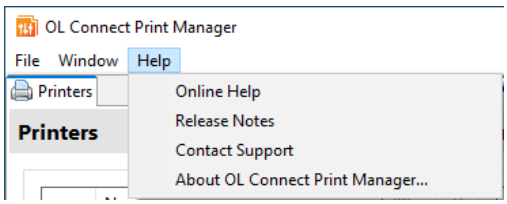
Installer Improvements

- **Silent Installer**: The default locale for Connect is now being set according to the system regional format in Windows, unless specified through the silent installer properties directly.

  The language silent installer property now requires just a two letter abbreviation for a specific language. (89376)
- When performing an upgrade installation, any existing setting of the server configuration (**server.security.enabled preference**) is retained, rather than always being reset to true. (90236)
- Improved Connect installer language and translations. (90482)
- Fixed issue with verbose Windows password containing special characters causing Connect Server installation to fail. (90555)

Print Manager

- Product version now displayed on splash-screen. (90264)
- Improved the information available under the **Help** menu option. (90266)



AFP Merge Tool

Improvements made to the Help. It now displays the OL Connect version and year. (89504)

The command line option for specifying a base directory (-b or --basedir) was not working. This has been fixed. (90197)

OneDrive Document folder now Project Default Root

Added support for the OneDrive Document folder.

The OneDrive Documents folder becomes the first choice for the project's default root folder. (89841)

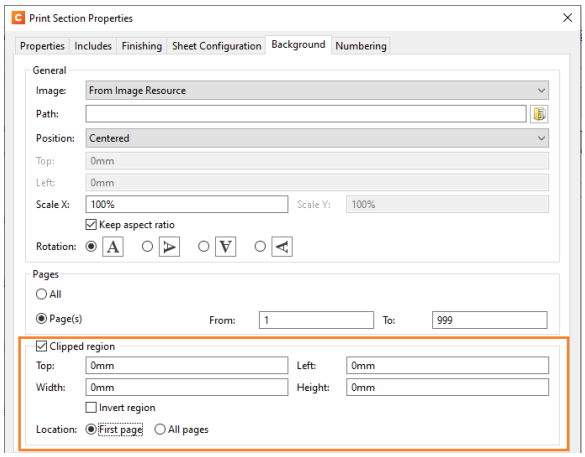
Automatically load data file with Template

The Designer now automatically re-opens the last data file associated with Template, when the Template is loaded. (87741)

This feature is applicable when using the Data Mapping wizard to load data, rather than a Data Mapping configuration.

Background image clipping

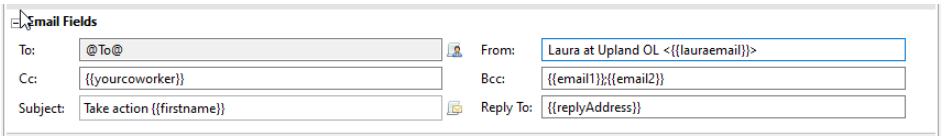
Added a feature that allows you to clip areas from section backgrounds (including PDF Data Mapper backgrounds). (90187)



This option allows either hiding a part of the original image from view, or displaying only a specific portion of it.

Expressions in email fields

Text based Handlebar expressions are now available for all email fields. (90072)



Just drag and drop data fields into the relevant areas.

Expression support for snippets

The **loadhtml()** function has been improved to enable loading Handlebars snippets in addition to HTML snippets. This replaces the lengthier `Handlebars.render()` notation. During this process, expressions are automatically evaluated. (90131)

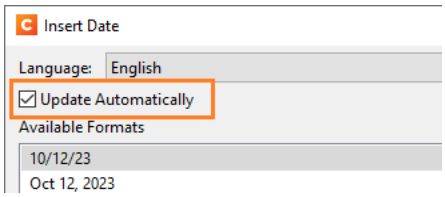
Inserting today's date

- Added support for a built in helper which inserts today's date, in the local date format: `{{date today}}`. (89412)

  Some usage examples:

  ```
  {{date today}}
  {{dateShort today}}
  {{dateLong today "nl"}}
  {{date today "MM//dd/yyyy"}}
  ```

- Designer now inserts this helper when inserting a date with the "*Update Automatically*" option selected. (90713)



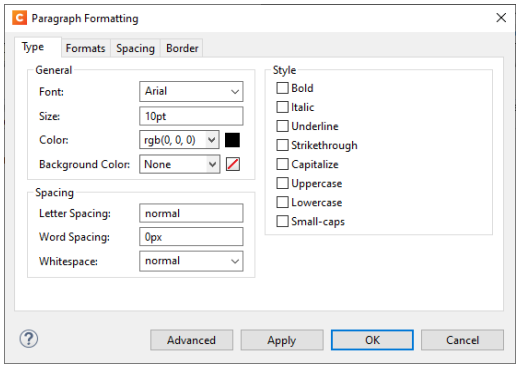
Scripting/Handlebar Improvements

- Fixed an issue with the "`const`" keyword in control scripts. (86085)
- **Post pagination**: Fixed issues with `paginate()` not working correctly with detail tables, which led to unnecessary headings on new pages. (89405)
- **Currency** formatter now uses non-breakable space instead of a regular space, to ensure the currency symbol and the data stays together, even if a line ending falls within. (89469)
- Adding a field from a detail table to the main editor now inserts an expression pointing to the exact record and field from the detail table.
  For example: `{{detail[3].UnitPrice}}`. (89488)
- If a string is passed to one of the Handlebar **Date** functions we now attempt to convert the string to a date, rather than always generating an error. (89933)
- Fixed issue with inserted detail table in Handlebars snippet producing incomplete HTML. (90109)
- Fixed an issue with scripted calls to `Handlebars.render` or `Handlebars.compile` that occurs when switching to Preview mode. (90519)

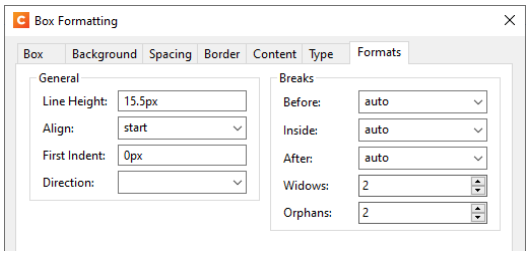
General Designer Improvements

- Removed unnecessary **Whitespace** option from the **Attributes** pane when viewing a master page. (22554)
- Elements on master pages that use **absolute positioning** can now be positioned with the top, left, right or bottom style rules or attributes.
  This means that they do not support the offset-x, offset-y and anchor attributes as their counterparts on section pages. (87208)
- Double clicking on a template file now brings that file to the fore, regardless of whether existing file was open and minimized. (87292)
- In the **Add Parameter** window, a string value that looks like a Date entry will no longer be automatically converted to Date. (87865)
- Improved handling of boundaries at design time. (89406)

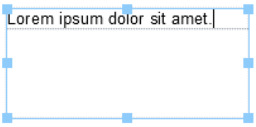
- Added the **Type** tab to the formatting dialog for **paragraphs**. (89516)



- The Box properties dialog now has a **Format** tab added. (89518)

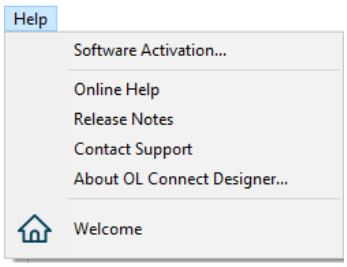


- Inserting a **Position Box** now populates the box with starter text. (89522)

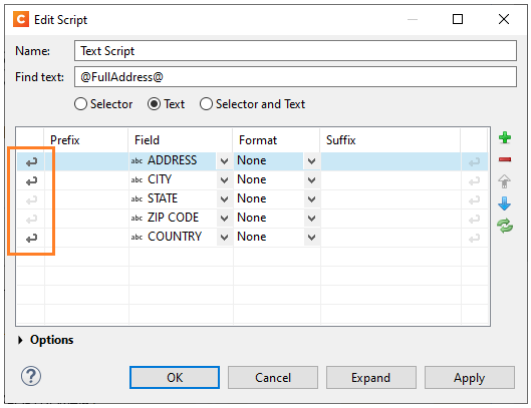


- The bottom edge of the body element now aligns with the bottom margin of the last page, allowing you to leverage the CSS property `{position:absolute; bottom:0}` to align elements to the last page.
  Note: This does not work with Absolute Positioned objects, which are anchored to the page. (89542)
- Fixed bug with Handlebars that prevented detail rows being shown in **Preview** mode when the detail table name contained an underscore followed by a numeric character. (89668)
- Improved **drag & drop** behavior for images and data fields in the Designer. (89784)
- Fixed issues with **Preflight** progress bar. (89795)
- Fixed an issue where **double clicking** a package file, that contains multiple templates and/or data mappers, did not open up the package in Designer. (89844)

- Fixed backwards compatibility issues with some conditionals in Templates written prior to Connect 2021.1. (89921)
- Improved the information available under the **Help** menu option. (89923)



- Improved **Properties** dialog for versioned DataMapper/Template configuration files. (90041)
- **Product version** information (version number and license edition) can now be copied from the About box. (90162)
- Fixed issue with Designer randomly throwing Assertion errors in some Templates. (90549)
- **Word-based template** import now correctly sets the background color of table cells. (90604)
- Support added for passing either a HTTP or HTTPS based URL to the **resource** function. (90662)
- Designer now saves the state of the **Show Edges** option, and returns to that state upon reopening the application. (90721)
- Added a column to the text script wizard and text helper wizard to **simplify adding a line break** to a script entry. (90835)



General DataMapper Improvements

- **Improved performance** for all data types in general and XML in particular. (74449)
- Improved performance for **XML files** with pretty format. (86667)
- Implemented support for **JSONPath functions**, such as `sum()` and `avg()`. (87721)
- Fixed **PDF boundaries** based on text with negative add/remove pages values. (89054)
- Improve display of Data Model Record[x] label in higher definition monitors. (89632)
- Applied a patch to the supplied **PCL converter** to avoid crashing on certain input files. (89861)
- Applied a patch to the **PCL converter** to improve PCL macro handling. (90184)
- Improved error handling when the data model value cannot be **cast** to the intended type. (90367)
- Improved extraction field creation. The "*insert selection*" no longer appends a semi-colon at the end of the JavaScript. (90477)
- Simplified field creation for **JSON** data types. (90480)

New output file name variables for Banner pages

- Several oft-used variables for output file names can now we used in the **Additional Content** feature (Text, Image, OMR or Barcodes) of the **Output Preset**. This simplifies displaying file names on banner pages.
  For example `${template.name}` and `${file.nr}`. (90432)
- Two new output mask variables `${job.name}` and `${job.owner}` can be used for the **Job Name** and **Job Owner Name** fields from the preset printer output options.
  When separating output by segment, document set or document, these will also exist at that level of separation. (90432)

Print Preset enhancements

- Recently opened **Job and Output Preset** files now appear in the recently used file list under both the File->Open menu and Welcome screen (88843/88865).
- **Leading or trailing spaces** will now be automatically removed from the job output mask and/or job name in the Output Preset. (89808)
- The **Print Dialog** now loads and displays the latest version of the last used Preset instead of displaying <Last Used Settings> if you print directly via the Print Dialog and not the advanced Print Wizard.
  If the advanced Print Wizard is used and the preset file no longer exists then the <Last Used Settings> will be displayed and loaded. (90400)

General Output Improvements

- A problem was fixed that prevented accessing a **USB token** (PKCS#11 hardware key-stores) for digital signatures on PDF output. 89258)
- The *Data Mapping* field is now empty for new **Job Creation** presets. In the dropdown, you will now see a selection of recently opened data mappers. (89675)
- Fixed issue with Barcode div line height (usually linked to the font size) being smaller than the barcode div height, which could generate incorrectly clipped barcodes. (89696)

- Extensive changes to internal handling of OpenType/CFF to improve support for **large CJK CID fonts**. (89724)
- Improved support for external Japanese font **Meiryo** (in the TrueType Collection). (89725)
- **Merge engine** now errors out when failing to draw some parts of a page, instead of ignoring the error and producing only part of the intended page content. (90536)
- Fixed an error that would occur when using an image containing invalid metadata, such as an invalid date/time field. (90553)

Print Output Improvements

- Applied fix for linear color profiles in PostScript output to bundled LCMS. (89732)

Email Output Improvements

- The **sequence of email attachments** has been modified, placing PDFs generated from the print context before other attachments. (70638)
- Improved **address validation**. Now we only generate a "*no recipients*" error if the total number of recipients ("To" plus "CC" plus "BCC") is empty. (89591)
- Improved **Send Test Email** error handling for data mapping. (89638)
- Emails missing their dynamic attachment will now longer be sent. (90183)

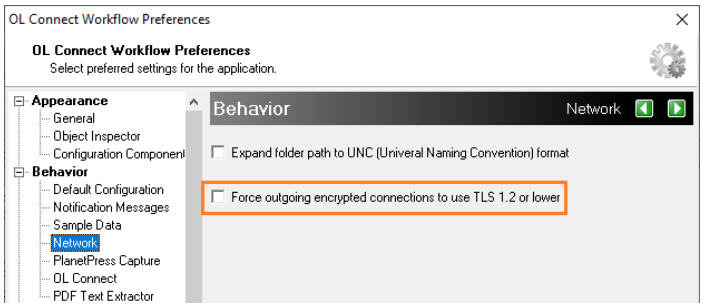
Security improvements

- Third party libraries updated to more recent versions.

Performance improvements

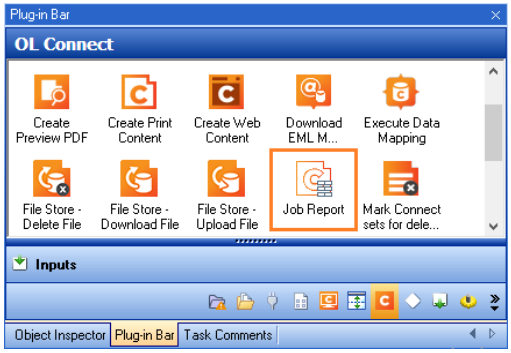
- Performance issues were encountered with Connect Plugins using the **HTTPS** protocol in Workflow. This was due to TLS 1.3 encryption being introduced in 2023.1.

  A new option "Force encrypted connections to use TLS 1.2 or lower" is now present in the **Preferences** settings to fix these performance issues if you encounter them. (90116)



General Workflow Improvements

- The **JobReport** plugin has been integrated into Workflow. (86947)



- A new `ExtractPDFVTMetadataAsJsonStr()` method has been introduced to the AlambicEdit API **to extract PDF/VT metadata** and return it as JSON string. (89367)
- Product version information (version number and licensed edition) now displayed on splash-screen. (90258)
- Fixed issue with **Telnet Input** not correctly interpreting special characters sent over to it due to an error introduced in an earlier version. (90365)
- Logging: When a self replicating process prevents the service from stopping, that process name is now logged in the Workflow service log. (90407)
- Fixed issue with **PlanetPress Classic Suite 7 forms** containing PDFs not being imported properly into Workflow. (90513)

Node.js improvements

- Fixed issue with accumulated temp files for Node.js input requests. (89697)
- Node.js input errors "*EMFILE: too many open files*" were resolved by changing the file libraries used internally. (88855)
- Fixed an issue with Orphan files not being properly removed from Node.js folders. (90325)
- Spaces and CR/LF character are now removed from the request file's file list. (90442)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2023.1

License Update Required for Upgrade to OL Connect 2023.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version, then install Connect 2023.1.

For further details on how to upgrade the Update Client and update your Connect License see the Update Manager 1.5 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2023.1 and Workflow 2023.1, as well as some important installation information.

Installing OL Connect 2023.1 and Workflow 2023.1

- OL Connect is released as a 64 bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that both OL Connect and Workflow come with a 30 day trial license by default.

Connect and Workflow installers require Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

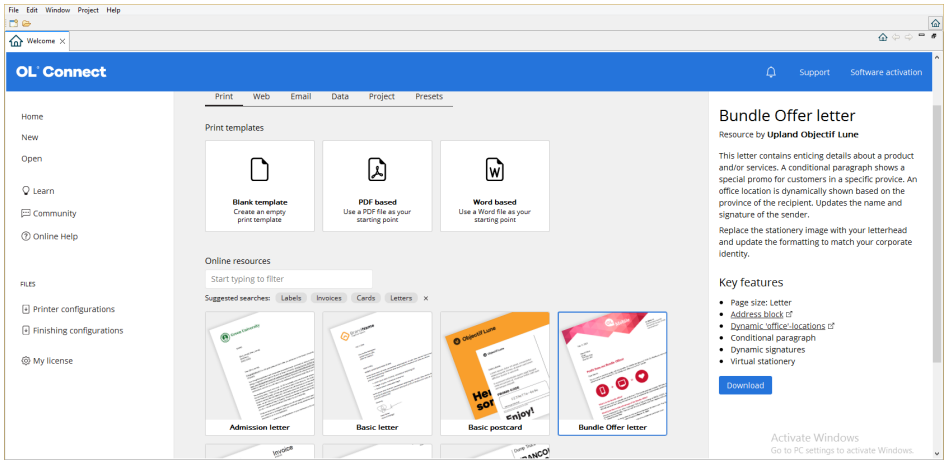
It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2023.1 Improvements

Welcome Screen improvements

A completely redesigned Welcome screen now allows you to perform the most common tasks from a single place.

The overhauled **New section** not only lets you create new documents but also download online resources.
Selecting an online resource shows a short description explaining its use case and detailing the techniques it uses. That information may even include links to how-to's, blog posts or pages in the online Help.



The new **Learn section** shows our most recent blog posts and lists the most popular hands-on tutorials and how-to's. It provides access to a central repository of OL Connect content, including sample files, example projects, tutorials and blog posts.

The **blog posts** provide information about new features, tips and techniques to increase productivity and the tutorials introduce basic features of OL Connect for those just getting started.

All this brings OL Connect related knowledge right to your doorstep.

MariaDB update

Updated the MariaDB which is installed with Connect to version 10.11.2. (89330)

Java update

Updated the Java platform which Connect runs on to Adoptium Temurin 17.0.6. (87357)

General Connect Improvements

- Better support for MS SQL Server backend databases. Retry code was added to work around deadlock issues. (87653)
- The database *vendor* and *version* are now included in the logs when the database connection is first made. (87325)
- Improved error messaging and localization. (88060)
- Fixed a session data memory leak issue in Connect Server. (88768)
- Old copies of the JavaSysMon native library (DLL) that were previously left behind after a Connect product had run, will now be cleaned-up at next product start-up. (89361)

Security improvements

- Connect dependencies have been updated to recent versions. (87357)
- Improved user account validation when testing credentials for Connect service, in Connect Installer. (88547)

Installer Improvements

- Installer language selection is now localized. (86625)
- We have altered the setup so that it is more robust. (86520)
  Changes include:
  - When a user enters "Username", we make the assumption that the user really means "HOSTNAME\Username"
  - When a user enters ".\Username", we make the assumption that the user really means "HOSTNAME\Username"

AFP Merge Tool

The AFP Merge Tool has been added to OL Connect installations.
This is a command-line tool that can embed external resources into an AFP file and concatenate multiple AFP files into one. (89088)

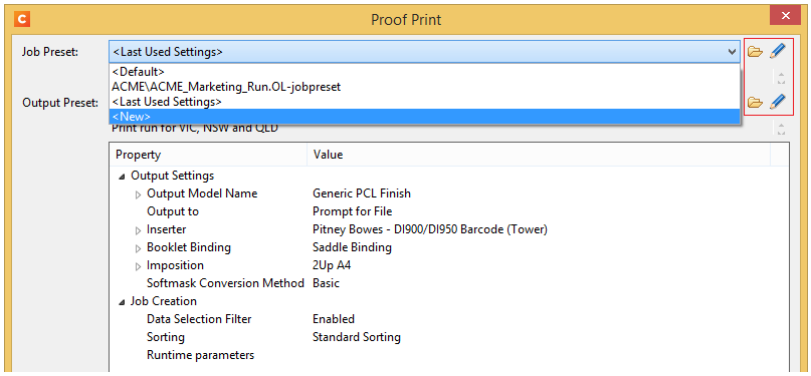
REST API Improvements

- The All-in-one (print) operation can now return produced xxxSetID(s). (83620)
- Added the ability to restrict searches to a discrete set of *DataSets*, *ContentSets* or *JobSets*. (88249)

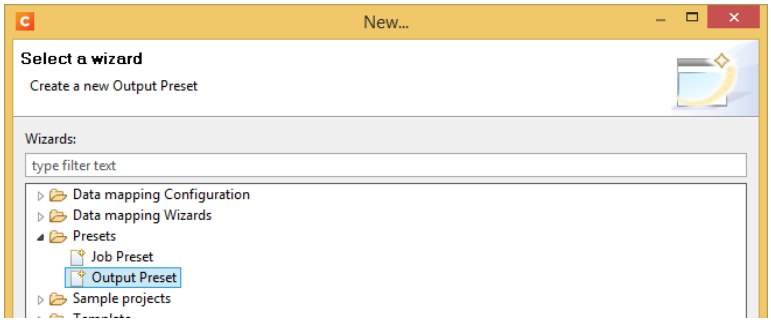
Job and Output Presets improvements

Job and output presets handling is changed from storing them in Connect Designer's workspace, to dealing with them as regular files that sit in a folder under your control. This way, job and output presets can be managed alongside templates and data mappers that belong to the same project or customer. (88926)

The Print and Proof Print dialog have been simplified to make them less cluttered, and simpler to use. They now provide plentiful ways for selecting, opening, editing and saving job output presets. (88903)

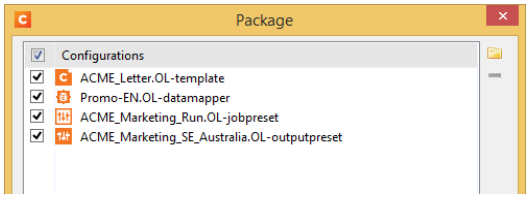


Added job and output preset items to the **File>New** wizard, so you can create presets in the same way as templates and data mappers. (88618)



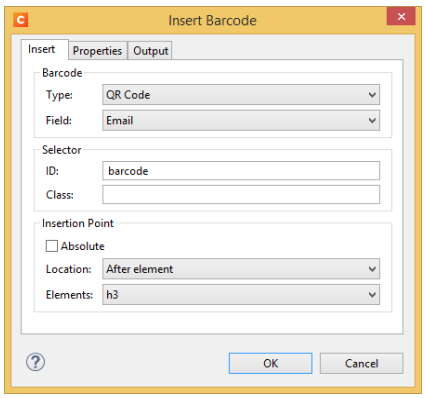
Job and output presets can now be opened for editing directly from the "Project files" view. (88615)

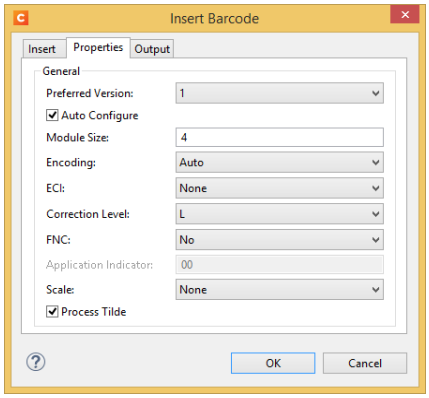
Presets are now handled as regular files in "Send to Workflow", "Send to Server", or "Package" options. (88922)



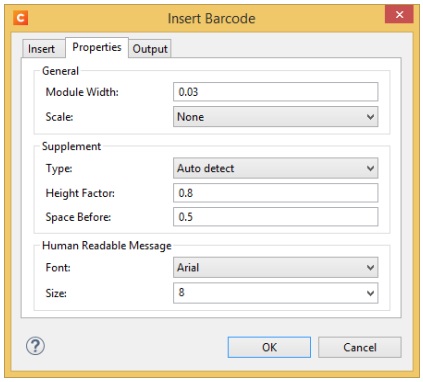
Barcode Improvements

- Barcode properties can now be added at the time of barcode insertion, rather than afterwards in the Barcode properties dialog. The barcode insertion dialog is now split over three tabs, **Insert**, **Properties** and **Output**. (87729)

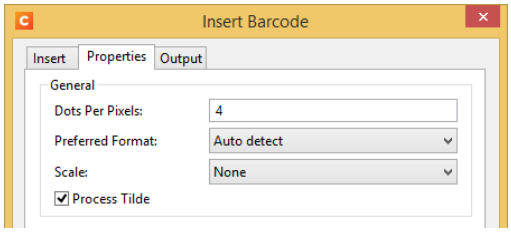




- The guardbars and human readable message are now always enabled for new barcodes, with the human readable text present at the bottom of the barcode. This is the case for all new **UPC-A**, **UPC-E**, **EAN8** and **EAN13** barcodes. (45352)



- Updated the barcode properties dialog for the **GS1 Datamatrix Barcode** properties, showing the correct properties for the barcode. (67410)



- Improved the validation of the Application Indicator property for **QR codes**. The Application Indicator is only used for the QR code, when FNC is set to Second, otherwise it's disabled. (85682)
- When a barcode is added to a section that has Handlebars expressions enabled the Designer will now add a Handlebars expression instead of creating a script. (87735)

Windows Scaling improved

Improved support of high DPI monitors in the Designer. Users with high DPI monitors typically use the Windows scaling feature which we have improved support for. (88412/89295)
Other fixes include:

- Dragging a data field onto the editor while holding the CTRL-key would insert the box in the wrong location. (86740)
- The helpers for dragging and resizing absolute positioned boxes were shown in the wrong location. (88406)
- The rulers on screen were set to the wrong size. Also dragging of guides would not follow the mouse correctly. (88428)

Improved handling of missing images

Several improvements were made. (87678)
These include:

- The `data-broken-image attribute` is added to the `<img>` element to specify that the image could not be found.
  This can be used in the selectors in scripts and CSS to apply custom styling or to insert a fallback image.
- Provide Alternative text (`<img src="" alt="">`) via the Attributes panel. When specified, this text will be rendered in the output in case the image cannot be found.
- No longer show a red cross for broken images in print output. We only show a red cross in design or preview mode in the Designer, and we only do this if the `alt` attribute is not present.

Improved Microsoft Word importation

When creating a new template based on a Word document, mail merge placeholders are replaced by handlebars expressions or @-placeholders depending on the *Evaluate handlebars expressions* setting on the active section. For example «foo» will now be replaced by {{foo}}. (89117)

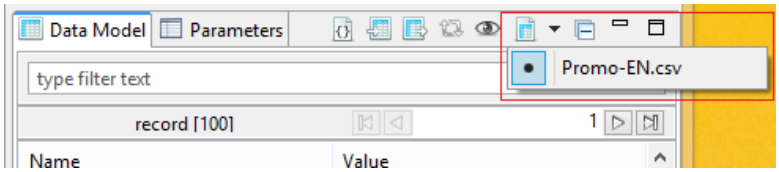
Improved Versioning

Several improvements were made to Versioning. (88627)
Specific improvements include:

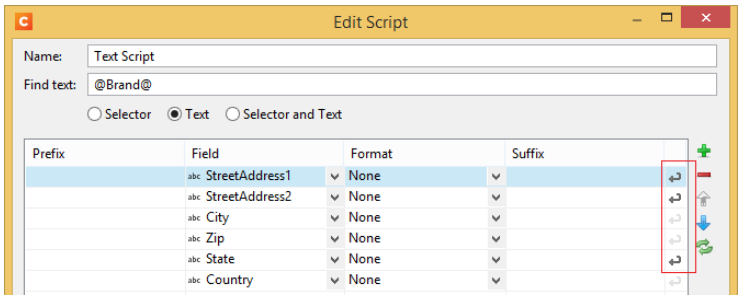
- Improved Project view. The option to "*Reveal in File Explorer*" has been introduced for the selected item in a versioning project. (88611)
- Numerous useability options are now available in the project file's context menu. (88643)
- Standardize icons in Project Menu. (88632)

General Designer Improvements

- Introduced simple way to see what data file is currently in use and to swap to another. (77490)

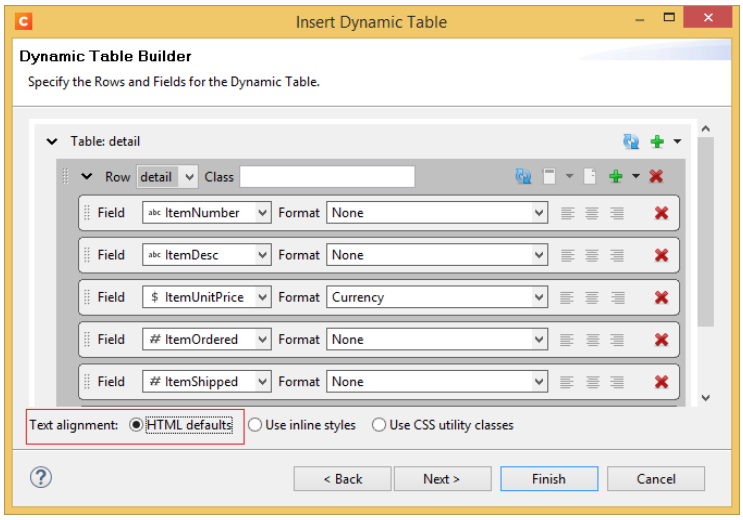


- Removed warning about missing `<br>` after replacing source and switching back to Design mode. (78852)
- In some scenarios the `src` attribute of an `<img>` element could be converted to an absolute path. This has been fixed. (82049)
- Fixed an issue with the positioning of proportionally scaled SVG barcodes. (88380)
- Added a column to the text script wizard and the text helper wizard to make it easier to add a line break to a script entry. (88509)

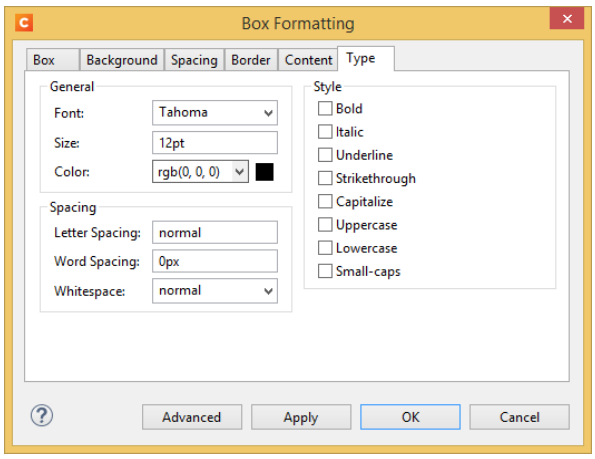


- Selecting text in an absolute positioned box is now contained to the box itself. (87477)
- The defaults for widows and orphans for tables now set to 1 so that small tables with tall rows are automatically paginated. (89076)

- Added new Text Alignment option of *Default html* which disables the text alignment options in the **Dynamic Table Wizard**. This allows you to style the text alignment of the columns yourself. (87214)



- A new *Type* tab has been added to the **Box Formatting** dialog. This allows you to apply text formatting styles on the text within a box. (87718)



- The collapsed and expanded state of folders in the resources view is saved with the template. When the template is opened (on a different machine) the resources view is restored. (87738)
- Handlebar templates can now use the following *data variables*: (87687)
    - - `@first`: this variable is true when the current record is the first of a data set.
    - - `@last`: this variable is true when the current record is the last of a data set.

    - `@index`: this variable contains the zero-based index of the record within its data set.
- Fixed issue with opening CSS files that contain extremely long strings. (87251)
- Fixed an issue in script wizards with the Field drop-down becoming unresponsive after selecting a runtime parameter. (88171)
- We no longer log an error when the resource function is used to check the existence of an HTML snippet (or any other non-image file). (87958)
- *TextHelpers* now have a Data scope which affects the fields that are available in the field column in the entries table. The Data scope allows text helpers to be used in detail tables with the same scope. (88282)
- Designer now remembers the state of resource panel in Templates. (88596)
- Fixed an issue with field names containing periods in text script wizards. (88825)
- Fixed an inconsistency with script profiling results. (88887)
- You can now turn a static table into a dynamic table through the Attributes panel. (88907)
- The defaults for widows and orphans for tables are now set to allow small tables with tall rows to be automatically paginated. (89076)
- In Handlebars mode, field names containing special characters are now escaped when they are dragged to the main editor. (89096)
- Support was added for Handlebars partials in the main (HTML) content editor.
  For technical reasons the implementation slightly differs from the partials in Handlebars snippets and uses the *plus* (+) sign instead of the *greater than* (>) sign (`{{+snippets/officelocation.hbs}}` or `{{+officelocation}}`). (89301)

OL Connect 2023.1 DataMapper Improvements

Improved Data Model filter

Improvements include (87773)

- The Data Model panel now has the same appearance as the filter options shown in the Scripts panel and the Resources panel.
- The filter now operates on either field or table names as well as field values.

JSON Improvements

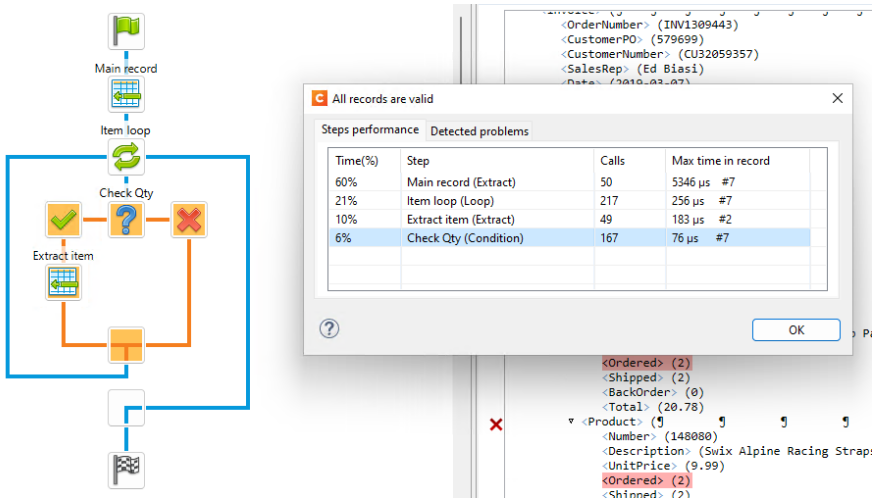
- **Add Repeat** action now automatically sets the name of the data table to the name of the property so that you don't need to manually change this from '*detail*' to the name of the property. (87705)
- Implemented support for JSON key nodes with spaces. (88571)

File Extraction Improvements

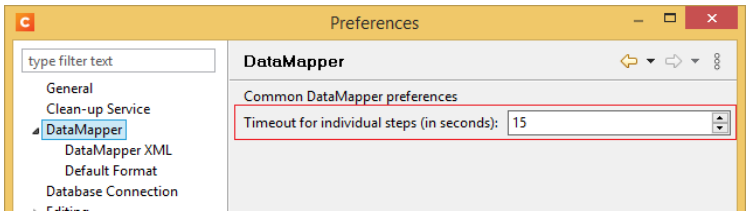
- Extra tooling added to **LincPDF** to assist in identifying issues with conversions from PCL to PDF. (86810)
- Handling of PDF files with incorrect TrueType font data has been improved. This improves data mapping of certain PCL files that are converted to PDF using the Connect **LincPDF** tool. (86936)
- Improved handling of **AFP input** files that rely on external resources. (88099)
- Fixed an issue with data extraction from PDF files that feature an unusual Unicode font encoding. (88560)
- Fixed an exception with PostScript input on machines with more than 8 cores. (88603)
- Extend Fontmap format and look for user-supplied Fontmap files in PDF resources folder. (88997)

General DataMapper Improvements

- Improved *Save* option in DataMapper JavaScript dialog. (80018)
- The *Validate all records* option has been enhanced to display a table of all operations performed and the time spent in each step. This allows you to identify the steps that consume the most time and that could therefore be candidates for optimization. (86657)



- Changed the behavior of the File Browse dialog in DataMapper wizards, so they now default to the last selected location. (88164)
- Improved scaling of tables when using high Windows DPI settings. (88410)
- Implemented check for infinite loop in DataMapper script engine. Any script with an infinite loop is canceled after the time set in **Preferences->DataMapper->Timeout** option. (88421)

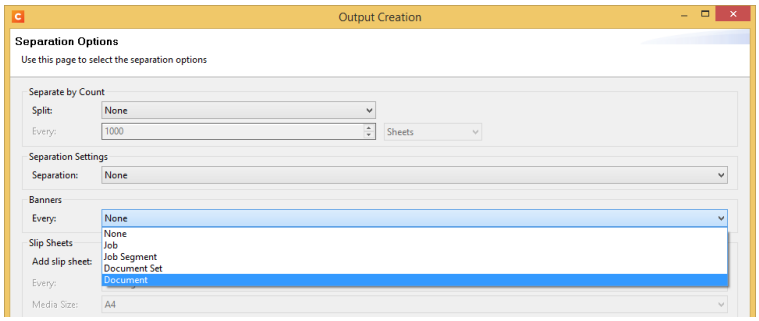


- Boundaries control width are now properly framed when changing to *On Text* boundary. (88527)
- Fixed issue with MySQL connections from script engine. (88622)
- Support non-UTF-8 font names in PDF files by extrapolating the encoding from the font's CIDSystemInfo if the font name is not valid UTF-8. (89035)

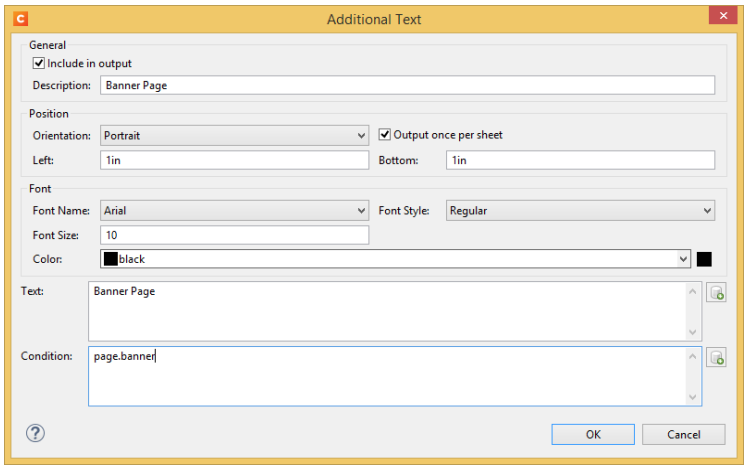
OL Connect 2023.1 Output Improvements

Added Support for Banner Pages

You can now incorporate banner pages to print jobs. (88372)



You can also add custom information to the banners, to better support print production processes. (88475)



General Output Improvements

- Relaxed the validation rules when using a file mask to allow embedded JavaScript. (74168)
- Fixed "*image optimization*" triggering an exception when encountering Separation- or DeviceN-color images in background PDFs. (87701)
- The formatting of **operation duration** values in a print summary will no longer cause a job to potentially fail when a duration value is reported as being negative. (87999)
- Fix crashes on DeviceN colorspaces with NChannel information on some but not all of the color components. (88640)

Print Output Improvements

- Enabled transparency for **PPML** output, so content is no longer flattened. (87148)
- Support has been added to allow external sorting in job creation based on the document sheet count. (89121)
- Added support for IPDS/AFP MICR Printing on Ricoh Continuous-Feed printers.(89176)

Email Output Improvements

- Fixed an issue where the style element containing media queries or font face definition could end up in the body of an email instead of the head. (83002)

Workflow 2023.1 Improvements

Security improvements

- Add support for TLS 1.3 (89124)
- Security vulnerabilities were fixed. (86169)

  As part of this effort, support for **Novell NetWare** has been removed.

- Other Workflow 3rd party libraries to latest versions. (86999)
- OpenSSL 1.1.1t is now the default openSSL library used by Workflow helpers and plugins. (88782)

Memory leaks fixed

- Memory leaks were fixed in the plugins used for **OL Connect Send** solutions. (57329)
- Fixed a memory leak in processes using **local variables**. (87193)
- A memory leak was fixed in the Workflow **custom plugins** framework. (87911)
- A problem was solved where a **Folder Capture** plugin configured to use regular expression and sort on either creation of modification date was leaking system handles. (87972)
- Memory leaks have been greatly reduced when using **self-replicated processes**, and eliminated when those processes don't use AlambicEdit. (88362)

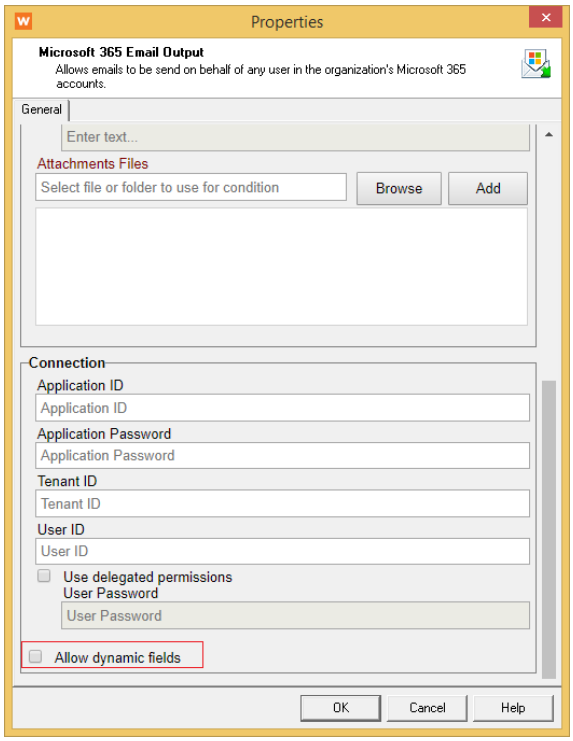
Several further memory leaks were fixed in specific plugins. The plugins affected are:

- The suite of **OL Connect** plugins. (87464)
- The **PDF Splitter**. (88083)
- The **Create Web Content** plugin. (88206)

General Workflow Improvements

- In the **PDF preview window**, it is now possible to browse up and down using the Page Up/Down keys. (82507)
- A **Create PDF** problem was fixed where PDF keywords were duplicated when they were modified. (84362)
- **Input SFTP/FTPS** plugin - When plugin is created, the default mask is now set to "*.*". (87886)

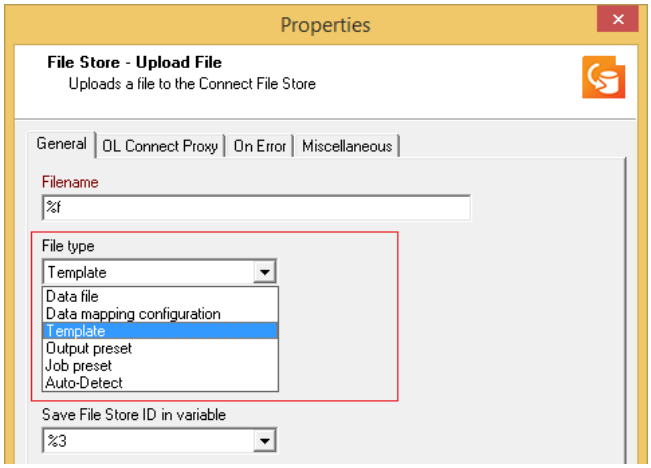
- Fixed deadlocks and potential data loss events when the **Telnet Input** server is under heavy load.
  Fix also cleans up the logs by grouping lines together per connection. (88203)
- Job creation options set to 'None' in the **All in One** plugin were not being handled properly when running in a language other than English. This has been corrected. (88239)
- Improved plugin **File Store - Upload File**. Different resource files can now be uploaded. (88284)
- You can now specify dynamic values in the connection parameters for all **Office 365** plugins by using variables and data selections to populate those fields. (88289)



- Refined the validation for the mail host port in the **Create Email Content** plugin, so that it now accepts all valid TCP ports. (88321)
- Fixed "*Sort By Name*" issue for duplicated names in the **Configuration Components** pane. (88445)
- Queries with a large amount of parameters can now be allowed by modifying the *parameterLimit* value of the server. (88651)
- **Office 365** email output plugin now allows HTML formatted email messages to be sent. (88778)
- Expanded Workflow variables are now supported in XSLT script files. (88963)

- Fixed a **WinQueue Input** issue where capturing print jobs in EMF format would cause access violations when polling print queues for jobs to capture. (89157)

- **File Store - Upload File** task now has a *File type* selection which can added in the task description and hint. (89180)



- HTTPS forwarding is now properly made for non localhost URLs. (89198)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2022.2

License Update Required for Upgrade to OL Connect 2022.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version, then install Connect 2022.2.

For further details on how to upgrade the Update Client and update your Connect License see the Update Manager 1.5 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2022.2 and Workflow 2022.2, as well as some important installation information.

Installing OL Connect 2022.2 and Workflow 2022.2

- OL Connect is released as a 64 bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that both OL Connect and Workflow come with a 30 day trial license by default.

Connect and Workflow installers require Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2022.2.3 Fixes

JSON Support fix

An issue was discovered relating to the JSON support introduced in 2022.2. (89123)

The behavior has been changed, so that when Connect now encounters JSON sample data in the Designer or REST endpoints that directly accept JSON data:

- If the template has a data model with a corresponding detail table, the object is mapped to a detail table with a single record (legacy behavior).
- Otherwise; the object is mapped to a field of type JSON (default behavior).

Workflow Printer fix

The **Workflow Printer** was not being installed properly. This has now been fixed. (89012)

DataMapper fix

Fixed an issue with DataMapper data extraction from PDF files that feature an unusual Unicode font encoding. (88560)

OL Connect 2022.2.1 Fixes

Java fix

An issue relating to the updated Java platform was discovered in the Technical Preview of 2022.2.0. This issue has now been fixed. (87664)

Handlebar fixes

Some issues were discovered with the new Handlebar implementation. These issues have now been addressed. (88358/88414/88438)

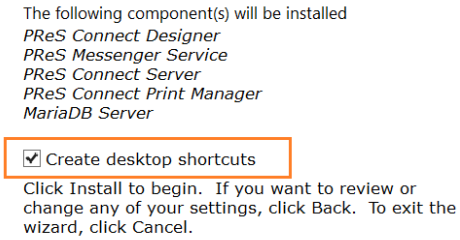
OL Connect 2022.2 Improvements

Upland branding

Connect branding has been updated, to reflect Upland Software acquiring Objectif Lune. (87143)

Installer improvements

- The OL Connect installer has been heavily compressed and is now about half of its old size. (87395)
- Installations can now be *rolled back*.
- Desktop shortcuts are now an optional selection. (86794)



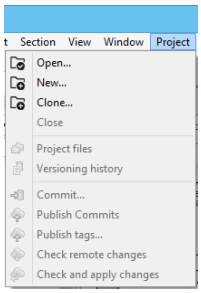
- Connect and Workflow now share the same Start Menu group.
- Fixed some issues with the installer flow, in regards to the default button and when using the Back button. (85386/86201/86397)
- Can now select language for installation in a silent installation. (86190)
- Fixed an issue with detecting/installing Microsoft VC++ 2013 redistributables. (86518)

- Setup failed to install the Connect Server Service if the Windows user used for running it had a password with an '&' or '<'. This has been fixed. (87073
- Fixed issue with upgraded older versions of Connect not successfully setting the Server user "olc-user" password, which would cause the Server to error on start up. (87089)

Versioning improvements

Improvements made to versioning. (84622)

New options have been added to allow sharing Projects through the use of a remote repository. Common repositories include GitHub, BitBucket, GitLab and Azure DevOps.



- **Clone** Create a new local Project from a remote repository
- **Close** Allows a project to be closed in Designer
- **Publish Commits**: Upload local changes to the remote repository
- **Publish tags**: Upload local tags to the remote repository
- **Check remote changes**: Download the changes from the remote repository and show them in history view
- **Check and apply changes**: Download the changes from the remote repository and apply them locally

Other Versioning improvements

- The **Project History** view now shows the remote history. (84630)
- Improved language translations. (85975)
- Introduced *Tags* to Versioning. A *Tag* is a name that gets affixed to a particular commit to identify it in a special way.
  For example, tagging a particular commit to identify which resources are used in production. (86076)
- When a sample project is created with Wizard, it is now versioned by default. (86600)
- Versioning projects are created in *<user.home>/Documents/OL Connect* folder by default. (86470)

Improved support for Microsoft Word

- *Stylesheets* created when importing content from Word documents or when creating a template based on a Word document, can now be edited in the **Edit Stylesheets Dialog**. (86050)

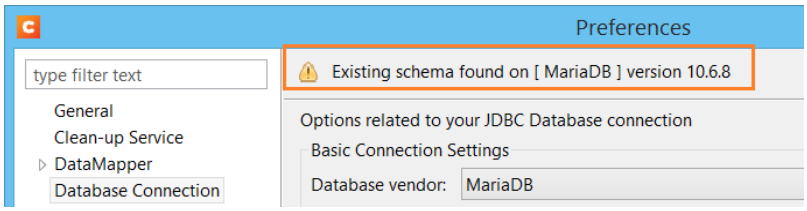
Java update

Updated the Java platform which Connect runs on to Adoptium Temurin 17.0.2. (85093)

MariaDB update

Updated the MariaDB which is installed with Connect to version 10.6.8. (85802)

You can now check what version the database is through the **Preferences->Database Connection**, "Test Connection" button. (81019)



MariaDB now defaults to only logging errors. This prevents unimportant warnings from cluttering the Windows Event Logs. (86487)

Other libraries updated

Upgraded to Rhino library allows for the use of ES6 features like arrow functions and for-of loops throughout all Connect components. (86527)

General Connect Improvements

- Added support for downloading intermediate certificates (AIA) not present on the local machine. (84947)
- Datamapper, Merge and Weaver engines that fail to launch will continue attempting to launch 15 times over 10 minutes (by default), but now there is a minimum wait interval of 30 seconds (by default) between launch attempts to optimize the chance of a successful retry. (86422)
- Improved deadlock prevention and handling with MS-SQL Server when sending datarecords from the server to a merge engine. (86615)
- Fixed issues with merge engines becoming unavailable when attempting to add them to a job that is running a Datamapper config containing a post-processor script, with the data records being persisted to the database. (86689)

Print Manager Improvements

- Fixed PostScript preview. Properties were not always being read, causing previews to fail. (86753)

REST API Improvements

- Add option to convert DM records to simplified JSON. (85515)

OL Connect 2022.2 Designer Improvements

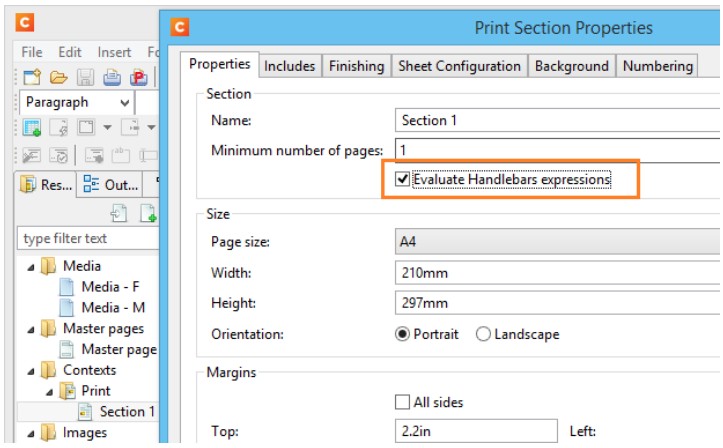
Handlebars - simplified personalization using expressions

OL Connect Designer now supports Handlebar expressions in the main editor.

An expression is a variable wrapped with double curly braces, for example: {{firstName}}. These are automatically replaced by the value of the matching data field and don't require a User Script. This drastically reduces the number of scripts for a typical OL Connect template.
Expressions are simple text strings, this makes them easy to insert, rearrange and style. Above all they are optimized for performance.

Specific changes made:

- Added a new option to automatically process Handlebar expressions in sections and master pages. This is turned on by default for new templates, but it needs to be enabled in the Properties dialog of sections for templates created in older versions. (86220)



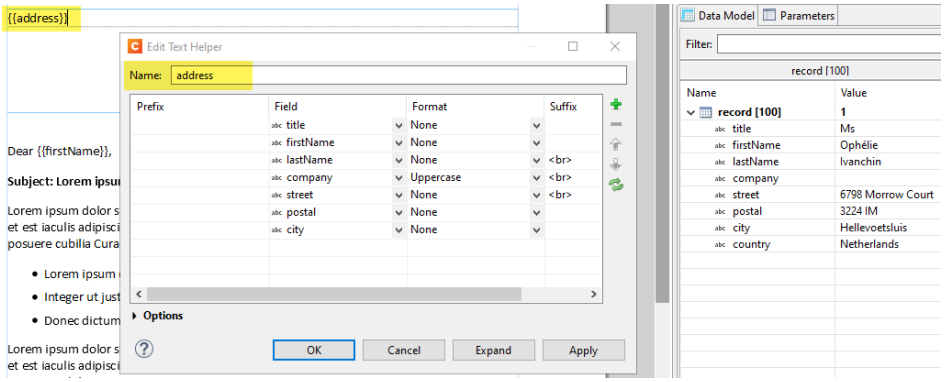
- Add Handlebars expression support to the Dynamic Tables wizard. (86226)

Text Helpers for Handlebars expressions

Introduced Handlebars Text Helper wizards. (86496)

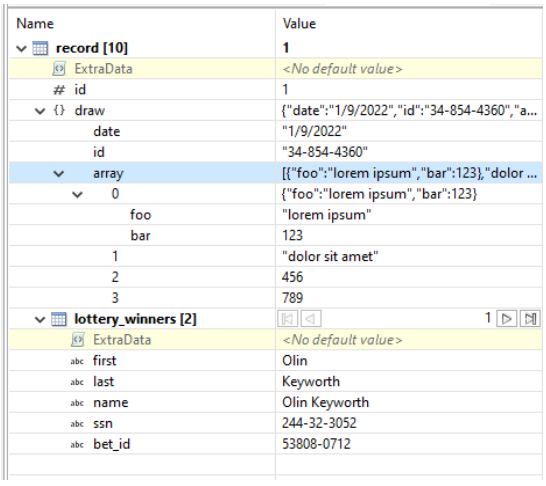
These provide an easy way to create a custom expression helper. It provides the same functionality as the Text Script wizard and allows you to concatenate data fields and optionally provide a prefix and suffix content. The name of the helper can be used as an expression in the content.

The following shows a classic Text Script approach to create an address block, in this case using the Handlebars Text Helper. When there are empty fields in the data, the respective line will be skipped. This prevents empty lines to show up in the text. In the example below this means their will be no empty line in case there is no company specified in the data.



Improved JSON support

- Improved native support for fields of type JSON in the Designer. (86609)
- JSON fields in the Data Model view can now be expanded and collapsed. (86700)



General Designer Improvements

- Added the "Color Output" option to Print sections menu. (85541)
- Add "**Properties...**" option to context menu for Remote Translation Files. (85655)
- Made it easier to select text at the beginning of an absolute positioned box. (86410)
- The attributes panel and the UI controls that allow changing the open resources are now disabled for remote resources. (83256)
- After changing a Package file, the default name suggestion for Template is now also used for the Datamapping component. (84817)
- Fixed an issue in the designer related to absolute positioned elements that have a transform style. (86053)
- Fixed an issue where dragging and dropping multiple data fields would only create one place holder and one script for the first selected field. (86182)
- Added support for email addresses with commas in the name.
  For example: "Test, User" <testuser.example.com> (86197)
- Fixed issue with speed degradation when loading and editing certain templates. (86321)
- Added missing context Help pages. (86406/86408)
- When using the COTG nested fields table feature in some situations clicking the Add Row button would not correctly copy the row. This has now been fixed. (86468)
- Re-enabled the range indicator in source editors. (86516)
- When setting the background color of a barcode to transparent ("none") it is no longer saved as translated string. (86766)
- Fixed minor drag-drop and double clicking issues. (86966)
- Fixed an issue with "*Each matched element*" scope that occurs when a script removes rows from a dynamic table or when a dynamic table uses data-hide-when-empty. (87107)
- **Send Email** and **Send Email Test** dialogs no longer default to mail.ca.objectiflune.com server as the mail server. (87223)
- Improved Tool Tip display. (87225)
- Upon first starting Designer the toolbar of the Data Model view was sometimes not visible when switching between perspectives. This has been fixed. (87435)
- Fixed a problem in the **Stylesheets Manager** related to the "border" shorthand. (87833)

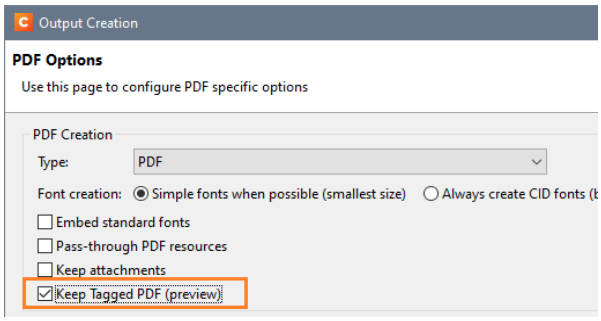
OL Connect 2022.2 DataMapper Improvements

- The **automation.properties.originalFileName** property is now set to the name of the current data file. If this changes to different data file, the property's value is automatically updated. (74921)
- Improved error messaging in ODBC driver for architecture mismatches. (80023)
- Fixed issue with **File->Properties** option not working if the DataMapper is open without any template. (82090)
- Added missing context Help links. (82469)
- Condition step: It was confusing when the Operator "is empty" (or "is true") was selected and the Operand types dropdown list was shown. To avoid this confusion the condition now has just Auto-Detect as on option. (82870)
- Improved logging. (83904)
- Fixed an issue where detail tables in groups were not properly supported in the **Dynamic Table Wizard**. (85752)
- Improved translations. (85958/85960)
- Updated Microsoft SQL JDBC driver which now uses encryption by default.
  The DataMapper was updated to retain backward compatibility for existing configuration files. (87795)

OL Connect 2022.2 Output Improvements

Tagged PDF

Add support for Tagged PDF files. (81757)



Note: This is a first cut and should be considered an experimental feature only, at this stage.

General Output Improvements

- A problem was fixed where CJK characters displayed using a TrueType font in a PDF got mixed up after a round-trip conversion to PostScript then PDF again. (84589)
- Fixed an issue with scaled PNG barcodes. (86698)
- Fixed a text positioning issue with fonts that falsely claimed to have a fixed width for all their glyphs (fixed pitch). (86924)
- The KIX database of the Dutch Postal Service PostNL has been updated to the 2022 version. (87131)
- Fixed an issue where a PNG barcode could be unscannable when the barcode div had padding or borders applied to it. (87557)

Print Output Improvements

- Fixed an issue with PDF/X-4 (and derived specifications such as PDF/VT) output where tiling patterns might retain incompatible device colorspaces. For example, a DeviceRGB in an output file with CMYK output intent. (69267)
- Increased the maximum number of copies in the print dialog and wizard to 100,000. (86327)
- Job preset sorting options have now been fixed so that when external sorting is active the content items are not pre-sorted by the standard sorting options first. This means that content items will be sent to the external sorting in the default process order. (85994)

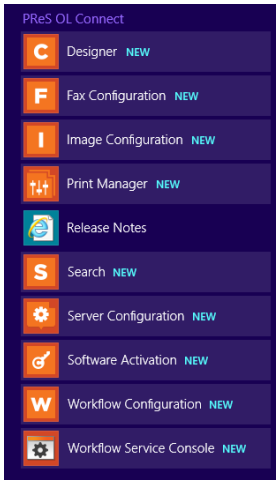
- **AFP and IPDS** improvements
    - Added a synthetic 'OS/2' table to TrueType 'OS/2' fonts that were missing them. (85082)

Workflow 2022.2 Improvements

Updated Installer

The Installer was updated to align with the Connect installer. (86682)
Changes include:

- It is now packaged as a single executable file.
- New Upland OL branding.
- Whether desktop icons are created or not is now a user-selectable option.
- Connect and Workflow now share the same Start Menu group.



Fixed issue with Workflow uninstallation, whereby the Update Manager/Messenger would be removed even if Connect remained installed. (80026)

SFTP and FTPS plugins

New SFTP and FTPS **input** and **output** plugins have been created. (84862)
As part of the SFTP plugin rewrite, an issue where a filename containing square bracket characters would cause the the plugin to go into an infinite loop was fixed. (87391)

SendGrid plugin

The SendGrid plugin is now able to set unsubscribe group information that SendGrid can use to populate and Unsubscribe link in emails. (85914)

Improved security - TLS 1.2 support

- The **Secure Email Input** and **Output** plugins now auto-detect the encryption level of the server to which they are connecting, up to TLS 1.2.
- The **HTTP Server** and **SMTP input** now have a configurable minimum encryption level, from SSL 2.3 to TLS 1.2, which is enforced when a client is connecting. (81387)

Retrieve records as simplified JSON

Introduced new output option for simplified JSON format in Execute Data Mapper and Retrieve Items plugin. (49330).

Security improvements

Updated Workflow 3rd party libraries to latest versions. (84387).

General Workflow Improvements

- **Office 365 OneDrive Input**: improved logging regarding found files. (76240)
- Fixed problem with binary data files in **Create Preview PDF** task. (82622)
- Custom plugins config parameters can now include non-ASCII characters, including paths retrieved from the Folder browser dialog. (83278/84603)
- The log entries for the **Set Properties** task now correctly refer to *Property Name* and *Property Value*. (83300)
- Fixed a memory leak in Custom plugins. (85458)
- Fixed an issue where parsing email addresses could fail, if the email contained multiple *To*, *CC* or *BCC* email addresses. This impacted upon Custom plugins such as **Mailjet** and **SendGrid**. (86164/86210)
- Add version number to **Mailjet** and **SendGrid** custom plugins. (86402/86404)
- Custom plugins now warn when an encoding problem prevents the configuration to be saved. (86655)
- **DocuWare Upload**: No error was being logged when uploading a file that was too large. This has been fixed. (87187)
- Fixed problem with CRLF in runtime parameters of **GoSub** plugin. (87699)
- Parameter list is cleared before each **GoSub** execution. (87711)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2022.1

License Update Required for Upgrade to OL Connect 2022.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version, then install Connect 2022.1.

For further details on how to upgrade the Update Client and update your Connect License see the Update Manager 1.5 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2022.1 and Workflow 2022.1, as well as some important installation information.

Installing OL Connect 2022.1 and Workflow 2022.1

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing (Installation Wizard) and licensing (Activating a License) OL Connect and Workflow can be found in the linked online Help pages.
- Note that OL Connect comes with a 30 day trial license by default.

Connect and Workflow installers require Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2022.1.5 Improvements

Scaled barcode issue

It was discovered that some OL Connect 2022.1.x scaled barcodes could be hard to scan. This affected "small" scaled (fixed to width) PNG barcodes in particular. This issue has now been fixed. (87162)

Workflow updated

An issue was found with how Workflow 2022.1.4 installed a Microsoft VC++ 2013 redistributable. This would cause subsequent OL Connect 2022.1.5 installation problems. This issue was fixed, and a new Workflow 2022.1.5 installer was prepared. (87190)

OL Connect 2022.1.4 Improvements

Windows 11 and Windows Server 2022 now supported

Both OL Connect and Workflow have now undergone thorough testing under the newest Microsoft Windows versions.

Having passed this testing process, OL Connect and Workflow version 2022.1.4 are now officially supported under **Windows 11** and **Windows Server 2022** operating systems.

Additional improvements made to Connect 2022.1.x installer.

- The language chosen for the installer is now applied to the installed products. (86545)
- The 'Select Setup Language' dialog language and the default language selection now use the local machine language. (86623)
- Fixed some German language translations, including one which that caused issues with the installation. (86317/86582)
- Fixed problems with non-ASCII text corrupting the Connect Service installation. (86436)
- Cleaned up text displayed on 'Installation Finished' page. (86237)

OL Connect 2022.1.3 Improvements

Add support for text extraction on CID fonts that rely on external Adobe "-UCS2" CMap files. (86069)

OL Connect 2022.1.2 Improvements

Additional security feature added, making Connect 2022.1.2 the first official Connect 2022.1.x release. (86250)

OL Connect 2022.1.1 Improvements

Some issues were discovered with the new Connect Installer during 2022.1.0 Release Candidate testing. These issues have now been fixed. (86124)

Version control added

Both the **Designer** and the **DataMapper** now use Git integration to maintain a history of files inside named **Projects**. (81457)

This feature allows you to combine all the files used in a Project and keep a record of any/all changes made to any of the files within the Project thereafter.

Projects are added via new **Project** menu.

The files comprising a Project can be viewed in the new **Project Files** panel.

Each time you *commit* a resource (i.e., when you save an updated version of it), any additional information provided gets recorded alongside the file.

You can browse through the history of all changes in the **Versioning History** panel, which contains a complete record of who did what, on which date, and for which reason.

You can elect to revert to a version from within the history if something turns out to be flawed in your current project.

Swiss QR barcode

Swiss QR barcode support added to Connect. (84012)

New Connect Installer

The OL Connect installer has been totally re-engineered. (81627)

The new installer is based upon more standard architecture and supports rollbacks in case something goes wrong during installation.

Maria DB installed with Connect

Fresh installations of Connect now install **MariaDB**, rather than MySQL. Pre-existing Connect installations that had the Connect MySQL component installed will be migrated to MariaDB at installation time.

Connect will continue to work with any customer provided MySQL databases that were present prior to Connect being installed. (83135)

> The migration process might take some considerable time (many minutes), dependent upon the size of the existing Connect MySQL database.

Java update

Updated the Java platform upon which Connect runs to AdoptOpenJDK 11.0.13+8 (82163)

Runtime parameters

The **DataMapper** now lets you specify runtime parameters directly inside condition and repeat steps so that you no longer have to write scripts.

We have also added Runtime parameters when calling **Workflow** sub-processes. The **Go sub** task now lets you specify which values to pass on to the sub-process without having to go through intermediate steps (by first storing them inside JobInfos, for instance).

This feature not only allows you to design your sub-processes faster, it also eliminates potential mistakes caused by improper values being passed to them.

Server Configuration

The Server configuration tool was not saving changes for Connect Server connection user ("olc-user" by default) correctly, if the new entry used the same password as the previous user. This could cause engines to stop talking to the Connect Server under certain circumstances. This has been fixed. (84937)

OL Connect 2022.1 Designer Improvements

Import Microsoft Word documents into print Templates

Added the ability to create new print Templates from Word documents and to import Word documents into an existing Templates. (83760)

The import uses the original Word document page size, margins, text formatting, lists, graphics, TextBoxes and even mail-merge information.

The contents of the Word file are added to the first Section in the Connect Template. Mail merge place holders in Word documents are used to create a datamodel and userscritps. Images are written to the Images folder and a base stylesheet is added to the Stylesheets panel. TextBox objects are converted to elements.

HTML templating using Handlebars

Provided new functionality that reduces the amount of scripting needed to create placeholder heavy and modular design templates. (83543)

If you've ever created OL Connect Designer templates for long documents like contracts and insurance policies, you're undoubtedly familiar with snippets. Snippet-heavy templates can become hard to maintain when each snippet requires multiple scripts to apply personalization and conditions. It requires the snippets and scripts to be thoughtfully organized (grouping, naming conventions, search optimizations). But even then, things can become cluttered and may even impact the overall performance of your template.

Users with a background in web design may be familiar with the Handlebars library. It is a popular templating library that provides a simple way to merge HTML snippets with data. This library has now been added to OL Connect Designer.

The Handlebars template snippets in OL Connect Designer will vastly reduce the number of scripts for the mentioned document types and help keep templates lean by creating semantic snippets.

Improved Web Font support

The OL Connect Designer already supported web fonts, but only via CSS Stylesheets. To simplify this, an option has been added to the Fonts folder to directly add remote fonts. (78790)

Conditional Script improvements

To further simplify Conditional Scripts, we have added two new conditions `is empty` and `is not empty`. (81585).

Search across Resources

The **Resources** panel has been expanded with a filter option. (83845)

This allows searching across master pages, sections, stylesheets and snippets.
The search results are grouped into the files containing the search term showing a preview and indicator of the term in each file.

Sample Projects

- The *Project Wizards* feature has been renamed *Sample projects* to reduce potential confusion with the newly introduced Version Control **Project** concept. (84804)
- Sample project Wizards now include a credentials page in which you can set the username & password for the Workflow configuration. (84338)

General Designer Improvements

- Fixed a problem with HTML snippets saved in source mode not actually saving the changes. (80971)
- Refresh types button in text scripts now enabled when needed. (82014)
- Templates are now still considered "changed" after making changes, saving and then undoing those changes. (83254)
- **Send to Server** dialog made wider. (83636)
- **Dynamic Tables**: The `data-show-row` option for TR elements in `THEAD` and `TFOOT` can now be set to *start-of-table* to have the row appear only in the first (or only) instance of the TABLE. (84135)
- Connect will now continue to recognize remote translation files after a template is saved, closed and then reopened. (84825)
- Updated 3rd party library to fix issues with **USPS IMb** barcodes. (84847)

OL Connect 2022.1 DataMapper Improvements

JSON Field Type added

Added a JSON field type that allows storing JSON objects in the data model. (83707)

The DataMapper and the Designer automatically parse the contents of these fields and allow you to address each individual property as if it were a field of its own. The Script Editors in both modules even offer code completion as you type.

Runtime Parameters added

Runtime parameters are now directly available for some steps. (82079)

The **Condition** step and **Repeat** step *Based on* property, and the **Extract** step *Mode property* now have Runtime parameter as an available option.

MariaDB support

The Connect DataMapper now supports MariaDB data sources. (83314)

General DataMapper Improvements

- The Ignore CR/LF at end of file option now also Ignores trailing Form Feed characters (FF) in data files. (67112)
- Added a new method named `isRuntime()`, to determine whether the data mapping process is currently running in debug or runtime mode. (80858)
- Fixed a problem with runtime parameter initialization. (SHARED81747-)
- Datamapper now supports JSON data using UTF-16 and UTF-32 encoding. (83886)
- Fixed Automatic Date parser for dates with negative time zone with non-zero minutes. (83528)
- Incorrect code completion suggestions for Arrays in JavaScript have been fixed. (83719)
- Fixed boundaries behaviour in TEXT emulation when the "Ignore CR/LF at end of file" option was set. (83980)

OL Connect 2022.1 Output Improvements

Tweak Postscript Headers

Raw Postscript can now be inserted at the header level. (83380)
Certain printers require very specific information in their header comments to work correctly. Although this could already be handled through custom Printer Definitions, it can now also be done directly through Dynamic PPD mode.

The inserts are added in the order they appear.

Use system fonts as fallback for missing input file fonts

When fonts are missing in input files (PDF, PostScript, etc.), it is now possible to have Connect use system fonts if the same fonts are installed locally. (54777)

At the moment this a technical preview feature, which requires manually editing the OL-outputpreset file, thus:

```
<fontSubstitutionConfig>
<pluginId>0</pluginId>
<useSystemFonts>true</useSystemFonts>
</fontSubstitutionConfig>
```

General Output Improvements

- Improved error/warning messaging. (75666)
- When the output engine flattening a transparency in a PDF file takes more than a minute, it will now logs messages to report progress on the flattening operation. (81683)
- It is now possible to change the default SMTP timeouts of the Merge Engine via setting system properties in the ini file. (82202)
- Added support for web font files (*.woff) in AddText components of custom Enhance workflows. (83567\83587)
- Prevented an "Attempt to paginate an already paginated DOM" error that could occur when scripts called merge.section.paginate() function. (84276)
- Improved handling of Open Type Format (OTF) fonts. (84502)
- Cloning a section that contained a dynamic table would break the expansion/pagination of the table. This has now been fixed. (84891)
- Updated 3rd party library to fix issues with **USPS IMb** barcodes in output. (85117)

Email Output Improvements

- Email content now always created through OL Connect Server. (28569)

Print Output Improvements

- Fixed issue in Dynamic PPD to include missing %EndPageSetup. (83388)
- "*Timeout errors*" could sometimes be encountered when connecting to a remote printer from Connect server. The timeout values have now been optimized to fix this issue. (83532)
- Fixed an error occurring in PDF output for certain types of images when image optimization is enabled (Indexed with a base colorspace that is not DeviceRGB/Gray/CMYK). (83559)
- Media items in DynamicPPD Output presets are now optional. (83827)
- Crop marks were incorrectly positioned on the back side of duplex output sheets when using 90 or 270 degrees rotated impositioning. This has been fixed. (84770)
- Fixed issues encountered in templates that use JQuery for setting up dynamic tables. (84814)
- Improved handling of PDFs with Transparency Groups when using Imposition. (84914)
- **AFD and IPDS** improvements
    - IPDS jobs that contained many barcodes were failing to print. The handling of barcodes has been changed to address this issue. (83927)
    - IPDS and AFPDS advanced logging now contains extra information on forms and images used. (84202)
    - Updated handling of PDF with Transparency Groups when using Imposition in IPDS output. This has resulted in improved processing times and smaller spool-file output. (84914)
    - To address an issue with IPDS Duplex printing starting out of sync after a restart, we now include Plex commands at the start of each physical sheet. (84916)

OL Connect 2022.1 Print Manager Improvements

- A problem was encountered on some IPDS printers, which would freeze after a period of time. This was due to issues with reading responses from the printer. A new "*IPDS streamed Reader*" preference has been added to the **Print Manager Preferences**, to cater for printers that experience this issue. (84485)

- Extra information now captured and displayed in the PrintManager. The extra information is available in the **History** and **Preview** panes. (85054)

Workflow 2022.1 Improvements

Runtime variables added to Go Sub

Introduced an easy way to use runtime variables in sub-processes of Workflow. (79248)

Legacy Plugins

A new plugin category has added for legacy plugins. These are plugins that are deprecated, and thus should not really be used in contemporary Workflow projects. (84674)

General Workflow Improvements

- A *CaptureOnTheGo* plugin issue was fixed where defining a cover page with some specific PNG files caused a warning dialog to appear when debugging the process, causing the Workflow service to hang. (59456)
- Improved handling of long paths to templates in the Merge Engine. (79100)
- Fixed issue with repeated credentials request in **Create Web Content** plugin when NodeJS server input is used in process. (79390)
- Enhanced security added for all **Microsoft365** plugins utilizing Microsoft Graph. (80932)
- Improved support for updated Connect Template or DataMapping runtime parameters in Workflow plugins. (81282/83144)
- Plugin dialog names changed to just "Properties" throughout Workflow. (84669)
- Fixed catastrophic failure that could sometime happen in **Create Job** plugin. (81011)
- Protocol mismatch error has been made clearer, when detected upon Connect server connection. (81434)
- Fixed an issue with Telnet Helper Service not receiving files properly and keeping client connections open indefinitely. (83088)
- The **FPT Input** and **FTP Output** plugins could modify the structure (line endings) of PDF files on some FTP systems. This issue has been fixed by ensuring file transfer is now always set to binary. (83900)
- Fixed errors that could sometimes occur when sending email messages with PDF attachments through the **SendGrid** and **Mailjet** plugins. (84312)
- An issue was fixed where installing a custom plugin on a brand new install of Workflow could fail with an error message saying that a "module cannot be found". (84491)

- Local variable values were incorrectly assigned to any global variable of the same name. This has been fixed. (84521)
- Fixed an issue where template names starting with a number could cause an error in the **Render Email Content** plugin. (84619)
- Memory leak fixed in the **DocuWare** plugins. (84644)
- Fixed issues with **OL Connect Send** solutions containing printed files names with a single quote (') character in the file name. (84940)
- Memory leak fixed in the **PDF/A-3 Attachment** plugin. (85011)

OL Connect 2022.1 Improvements

Security and Installer improvements

- Connect dependencies have been updated. (83698)
- The Adobe PDF Library and PDF Converter libraries were updated to their latest version. (83712)
- Improved resilience of Server connection. The Maximum queues setting has been removed from the **Server Configuration** preferences. The value is now automatically optimized, based on the maximum threads setting. (84029)
- The OL Update Manager included as part of the Connect/Workflow has been updated to version 1.5 in Connect/Workflow 2022.1. (85508)
- Updated Windows Service Wrapper library to improve security. (85076)
- NodeJS log lib (log4js) updated to improve security (85078)
- Upgraded dependencies to protect against Spring vulnerability CVE-2022-22965. (85809)

REST API Improvements

- A new header named `Content-Disposition` has been added to the response of the FileStore download REST endpoints to return the original file name. (84592)
- The content creation for email working examples in the Connect REST Cookbook can now correctly connect to a mail server that does not require authentication. (83861)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2021.2

License Update Required for Upgrade to OL Connect 2021.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up

until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version, then install Connect 2025.1.

For further details on how to upgrade the Update Client and update your Connect License see the Update Manager 1.3 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2021.2 and Workflow 2021.2, as well as some important installation information.

Installing OL Connect 2021.2 and Workflow 2021.2

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that OL Connect comes with a 30 day trial license by default.

Connect and Workflow installation requires Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2021.2.1 Improvements

Fixed Fatal IPDS errors

IPDS jobs that contained many barcodes were failing to print. The handling of barcodes has been changed to address this issue. (SHARED-83927)

Fixed LPR printing errors

"*Timeout errors*" could sometimes be encountered when connecting to a remote printer from Connect server. The timeout values have now been optimized to fix this issue. (SHARED-83532)

OL Connect 2021.2 Designer Improvements

Sheet Configuration Repeats added

Sheet configuration can now be repeated after a fixed number of sheets.

For example when a script adds a number of paragraphs from a snippet based on the record index, the number of pages and sheets will increase when browsing through the records. (SHARED-82191)

Retrieve a list of image resources

Added an option to retrieve a list of image resources via user/control scripting. This caters for scenarios such as validating whether a certain image exists in a template before setting the `src` attribute of an element in script. (SHARED-82303)

Improved handling of read-only Templates

Improved the handling on read-only OL-template files. (SHARED-82177)
Improvements include:

- Added a visual cue in Designer to indicate a Template is read-only.
- Saving changes in read-only files now launches the *Save as* dialog.

Retrieve the position information of elements

Introduced new script option `results.position()`, which can be used to retrieve the position (X and Y offset) of matched elements. This allows you to determine locations on the page and dynamically perform actions based upon the positions. (SHARED-81098)

Improved Drag and Drop support in data model fields

It's now possible to drag scripts with simple class and/or ID selectors to the template, similar to how this was already possible for scripts with text based selectors.

Dragging a data field from the data model to the scripts folder will create a script with a class based selector.

Holding down the ALT-key while dropping the data field will create a scripts with a text based selector (similar to the behaviour when dragging a data field into the template)

Also fixed a bug where no table was inserted when the dynamic table wizard was opened after ctrl-dragging a detail data field into the template. (SHARED-82152)

General Designer Improvements

- Improved translations throughout. (SHARED-81664)
- A toolbar button has been added to the **Translations View** to open the **Edit Locale Dialog** directly. (SHARED-81316)
- When navigating through records in the Data Model view, the first click of an arrow was ignored. This has now been fixed. (SHARED-79830)
- You can now add a custom Template property by clicking the first empty line in the Custom properties dialog (File > Properties ... > Custom). (SHARED-80775)
- **Packaging** a Template no longer throws an error when a section that a thumbnail is created for is disabled by script. The packaged Template will now use the disabled section. (SHARED-81237)
- Fixed an issue with the formatting of default values of date fields, which caused errors when the Template was re-opened. (SHARED-81721)

- Fixed issues with URIs containing HTML encoded backslashes in the file path. (SHARED-81887)
- Prevented an error after saving a resource while a section with shared content is active. (SHARED-82200)
- The execution scope of a script folder now properly inherits the execution scope of its parent folder. (SHARED-82026)
- Fixed errors that could be encountered when previewing a template from Designer (using the Preview HTML button) or when using a web page served by Workflow or Node-RED. (SHARED-82532)
- The Section properties dialog and the scripting API now both allow the width and height of a background image to be scaled independently. (SHARED-82616)
- When a data field is dragged to the email fields in an email section, the dragged field will now be used in the script generated. (SHARED-82876)
- Improved the way translation files are matched to the active locale. (SHARED-83079)
- The default insertion method for new text scripts has reverted to HTML, rather than text. (SHARED-83412)

OL Connect 2021.2 DataMapper Improvements

Improved custom SQL dialog

The interface for configure SQL queries has been improved for longer SQL queries. The interface now adapts based upon the display size and the length of the SQL query. (SHARED-74633)

Simplified Boolean Conditional tests

A new "*Is True*" operator has been added to for Condition steps. This allows a condition to check a boolean value directly without having to first supply a second operand. (SHARED-79342)

Determine the number of lines in the current record

A new `steps.lines` property has been added. This property contains the number of lines in each record for Text-mode data mapping configurations, which allows for much easier looping through all the lines in a record. (SHARED-78996)

Improved Timestamp processing

A number of improvements in the DataMapper management of Date/Time fields have been implemented. These allow for improved handling of time zones and of time stamps without

time zone information. (SHARED-77648)

Extract CSV data by column index value

Introduced a new java script function `data.fieldExistsByIndex()` to facilitate extraction based on index. This allows the DataMapper to extract CSV fields by column index instead of column names. This can be particularly useful for translated CSVs whose column headers change with each language. (SHARED-80020)

General DataMapper Improvements

- Improved translations. (SHARED-81664)
- Introduced new option to autorotate PDF data files. (SHARED-54915)
- Fixed an issue with cancelling *Refresh boundaries* operations. (SHARED-76112)
- **Merging PDF pages** using the AlambicEdit API in a script or through a Custom Plugin could lead to content being lost. This issue has now been resolved. (SHARED-76622)
- Fixed handling of radio button focus when clicking in the XML DataMapper input data panel. (SHARED-76855)
- *Runtime Parameters* have now been added to the PDF version of the **Export Report**. (SHARED-79186)
- Improved **Repeat step** comparisons on numeric data (floating point) entries. (SHARED-79825)
- When navigating through records in the Data Model view, the first click of an arrow would be ignored. This has now been fixed. (SHARED-79830)
- Fixed an issue with preprocessing scripts and AFP metadata. (SHARED-81723)
- Runtime parameters were not accessible in the Properties initialization script. This has been fixed. (SHARED-82005)
- Altered runtime parameter default values for better backwards compatibility. (SHARED-82534)

OL Connect 2021.2 Output Improvements

IPDS duplex 2-Up printing improvements

- Fixed a page size issue in Content generated by DataMapper. (SHARED-81960)
- Added support for selecting Twin System IPDS printers in **Print Manager**. (SHARED-82088)

- Added a **Print Manager** option to specify how long to keep a session open, once the last print job finishes.
- A new **Print Manager** option has been added to allow the printer to push all committed pages to the stack.

Improved password protected PDF output

- Fixed an issue with PDF output unnecessarily prompting for a password when viewed in some Android and iOS browsers and apps. (SHARED-82931)
- Fixed an issue that could lead to blank pages on some Acrobat Readers (iOS, Android and Chrome Extension). (SHARED-83034)

General Output Improvements

- Fixed and issue with PCL input conversion. (SHARED-81616)
- Include the *file currently open for flattening* in debug log messages. (SHARED-81685)
- Fixed an issue where detail table data from a previous run/record showing up in the next run/record, when a scripted table was included that did not use the standard table expander. (SHARED-82466)
- Pagination errors could occur when large scripted tables did not use the standard table expander and were split across pages. These errors have been fixed. (SHARED-82198)
- Issues with the ascender height calculation of **IMB barcode** marks have been fixed and IMB barcodes now meet USPS specifications. (SHARED-80812/83082)
- The output engine now remembers files it has previously run through the transparency flattener, reusing the flattened file if possible, improving output speed. (SHARED-80935)

Email Output Improvements

- Fixed an issue whereby all web sections were attached to email instead of only the default web section. (SHARED-82986)
- Fixed an issue where elements with inline style `display:none` would be included in email content when the CSS inlining was not set to "*Apply CSS properties on elements*". (SHARED-82788)

Print Output Improvements

- Added support for null comparisons in **Finishing Rules**.
  Two new data selection filter operators allow for detecting null data/param values in

data rules and null/missing properties in *property rules*. The new operators are "*is set*" and "*is not set*". (SHARED-72938)

- The "*Job Size/Page Count*" and "*Speed/Overall Speed*" fields in the print summary in the "Connect Designer" logs are now being correctly reported. (SHARED-81385)
- A problem was fixed whereby image data from PNG files merged into a PDF were left uncompressed, increasing the file size. (SHARED-81804)
- Fixed issues with output presets edited in languages other than English. (SHARED-81843)
- Dynamic PPD printerdef and output presets were set to %%Title: ${template}. The %Title field now uses the jobName when using LPR. (SHARED-81958)
- Fixed pages size issue with DataMapper auto-generated Content Creation. (SHARED-81960)
- Improved Output Creation error messaging if the *impositioned pages won't fit on the sheet*. (SHARED-81962)
- Allow up to a 1pt deficit in media size when checking the required size for **impositioning**. This reduces errors if some lengths were specified in different units. (SHARED-82413)
- If the Binding type in a Dynamic PPD was set to Default, this would result in NULL errors when running jobs. This has been fixed. (SHARED-83310)
- **AFP and IPDS** improvements
    - When creating AFP/IPDS output with TrueType fonts, Connect uses Unicode values to refer to the glyphs. This would only work if there was a one-to-one mapping from Unicode to all the glyphs being used.
      Connect now fallbacks to using a bitmap font when unknown Glyphs or duplicate codes are encountered. (SHARED-80954)
    - Improved vector conversion in AFP and IPDS. (SHARED-82246)
    - Report errors and stop the job, when an AFP output rip fails. (SHARED-82814)
    - **TLE metadata** in AFP output now allows selection of character encoding type. (SHARED-82356)
    - When running an Enhance configuration in Connect, TLEs read from an AFP input file were duplicated in an AFP output file. This issue has been fixed. (SHARED-82619)

Workflow 2021.2 Improvements

Performance improvement to Retrieve item plugin

A new new *Optimized* checkbox option has been added to the **Retrieve Items** plugin. When selected, the plugin will use an improved and more efficient version of the REST API.

The checkbox can only be enabled for *Entity Records* and *Record Set*. Other Entity types retain the old functionality. (SHARED-74403)

GUI improvements

- Workflow interface descriptions made more descriptive and user friendly. (SHARED-49584)
- Improved translations. (SHARED-81662)

General Workflow Improvements

- The Java Runtime Environment (JRE) used for the **Connect Send** plugin and the back-end database has been updated to more current JRE (1.8.0-292). (SHARED-81937)
- Fixed errors in **Create Output** that could sometimes occur when creating large output files. (SHARED-67092)
- Improved **HTTP Server Input** performance. (SHARED-72877)
- Added support for duplicate key names in **NodeJS Server Input**. (SHARED-74512)
- Fixed a rare but insidious memory corruption (stack overflow) problem. (SHARED-80189)
- **Retrieve Items** "out of memory" errors fixed. (SHARED-80840)
- Fixed issues with using an Apple iDevice to browse to a site hosted by the **NodeJS Server** Helper. (SHARED-81658)
- Fixed an issue with the **Send Email** plugin, whereby the HTML email body would be generated as an attachment to the email rather than showing as the actual email body. (SHARED-82252)
- An issue was fixed with TCP ports leaking if connection attempts from either Messenger or the Workflow LPR client were unsuccessful. (SHARED-82868)
- **Create Print Content** "out of memory" errors fixed. (SHARED-82485)
- Fixed a **HTTP Server Input** issue whereby a static resource type would not be interpreted properly. (SHARED-82457)

- Validation using runtime parameters are now properly processed in **Data Mapping** plugin. (SHARED-82315)
- Fixed issues with German Umlauts in email message text, when using the **Send Email** plugin. (SHARED-83470)

OL Connect 2021.2 Improvements

Security improvements

- Java Runtime Environment (JRE) used for Connect Send plugin and back-end database updated to current JRE 1.8.0-292. (SHARED-81937)
- Multi-line log messages converted to a single line, to preclude *CRLF Injection* exploits. (SHARED-74718)

Print Manager Improvements

- Improved translations throughout. (SHARED-81664)
- **Twin IPDS printer configurations** are now better supported. (SHARED-81955)
- Added warnings to the *Messages* pane when a spool file is being held by another process and cannot be processed. (SHARED-82885)

General OL Connect improvements

- Improved backend database startup process across Connect applications, reducing issues with startup failures in containerized environments. (SHARED-80360)

REST API Improvements

- **HTML content creation: leave remote resource links intact when inlining**. Introduced a new inline LOCAL mode for embedding local resources in HTML content creation endpoints. This allows you to specify that remote resources remain external whilst inline resources are local. A resource is considered local if it is embedded in the template or if it has a URL with the file: protocol. In other words, a resource is not considered local if it has a URL with the HTTP: or HTTPS: protocol. (SHARED-82592)
- **Render Email Content** now checks email addresses. This means that when an email address is invalid, no email content will be created and an error is reported for the record with an invalid email address. (SHARED-81313)
- The endpoints for **Email content creation** now return status code 400 (bad request) instead of 500 (internal server error) if the template does not have a email context or if the requested section was not found. (SHARED-81218)

- Fixed an encoding issue when the /entity/datarecords/values REST API endpoint was called with the optimized parameter set to true. (SHARED-82833)
- Generate a 400 (Bad Request) error instead of a 500 error (Internal Server Error) if an invalid inline mode is passed to one of the HTML content creation REST endpoints. (SHARED-82899)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2021.1

License Update Required for Upgrade to OL Connect 2021.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version, then install Connect 2021.1.

For further details on how to upgrade the Update Client and update your Connect License see the Update Manager 1.3 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2021.1 and Workflow 2021.1, as well as some important installation information.

Installing OL Connect 2021.1 and Workflow 2021.1

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that OL Connect comes with a 30 day trial license by default.

Connect and Workflow installation requires Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2021.1 Enhancements

OL Connect 2021.1.1

In the OL Connect 2021.1.0 partner preview it was discovered that dragging group data from the DataMapper into a template was sometimes throwing an error. This has been fixed in

the 2021.1.1 patch release of Connect. (SHARED-82134)

Security Improvements

- For increased security the cookie, localstorage and indexedDB features have now been disabled for the internal browser used by the Designer and Merge Engines. (SHARED-77792)
- The **Enter credentials** dialog has been improved. Not only does it now support selecting a Server on another machine through URL selection, but the connection can now be tested from within the dialog box. (SHARED-80112)

HTTPS support

As part of the ongoing effort to improve security, OL Connect Server now has the option to use and/or accept HTTPS connections from clients such OL Connect Workflow and Designer. (SHARED-78230)

Added option to define a REST Server port that is HTTPS secured. (SHARED-74997)

Print Manager Improvements

For IPDS printers, **Multi Job mode** is now set as the default. To disable Multi Job mode, select *Legacy mode* option. (SHARED-78768)

Installer Improvements

- Both Connect and Workflow can now be installed when no internet access is available. (SHARED-79964)
- The issue with slow responsiveness in Installer component selection has been fixed. (SHARED-76902)
- On a Connect update/upgrade, the installer Welcome page now clearly displays both the currently installed version and the upgrade (update) version about to be installed. (SHARED-77819)
- You can now choose to omit the **Update Manager** from the installation, using the command line parameter `UPDATEMANAGER=0`. (SHARED-79761)
- Fixed an uninstallation issue caused by some Connect temporary files being set as read-only. (SHARED-79765/79838)
- Fixed an installer issue that was preventing Connect from installing on Hyper-V (and some other virtual environments other than VMware). (SHARED-79918)

Server Improvements

- Server extension no longer starts up a second set of server webservices. (SHARED-78669)
- Fixed some index out of bounds exceptions that were appearing in the logs from time to time when engines are restarted. (SHARED-79522)
- When the "Enable server security" checkbox preference is changed and applied a dialog now appears warning that a restart of the Server is required. (SHARED-80550)
- Server and Designer products that are set to use internal engines (not separate processes) will now no longer terminate abruptly if the security preferences for web socket authentication are not initialized in time. (SHARED-80973)

REST API Improvements

- The `contentcreation/{templateId}/{dataSetId}` REST endpoint now accepts runtime parameters. (SHARED-79854)
- On successful authentication/login, the `Authenticate/Login to Server` REST API end-points now return a "*Token-Expires-In*" response header with a value set to the expiry duration of the authorization token in seconds. (SHARED-80138)
- The standard 'JSON Error' response from the Connect Server will now always return a 'Content-Type' header with a value of "application/json". (SHARED-78633)
- Deleting managed files from the filestore via the REST API now has improved error handling to cater for files that are still required by OL Connect. (SHARED-79353)
- Submitting a "blank" password value when using basic authentication now returns a "401 Unauthorized" response rather than a "500 Internal Server Error" response. (SHARED-80641)
- The basic authentication error "No account found for" will no longer be shown in the server log when server security is disabled. (SHARED-80683)
- The basic authentication error "Bad credentials for: <username>" will no longer be shown in the server log when server security is disabled. (SHARED-80685)

OL Connect 2021.1 Designer Improvements

Conditional Wizards

The Designer conditional Wizard for elements and the Wizard for conditional print sections have been dramatically improved, allowing very complex rules to be created without any need for scripting. (SHARED-79806)

COTG Nested Field Tables

COTG Fields Tables now support nested (fields) tables. (SHARED-77333)

Send to Server

The updated **Send to Server** now supports uploading configurations and templates directly to web applications (via arbitrary URL), as well as to Connect or COTG Servers.

The web server settings first need to be added via the Preferences dialog, and they will then become available as an option in the updated **Send to Server** dialog. (SHARED-80112)

Scripting Improvements

- Added support for selecting runtime parameters in text scripts. (SHARED-78793)
- Added support for creating conditional rules based on runtime parameters. (SHARED-78799)
- Expand dynamic tables fetched with `Loadhtml()`. (SHARED-80263)
- The default insert method for new search-and-replace scripts is now "Text" instead of "Html", to improve performance. (SHARED-80376)
- Fixed syntax highlighting issue with the Script keyword '*let*' . (SHARED-78644)
- We now generate an error if a script tries to move an element relative to itself (which Connect does not support). (SHARED-79945)
- Fixed an issue where an incorrect selector would be generated for scripts when a datafield containing spaces in the name is dragged from the data model to the template. (SHARED-80535)
- Fixed a problem with UNC paths in the Dynamic Image Script wizard. (SHARED-81145)

General Designer Improvements

- Improved Designer translations/localization. (SHARED-76798)
- **JSON editors** and the `loadjson()` function now support UTF-16 and UTF-32, in addition to UTF-8. (SHARED-74159)
- After adding an **HTML snippet** you are now prompted to select encoding (UTF-8 by default). (SHARED-78552)
- Runtime parameters can now be used to set template **Locale**. (SHARED-78787)
- Added validation to **JSON resource** editors. (SHARED-79937)

- New prompt added when dragging a field from a detail table to a location outside a detail table. (SHARED-80116)
- Add support for copying contents of an extraction field into the clipboard. (SHARED-80164)
- Can now set the Virtual Stationery of a media back to "None" by using the clear button. (SHARED-78802)
- Fixed an issue within the Source tab, whereby HTML in the source tab unintentionally contained helper elements for resizing and dragging. (SHARED-78928)
- Fixed an issue with converting a paragraph to a header1 (p to h1), in which the parameter ID would be lost. (SHARED-79022)
- Fixed an issue where the cellpadding and cellspacing values that were set directly on a table were removed after switching to Preview, if these values were not defined in stylesheets. (SHARED-79543)
- Fixed issue with the popup menu in the breadcrumbs bar. (SHARED-79727)
- Show Edges settings once again persist through view changes. (SHARED-79970)
- Fixed a problem with absolute positioned elements nested inside other absolute positioned elements. (SHARED-80003)
- Printing via the server (not proof printing) from the Designer now properly passes the runtime parameter values set in the Designer. (SHARED-81362)

OL Connect 2021.1 DataMapper Improvements

Native JSON support

The DataMapper now supports JSON as a native data type, with no conversion required. The power of JsonPath selectors helps you streamline the data extraction process, allowing you to handle huge record sizes effortlessly. This makes designing web-based solutions much more straightforward. (SHARED-78228)

Streamline repetitive actions

Added the ability to break out of a loop and move onto the next step when the desired condition has been met. (SHARED-79527)

We have also made the Goto step optional inside Repeat loops. This allows you to loop through, for instance, JSON structures without having to move the data pointer, making the operation less error-prone and more efficient. (SHARED-79524)

General DataMapper Improvements

- Improved DataMapper translations/localization. (SHARED-76798)
- Speed improvements made for processing large transactional text data files. (SHARED-80006)
- Added helper function in JavaScript to update field values in records or to add new rows in detail tables. (SHARED-34191)
- Add support for copying contents of an extraction field into the clipboard. (SHARED-80164)
- You can now set different values in each copy of a record (specified through `record.copies`) to distinguish whether a record is the original or a copy. (SHARED-80372)
- Connect now tracks how many records (along with the data type) are processed for each data mapping. (SHARED-80455/80463)
- Certain AFP files containing a data matrix were being rendered incorrectly. This has been fixed. (SHARED-81406)

OL Connect 2021.1 Output Improvements

Dynamic passwords for protected PDFs

The password for opening/protecting output PDF files can now be set dynamically. (SHARED-80537)

General Output Improvements

- The stability of the Output engine has been improved. Particularly when the engine is running with low memory. (SHARED-77203)
- The output engine is now more accepting of certain invalid CFF fonts (those containing hintmask without hstemvm/vstemvm). (SHARED-76464)
- Fixed an error in the output of Unicode (3,1) cmap tables for TrueType fonts which could trigger an exception if multiple glyphs have the same Unicode mapping. (SHARED-78284)
- Fixed issues with Type 1 and CFF font glyphs. (SHARED-79973)
- Versioning has been added to font mapping files for TextRestore. To prevent problems in text extraction going unnoticed, font mapping files without a version or with a mismatching version will result in an error. Resolving those errors requires manual updating of these font mapping files. (SHARED-80568)
  This change only applies to Enhance configurations run from Connect.

Print Output Improvements

- Upgraded the Datalogics library (APDFL). (SHARED-79251)
- Fixed an issue with misleading warning messages for certain output presets when opening the production Print Wizard or Output Preset editor. (SHARED-74399)
- **Dynamic PPD rule validation** is now immediately updated after the additional elements for a selected PPD element are updated. (SHARED-77060)
- Restored **maximum stack depth** setting in the Imposition options page back to a maximum of 100,000. (SHARED-79357)
- Increased **maximum paper sizes** available for PostScript output. (SHARED-79433)
- The output engine now can process intermediate files (from merge engine) that are over 2 GBs in size. (SHARED-80490)
- For some output formats, such as PostScript, output creation didn't terminate for certain input files containing transparency, because Connect's transparency flattening tool was unable to process it. This has been fixed. (SHARED-81542)

Workflow 2021.1 Improvements

Consolidate print processes

The name of the LPD queue that receives a job is now stored in the **LPD Input** task as `JobInfo 6`, allowing you to build a single process that can handle jobs coming from

multiple queues. This feature will help you cut down on the number of distinct processes required to handle all print queues.

Bypass Content creation

The **Execute Data Mapping** task can now *Bypass Content creation* to use an existing PDF as content, without requiring a template. (SHARED-80168)

General Workflow Improvements

- Improved Workflow translations/localization. (SHARED-76783)
- The **Fax Service** plugin now works with PDF/VT files created by Connect as welll as legacy PlanetPress Suite .PTK forms.(SHARED-67742)
- Script reference to *"https://code.jquery.com/jquery-3.3.1.min.js"* removed from NodeJS Server login page to avoid problems on secured networks without internet access. (SHARED-79298)
- The obsolete *"Run from Desktop"* option has been removed. (SHARED-79849)
- An option (unchecked by default) was added to the **FTP Input** plugin to grab empty files found on the FTP server. (SHARED-76175)
- Fixed periodic **NodeJS Server** connection issue when using Active Directory authentication. (SHARED-77265)
- **Messenger Service** now tries to identify issues with MS Access drivers that could prevent the service from starting. (SHARED-78450)
- Improved the SOAP timeout settings, so any value is now accepted. (SHARED-78646)
- Fixed paths issue with **Send Image to Printer** plugin. (SHARED-78705)
- Pasting attributes of **Secure Email Output** would result in plugin configuration display to forego the CRLF characters in the email body at configuration time, although it did not alter the actual plugin output. This has been fixed. (SHARED-79627)
- Fixed issue with the **SendGrid** plugin not setting the *cc* and *bcc* fields. (SHARED-79684)
- **NodeJS Server** now correctly handles requests having a Content-Type of application/json which also specify a charset encoding. (SHARED-79871)
- Improved **NodeJS Server** handling of custom response codes. (SHARED-79960)
    - Invalid HTTP Response codes are now filtered and handled properly, returning error 565 to the client.

    - A NodeJS input with the "Loop through attachments" option selected that does not receive an attached file now advises no file was found in the log and its output can be properly processed by secondary input plugins.
- Once the connection to a server is established, **Secure Email Output** plugin no longer waits indefinitely for a response and will instead raise an error after the timeout delay set by in the plugin configuration. (SHARED-79987)
- Some plugins could not be selected as default input/output in the Workflow preferences. This has been fixed. (SHARED-80014)
- If the "Use Dynamic SQL" option is used in the **Database Query** plugin, then any value in the non-dynamic ODBC location string is ignored for plugin configuration validation. (SHARED-80700)
- Improved Connect plugin performance when using server authentication. (SHARED-80834)
- Fixed issue with the **NodeJS Server Input** task not processing jobs if the action name was a superset of another action name declared previously. (SHARED-81129)
- HTTP Posts actions using the Content-Type application/x-www-form-urlencoded did not process unencoded % characters properly in an url encoded data stream. This has been fixed. (SHARED-81174)

OL Connect Send Improvements

Citrix and Microsoft Terminal Services (Remote Desktop Services) support

Version 1.8 of OLCS adds support for both Citrix and Microsoft's Remote Desktop Services environments. (SHARED-71164/71167)

Some of the changes made to allow this include:

- The license check for **user based licenses** has been changed to distinguish between different users on the same machine/system. These are now separately counted. (SHARED-80631)
- Improvements were made so you can now get the browser pop-up when printing from a desktop on a multi-user system, for a full interactive OLCS experience. (SHARED-80443)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL ConnectRelease Notes 2020.2

License Update Required for Upgrade to Connect 2020.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading to 2020.2 from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Manager, as newer versions of the Update Manager can update your OL License to the required version, then install Connect 2020.2.

For further details on how to upgrade the Update Client and update your Connect License see the Update Manager 1.3 - Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2020.2 and Workflow 2020.2, as well as some important installation information.

Installing OL Connect 2020.2 and Workflow 2020.2

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Connect and Workflow installation requires Administrator rights

Please note that the OL Connect and Workflow installations can *only be run by users who have Administrator rights*.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2020.2.1 Enhancements

Fixed Connect 2020.2 Installation error

An installer issue that prevented Connect 2020.2 from installing on Hyper-V (and some other virtual environments other than VMware) has been fixed. (SHARED-79958)

Fixed Connect 2020.2 uninstallation error

It was discovered that the Connect 2020.2 Designer would not fully clean up temporary folders when the Table Wizard was used. This would lead to "*Failed to uninstall*" error messages during uninstallation, even though Connect would uninstall successfully. This issue has been fixed in Connect 2020.2.1. (SHARED-79922/79765)

OL Connect 2020.2 Enhancements

Performance Changes

Changes have been made to the base performance of OL Connect and to the optional Performance packs. These provide greater parallel processing and a higher maximum

output speed than previously.

As a result, OL Connect will now process up to 64 jobs in parallel to provide better overall performance on smaller jobs. (SHARED-78819)

	Existing		New	
	Tasks that can run in parallel	Maximum speed*	Tasks that can run in parallel	Maximum speed*
PReS Connect	8	10,000	16	10,000
PReS Connect + Perf Pack	16	20,000	24	20,000
PReS Connect + 2 Perf Packs	24	30,000	32	40,000
PReS Connect + 3 Perf Packs	32	40,000	64	80,000

* *The Maximum Speed is measured in Pages Per Minute (PPM) and is the theoretical maximum speed that a job can be processed at. The speeds actually achieved, however, will be dependent upon a combination of how powerful the hardware is and the complexity of the jobs being processed.*

OL Connect Security Improvements

As part of our ongoing effort to transition to the cloud, we are improving the security of our software. This is ongoing process, with improvements made to this release and more planned for upcoming releases.

For this release, we have introduced role based authorization. Multiple users can now be setup, each with their own combination of roles. A *olc-user* user is added automatically at installation time (or the existing *ol-admin* retained when upgrading from earlier version) to allow initial access to the Server. Additional users can added after installation, using the **Server Configuration Tool**. (SHARED-75492)

Connect Designer will now prompt for credentials to connect to Connect Server when printing, if they are not already set.
Credentials will likewise need to be set manually in Connect Workflow.

In this release, the authentication is still based on local users managed by OL Connect. Expect further improvements in later releases.

The Workflow **PDF to Bitmap** plugin now requires Connect to produce an output. If Connect 2020.2 is not available, an error will be returned. (SHARED-77308)

This can cause problems when running older configurations that did not have this requirement. To cure such issues, set the Workflow *Connect Server connection settings*.

Other Security improvements

- Security of the Weaver engine has been improved. (SHARED-74605)

  Important note for those running an Enhance config in which a JavaScript uses Java classes. JavaScripts evaluated by the Weaver engine no longer have access to the Java API by default. In order for such an Enhance config to run, any necessary Java classes must be whitelisted. For further details, please see Output creation fails when Enhance configuration accesses the Java API in the OL Connect Workflow Help.

- Multiple OL Connect dependencies updated. (SHARED-76431)
- Improved support for encrypted email output (TLS/SSL). (SHARED-76460)

Print Manager Improvements

- Deleted printers were not cleared from the system and they could not be reintroduced. This has been fixed. (SHARED-41939)
- Standalone Print Manager installation now installs the Software Activation Tool. (SHARED-70967)
- Applying a job history limit will now take immediate effect. (SHARED-77071)
- Multi-job mode is the recommended setting for IPDS and it has now been made the default. (SHARED-77176)

General Improvements

- Cleaned up the error messages returned by Connect Server. Messages will no longer be duplicated, and they will not have exception class prefixes, such as "ApplicationException:". (SHARED-70963)
- Improved Server side error handling when a template fails to load. As part of this improvement we also now log the template name when a template is loaded. (SHARED-71024)

REST API Improvements

- Add option to download the "*octet output file stream*" as a file for remote tests. (SHARED-42559)
- Introduced a new endpoint that can be used to retrieve a list of persisted resources. (SHARED-70071)
  [GET] /rest/serverengine/filestore/resources/{type}

- '*Get Page Details for Content Set*' REST API endpoint of the Content Set Entity Service has been updated to return the correct number of JSON 'pages' objects regardless of the data record size or the number of merge engines used to create a given content set entity. (SHARED-71969)
- The endpoints for HTML content creation now return status code 400 (bad request) instead of 500 (internal server error) if the template does not have a *WEB context* or if the requested section was not found. (SHARED-76296)
- Introduced a new email POST endpoint that allows callers to pass a data set ID. (SHARED-76346)
  [POST] /rest/serverengine/workflow/contentcreation/email/{templateId}/{dataSetId}
- Introduced an endpoint that can be used to retrieve a report for a template (either JSON or XML). (SHARED-76535)
  [GET] /filestore/template/report/{fileId} (either a numeric ID or a name)
- Resources uploaded via the REST API are now classified based on file type. (SHARED-76979)
- You can now pass runtime parameters to Content Creation endpoints. (SHARED-77102)
- The '*Download Contents of Managed Directory*' REST API endpoint of the File Store Service has been updated to correctly return a '404 Not Found' response when an invalid path is specified for the 'relPath' path parameter. (SHARED-77822)
- Added **getManagedResult** endpoint to allow the results of *AllInOne* and *Output Creation* to be retrieved as ManagedFiles. (SHARED-77997)
- With the exception of the 'Upload Data File' method, all '*file upload*' methods in the File Store REST API Service have been updated to correctly return a '400 Bad Request' error response in the event that the request submitted contains no payload/data. (SHARED-78687)
- *Unauthorized/Forbidden access* to the REST API now returns JSON rather than plain text. (SHARED-78766)

Installer Improvements

- Add a feature to the **silent installer** of Connect to allow specifying a target folder other than the default folder, using "*install.directory=xxx*". (SHARED-75669)
- **Silent installations** can now specify connecting to a database with an encrypted connection, using "*database.usessl*". (SHARED-75671)
- **Silent installations** no longer need a connection to a database to complete. (SHARED-75775)

- In the Connect installer, the **Messenger Service** is now only installed when the Designer module is selected. (SHARED-76189/76318)
- Improvements made for installations on machines where Connect failed to uninstall correctly. (SHARED-77148)

Connect 2020.2 Designer Improvements

Improved Background Image options

- The **Section Background** functionality has been updated to support image types other than PDF. This includes .png, .jpg and .tiff (including multi-page TIFFs) files. This is also supported via the Control Script API.(SHARED-72730)
- In versions of Connect prior to 2020.2, if a page had no content except for a linked DataMapper background, then Connect would consider it an "empty" page when determining whether or not to "*Omit Master Page Back in case of an empty back page*" (available as an option in the sheet configuration of a section). This has now been fixed in Connect 2020.2, and such pages are no longer considered empty. (SHARED-72588)

  This can affect the output of existing templates.

Runtime Parameters for Content Creation

Runtime parameters provide a more elegant way to pass information from Workflow into the document creation process. They were previously available for Data Mapping and Job Creation, and this version makes them available for Content Creation. (SHARED-77345)

A new Parameters View has been added to the Designer, to list the available parameters, their default values and the current values set via the JSON editor of the template. (SHARED-77595)

Scripting Improvements

- Added ability to quickly search through script names and contents for specified text. (SHARED-60305)
- Auto suggestion improvements made for table properties in the JavaScript editor. (SHARED-70690)
- Introduced **prev()** and **next()** functions to the "*results*" object in our scripting API. (SHARED-70764)
- Fixed a problem with **Find Previous** in the script debugger. (SHARED-76737)

- Fixed minor issues in the script folder properties dialog. (SHARED-76996)
- Added the ability to retrieve the size of a print section or medium in any script, and the ability to change the size in a control script. (SHARED-77446)
- Fixed *results.add()* functionality. (SHARED-78447)

General Designer Improvements

- Inserting a *Web Form Element > Button* now has an option to add a new **Button** type in addition to the Submit and Reset options. (SHARED-75256)
- When the size of a print section is changed, you can now choose to "*Update linked media*" if the media is only used by that section. If it is not, then the option will not be available. (SHARED-65278)
- Holding shift while dragging boxes now drags the box in either horizontal or vertical directions. (SHARED-72158)
- Fixed failures with the '*Get Job Data File on Submit*' feature. (SHARED-72697)
- Fixed an issue with the *Export Report* function skipping Web information for some templates. (SHARED-73640)
- *Apply* button added to **Section Properties Dialogs**, which allows applying changes without closing the dialog. Clicking the *Cancel* button reverts any previously applied changes. (SHARED-74164)
- Fixed an issue with broken PDF hyperlinks that would occur when not all child elements of a hyperlink were on the same page. (SHARED-74482)
- Fixed some issues in the formatting dialog. (SHARED-74908)
- The case sensitive option of Conditional Scripts is now also available for HTML String fields. (SHARED-74970)
- Added support for multi-selections to the **Make Conditional** feature. (SHARED-76443)
- A change introduced in 2019.2 disallowed the selection of multiple elements in the Outline tree in Designer. This has now been remedied. (SHARED-76550)
- Fixed issue with *Save As...* option sometimes being disabled. (SHARED-76740)
- Fixed an issue whereby inserting a Table or Dynamic Table into a snippet would not mark that snippet as changed. (SHARED-76875)
- Introduced a new preference to control the behaviour when dragging data fields to the main editor. By default, scripts now use a class selector. (SHARED-76934)

- Introduced "*Send to Connect Server*", which can be used to upload templates and configs to Connect Server. (SHARED-77707)
- Fixed an issue where pagebreaks on the same page were ignored when a pagebreak was applied to the last row of a table. (SHARED-77946)
- Fixed an issue whereby some dates in JSON data would not resolve correctly in the Designer. (SHARED-78218)

Connect 2020.2 DataMapper Improvements

Improvements for PDF processing

- Improved search algorithm in the *data.find()* method used by the **Extract Step**. (SHARED-57386)
- Improved **GoTo step** search functionality. (SHARED-76494)
- Allow choice between the new improved search method or the old functionality (for backwards compatibility). (SHARED-77642)

JSON as new Data Type in DataMapper

In addition to the input data types already available in the DataMapper (Databases, PDF, XML, TXT/CSV files and the like), the JSON format is now also supported. As JSON is similar to XML in many ways, the information is presented in the same familiar structured way. (SHARED-77286)

General DataMapper Improvements

- Added support for resizing the height of tables in the *DataMapper Settings View*. (SHARED-75617)
- Added ability to connect to MySQL database with empty password. (SHARED-76292)
- Added *Save As* option for sample data. (SHARED-76453)
- AFP to PDF conversion issues fixed through updated 3rd party library. (SHARED-78144)
- Fixed issues with extracting Thai text from some PDF input files. (SHARED-78458)

Connect 2020.2 Output Improvements

2D Vector Barcode optimization

2D Scalable Vector Graphics (SVG) barcodes now optimized. (SHARED-77579)

This optimization includes these barcode types:

- Aztec Code
- DataMatrix
- QR Code
- Variations of the above, such as GS1 DataMatrix

True Type font handling

A major overhaul of TrueType font handling in the Connect output engine was conducted. This should help in reducing errors relating to both caching and parallel processing. (SHARED-61839)

PDF Digital Signature Improvements

- When creating a digitally signed PDF that includes a timestamp acquired from a timestamp server, Connect now automatically chooses the hash algorithm for the timestamp request.
  SHA-256 is tried first and if the timestamp server does not support it, Connect then tries SHA-384, SHA-512 or SHA-1 (in that order), defaulting to SHA-256 if none are supported. (SHARED-77850/78085)
- Fixed an issue whereby the use of certain certificates would result in incorrectly signed PDF output. (SHARED-77915)

PDF Output Improvements

The PDF Output options have been substantially improved. This required a redesign of the **PDF Options** page, to provide all the new options on a single Wizard page. The various options have now been placed in logically distinct tabs.

The improvements include the following:

- New image optimization options added for down-sampling mono, greyscale, and color images. (SHARED-75219)
  This includes a new option to "*Optimize for Fast Web View*". (SHARED-75210)
- Added PDF Security options. (SHARED-75213/75216)
  The Security options mirror those available in Adobe PDF applications.
- Improved support for processing digitally signed PDF/A-3 attachments. (SHARED-77267/78255)
  You can now choose to *Keep attachments* and whether to retain PDF/A-3 extension schema metadata, in order to support production of digitally signed PDF/A-3 using that metadata.

Print Output Improvements

- Fixed issues with missing glyphs and space characters in some double-byte jobs. (SHARED-74531)
- Placeholders for missing glyphs are now output as black, allowing output to remain monochrome. (SHARED-75513)
- Fixed issues with large double-byte font jobs. (SHARED-76730)
- The upper limit for the *Separation by Count* setting in the **Separation Options** Page has been increased to 10,000,000. (SHARED-77375)
- Optimized line drawing commands to reduce size of AFPDS output. (SHARED-77622)
- Update and improve **image support** in AFP, PCL and IPDS output. (SHARED-77716/77967/77972/77993)
- Added support for **Overlays** (these will be scaled to the needed size) to AFP output. (SHARED-77768)
- *Pattern support* setup code modified in PostScript output, so it will no longer interfere with custom *Page Device setup* from printer definitions.
- Fixed a license watermark issue in PostScript output with pages containing negative font size text. (SHARED-78076)
- Increased the limit of both **Black and White thresholds** to 50% for those print output types that support it. (SHARED-78298)

Email Output Improvements

- Improved error messaging for emails containing PDF attachments containing hyper-link images. (SHARED-76540)

Workflow 2020.2 Improvements

Thai Tax Invoice schema

Added new Thai Tax Invoice schema to the **PDF/A-3 Attachments** task. (SHARED-78258)

Workflow Capacity Improvement

The LPD Input task has been enhanced. (SHARED-56608)

Improvements include:

- Allowing multiple queue names to be specified.
- Support added for handling empty LPR queue names.
- Allowing PDFs to be created from the input document (like the Winqueue Input task), if the input is PostScript format.

General Workflow Improvements

- Connect resources from the **Configuration Components** treeview can now be dragged and dropped into a Workflow process, automatically adding the corresponding Connect plugin. (SHARED-70313)
- **Custom Plugin** categories created via imported PPKs will now be retained when the plugin toolbar is reset to default. (SHARED-73590)
- Fixed an issue with **Emulated Data Splitter** "split by page count" producing unnecessary extra split files for CSV data shorter than the page count specified. (SHARED-74076)
- **NodeJS Server Input**:
    - POST request parameter information now included in response data. (SHARED-75081)
    - Fixed default timeout value in NodeJS input plugin. (SHARED-76061)
- Further polling improvements mean processes that have a **Trigger-based SMTP Input** now use 0% CPU cycles until an email is ready to be processed. (SHARED-75299)
- **Render Email Content** now logs any non-fatal errors that occur during processing. (SHARED-76256)
- **HTTP Server Input** now instantly triggers for *only* the processes containing the called action, when triggers are being used for those particular processes. (SHARED-76232)
- Fixed an issue in the **Mailjet plugin** whereby emails containing a *Reply To* email address would fail. (SHARED-76449)
- **M-Files plugins**:
    - Both the Download and Upload plugins were updated to use newer plugin interface. (SHARED-76497/76500)
    - New functionality allows adding extra Property Row fields. (SHARED-76655)

> These updates require existing M-Files nodes to be re-opened and saved for the configurations to be converted to the new version, before working correctly.

- **DocuWare plugins:**
  - Plugin now outputs *document index data* to jobfile as XML/JSON. (SHARED-77207)
  - Improved connection error messaging. (SHARED-76367)
- **Custom plugins** set as inputs can now be configured to respect the polling interval set on the process. (SHARED-77869)

OL Connect Send Improvements

OLCS Cloud Support

Cloud support added to OLCS. Customers will be able to have any number of users printing to their Cloud system through OLCS. OLCS jobs that are received by cloud licenses of Workflow are encrypted so that only cloud licensed Connect can process them. (SHARED-78158)

Improved OLCS Printer Driver download and installation

A new way of pre-configuring the installer has been introduced that simplifies the installation process. The installer can now download settings directly from a web site. This means it is no longer necessary to download multiple files and follow instructions to install. Instead, simply download the installer, copy/paste a URL in the installation wizard and the installer takes care of the rest.

Specific changes include:

- Added installation option that allows remote print providers the option to provide their customers with customised OLCS printer driver installations, which do not require customer intervention. (SHARED-75200)
- Changed the compression method, resulting in increased performance when submitting some large files. (SHARED-75205)
- Improved “silent” (unattended) Printer Driver un-installation. (SHARED-72450)
- Installer no longer replaces '*localhost*' with 127.0.0.1. (SHARED-77192)
- A print service provider can now include customer subdomains, so their customer’s users will be able to print regardless of domain structure. (SHARED-77378)

- Added support for a limited number of users without domain restriction. (SHARED-77381)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2020.1

License Update Required for Upgrade to Connect 2020.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license now contains an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading to 2020.1 from Connect versions earlier than 2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Client, as version 1.2.40 of the Update Client can update your OL License to the required version, then install Connect 2020.1.

For further details on how to upgrade the Update Client and update your Connect License see the Update Client 1.2.40 Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing

production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2020.1 and Workflow 2020.1, as well as some important installation information.

Installing OL Connect 2025.1 and Workflow 2025.1

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Workflow Installation requires Administrator rights

To reduce potential later problems, please note that the Workflow installation can only be run by users who have Administrator rights.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2020.1 Enhancements

MySQL update and improvements

The database that comes bundled with OL Connect has been updated to the latest version of MySQL (8.0.18), along with the drivers used to connect to it. (SHARED-63665/74191)

The new packaged database will only be installed with new installations. This is to avoid any chance of issues that might occur in migrating an existing database to the new version. Existing users who want to upgrade their packaged database, will have to start with a fresh database.

Other MySQL improvements

- Connect now supports connecting to existing MySQL version 8 databases. Both as the Connect back-end database, or for DataMapping purposes. (SHARED-71285)
- Under certain circumstances, precision was lost when using MySQL databases, causing content on pages to be misplaced. The precision of numbers has been increased. (SHARED-74458)
- Fixed issue with the Connect cleanup service failing to detect partitioning support on newer MySQL instances (v8.x). (SHARED-74623)

Runtime parameters added to Data Mapping

Runtime parameters have been introduced to DataMapper configurations. The DataMapper engine can now use runtime parameters directly, when running DataMapper configurations. (SHARED-74572)

REST API Improvements

An output file's content/data is now correctly returned for synchronous print operations when the result is a single output file (requested via '*Process All-In-One (Adhoc Data)*' method of the **All-In-One Service** REST API service). (SHARED-75175)

Bypassing Content Creation

Many OL Connect jobs require making simple changes to print-ready batches of documents: adding OMR marks, commingling jobs, postal sorting, and the like. For those types of jobs, no merging of any data is required and yet, after data mapping is done, you must still go through the Content Creation step before moving on to Job Creation.

A new REST API method has now been introduced to allow you to create print content directly from paginated input: *DataMiningRestService.processPaginatedDmCc(dataFileID, json)*

This adds the ability to use an existing PDF as content, without using a template. Document boundaries are defined using a data mapper configuration. (SHARED-74995)

Security Improvements (and library updates)

- Connect dependencies have been updated to recent versions, curing a number of smaller security issues. (SHARED-73677)
- Oracle third party dependencies have been updated. (SHARED-74196)

- LincPDF has been updated. (SHARED-74370)
- NodeJS version updated to version 12.13.1. All npm packages were updated to latest versions.
  Vulnerabilities listed by npm audit goes from 54 to 0. (SHARED-73468)
- A myriad of other libraries have been updated. (SHARED-74605)

Print Manager Improvements/fixes

- Speed up loading of History table, to avoid issues with cleanup services failing. (SHARED-76485/76676)
- Fixed issue with Print Manager not removing Finished jobs when the selected language was set to a language other than the default. (SHARED-76922)
- Added the ability to limit the number of jobs loaded in the job History table. The setting is available in the Print Manager General Preferences page. (SHARED-76949)

General Improvements

- Several interface translation improvements and fixes have been made. (SHARED-73925)
- Improved logging of Connect engine activity. Particularly in regards engines (data mapper, merge engine, output engine) running in tight memory conditions. (SHARED-74548)
- Externally launched engines and HSQLDB database instances will now launch with the language specified in the 'Language' preferences. (SHARED-74807)
- Fixed issue with the "background.rotation" property having effect on section clones. (SHARED-76744)

Connect 2020.1 Designer Improvements

Improved Table wizard and Nested Tables

The 2020.1 release of OL Connect has a revamped **Insert Detail Table wizard** to help you build more sophisticated tables. (SHARED-56213)

It simplifies the process of setting up dynamic tables that show complementary information based on nested detail data. In previous Connect releases you had to write complex scripts to achieve this.

The new wizard lets you setup these tables from within the user interface of the application. Advanced users will find new HTML attributes at their disposal to fine tune the behaviour

even further. These attributes let you add data to your dynamic table and modify the table structure without writing a single script.

Improved backwards compability

Save a Copy now modifies script wizards in such a way that they will work in Connect versions that pre-date the wizards.. (SHARED-74183)

Scripting Improvements

- Introduced new `translate(string[, context])` script function that can be used to translate a string and optional context. (SHARED-74827)
- **Control script change**. Assigning the value "undefined" to masterFront or masterBack should override the master page settings in the Sheet Configuration dialog and ensure that no master page is used. In 2019.2 you needed to pass either "null" or an empty string. (SHARED-74887)
- Fixed an issue with scripting that could cause statements like query("<tbody>") or query("<tr>") to fail. (SHARED-74898)

General Designer Improvements

- The underline feature now creates wrappers with text-decoration:underline instead of <u> wrappers. This prevents a preflight warning about the <u> tag not being supported. (SHARED-69181)
- "Save as" now uses the directory of the original resource as default save location, unless the resources was never saved or was included in external package. (SHARED-70057)
- CSS can be pretty printed manually by invoking the new "**Format**" command or through the `Ctrl+Shift+F` shortcut. (SHARED-71419)
- Added support for custom properties and the `var()` keyword in CSS. (SHARED-72634) For more info, see https://developer.mozilla.org/en-US/docs/Web/CSS/--*
- Table columns can now only be re-sized in DESIGN mode. This change was introduced to prevent columns from being unintentionally re-sized in PREVIEW mode. (SHARED-73021)
- The **Styles** panel now includes styles for pseudo-elements like *::before* and *::after*. (SHARED-73327)
- Fixed a problem with the Align toolbar buttons not always properly aligning the selected elements. (SHARED-73493)

- Improved the checking of presets, to avoid prompting for unnecessary preset imports. (SHARED-74028)
- It is now possible to control which email section is processed by enabling or disabling email sections in a control script. (SHARED-74051)
- Fixed an issue with the Import Resources feature that could cause an UnsupportedOperationException error. (SHARED-74889)
- Improved the default line settings for the USPS IMb barcode. (SHARED-75097)
- Fixed an issue that would disallow changing font names in the Edit Font dialog. (SHARED-75525)

Connect 2020.1 DataMapping Improvements

XML data files DataMapper improvements

A new option to show all nodes of XML data file has been added. It is now possible to extract information from nodes that are outside of the record and come after the last record. (SHARED-73728)

General DataMapper Improvements

- Improved rename option for PDF sub-fields. It is now possible to rename the field directly from DataModel view. (SHARED-73514)
- Boundaries "On Change" was not triggered with empty data selection. Now the boundaries can evaluate correctly the boundaries split condition based on empty text, which can be selected even beyond the last character in a line of text. (SHARED-73863)
- DataMapper Wizard now opens database links more quickly. (SHARED-74279)
- Fixed problem with "Is Empty" operator for Condition step in Text emulation. (SHARED-74434)
- Improved region selection for Condition/Loop step. (SHARED-74451)
- Problems were encountered when importing certain PDF files. This was due to issues with a third-party library that have now been resolved. (SHARED-74631)
- Added the ability to re-order sample data files within the DataMapper. (SHARED-74767)
- Fixed issues within lower levels of nested tables. (SHARED-74790)
- Added ability to use an existing PDF as content, without using a template. Document boundaries are defined using a data mapper configuration. (SHARED-71632)

Connect 2020.1 Output Improvements

PDF Digital Signature Improvements

- The digital signature now includes the signer, the location, the reason for signing and the time of signing (if configured to use a time stamp). (SHARED-73528)
- Connect used to create digital signatures twice. Once to determine the size of the entire signature (including the timestamp if enabled) in the output file, and once for including the actual signature in the output. As of 2020.1 this "double clicking" of timestamping and digital signing has been removed, and the signature now only created the single time. (SHARED-71872)

New PDF Signature dialog box options.

- Can now pick an alias from a list of aliases, using new browse button. (SHARED-75183)
- It is now possible to specify the page of the output file that the signature should appear on. (SHARED-71881)

Print Output Improvements

- A retry option has been added for when the output file exists but cannot be deleted. (SHARED-74292)
- Print Output presets now have all non-relevant entries set to default values. (SHARED-74594)
- When a table was split across more than two pages, some table rows would be missing from the output. This issue has now been fixed. (SHARED-74903)
- Fixed a bug where in certain specific conditions with templates that generate a lot of pages, content would be missing from the output (detail tables, batched pagination). (SHARED-74940)
- Improved handling of angled and skewed text in AFPDS , IPDS and PCL outputs. (SHARED-75094)
- PDF initial view options are now available as settings. These allow you to specify how the PDF output is to be viewed upon opening. (SHARED-75224)
- Fixed a crash when uncoloured tiling patterns were used with an Indexed colour space. (SHARED-75250)
- Fixed issue with TTF outline characters appearing squashed in AFPDS and IPDS outputs. (SHARED-75462)

- Fixed an issue where setting Rasterize Options on a `div` would cause errors while generating output. (SHARED-75577)
- In PostScript level 3 output, when a shading refers to a named colorspace resource, Connect now ensures that the colorspace gets written before the shading itself. (SHARED-76342)
- Fixed a problem with decimal number precision when writing out PDF Type 3 font in PDF pass-through mode. (SHARED-76406)
- Fixed an issue with Ledger paper size in PCL output. (SHARED-76630)
- PCL custom page sizes failed to select the correct size, when paper in Landscape mode. This has now been corrected. (SHARED-76634)

Email Output Improvements

- Added a new CSS inline mode for email. (SHARED-74350)
- Restored support for email attachment paths containing spaces (which was broken in 2019.2). (SHARED-75050)
- Fixed a problem with scripted email headers that was introduced in 2019.2, whereby the headers were not reset after each record. (SHARED-75413)

General Output Improvements

- Fixed an issue where in rare cases the wrong indexToLocFormat could be written in a TrueType font's 'head' table, effectively corrupting the font. (SHARED-74383)
- Fixed an issue that crashed the output engine when processing some PDF input files that containing lots of resources (such as color spaces). (SHARED-74655)
- The Dutch Postal Service PostNL KIX database has been updated to the 2020 version. (SHARED-74801)
- Improved Connect support for Type 1 fonts. (SHARED-74910)
- Fixed an issue that broke WYSIWYG output when using a resizable table in combination with the rowspan attribute. (SHARED74915)
- Fixed an issue with text positioning encountered when writing text scaled down to zero. (SHARED-75096)

General Print Wizard Improvements

- Fixed an issue where occasionally custom OMR size validation code in the Wizard would fail with an unexplained validation error, preventing the Wizard from being

finished. (SHARED-74264)

- Fixed logic with media selection size rules when the media is landscape orientation. (SHARED-74540)

Workflow 2020.1 Improvements

New DocuWare connectors

We are continuing our focus on connectors to integrate with ECM systems, either in the cloud or on-premise. With Connect 2020.1, we are releasing a connector for uploading documents to DocuWare and another for downloading files from DocuWare. (SHARED-72654) These connectors simplify use cases such as

- Storing delivery notes in DocuWare when implementing our Proof of Delivery solution
- Archiving invoices, and downloading attachments to outgoing invoices from DocuWare in our Accounts Receivable solution.

Connect now provides easy integration with the following ECM systems: **DocuWare**, **M-Files** and **Therefore**. Both in the cloud and on-premise.

The new plugins can be found on the (help.objectiflune.com) site.

New Microsoft Office 365 connectors

New **Office 365 OneDrive** and **Office 365 Email input/output** tasks are now available in Workflow. (SHARED-74333)

The **OneDrive** input/output processes allow retrieval from, and posting to, any OneDrive account in the organization's Office 365 services.

The **Office 365 Email** input/output tasks allow processes to access any mailbox in the organization's Office 365.

Note that in order to achieve this, the organization's IT department must allow Workflow to access the services by generating an Application ID in the Azure Dashboard. The Application ID must then be provided in each of the tasks that want to access the Office 365 services. Without this Application ID, the services remain unavailable to Workflow and the plugins cannot be used.

In the future these Office 365 connectors will include SharePoint tasks, making Workflow even more cloud-aware.

New File Count option

A new Input/Condition **File Count** task has been added. (SHARED-75389)

This task checks if a folder contains a specified number of files before processing. This means a process can now be set to be executed only when all required files are available.

Execute Data Mapping plugin improvements

A new option to *Generate error when validation cannot be performed* allows Workflow to continue past a validation error when an invalid data format is detected.
For example, when erroneously sending a PDF file to a CSV validation Data Mapper. (SHARED-74404)|

Improvements made for version control comparisons

It is now possible to reliably track differences between changes to the Workflow configuration using version control tools. (SHARED-65310)

Updated Microsoft Access Database Engine

The Microsoft Access Database Engine has been updated to version 2016 (ACE 16.0) as the previous engine used (2010) is scheduled for end-of-life in 2020.

Installation of the Access Database Engine 2016 will be skipped if a later version of Access/Access Database Engine is already present on the machine. (SHARED-68863)

Improved efficiency of Input tasks

The following Input tasks now react to system events rather than polling for changes:

- HTTP Server Input
- LPD Input
- NodeJS Server Input

This reduces CPU usage levels, considerably. Especially in cases where the polling interval was set to 0, which means "check constantly". (SHARED-75302)

JSON files updated to UTF-8 encoding

The following plugins have been updated to use / expect UTF-8 encoded JSON job data files:

- Render Email Content
- Download EML Messages
- SendGrid
- Mailjet

If you intend to use Workflow 2020.1 then you should update to the latest SendGrid and Mailjet plugins. The updated plugins can be found on the (help.objectiflune.com) site.

If you are intend to continue using Workflow 2019.2 then you should not update the SendGrid and Mailjet plugins. (SHARED-75770)

General Workflow Improvements

- Custom HTTP headers can now be set in the **HTTP Client** input. (SHARED-66877)
- Uploads and Downloads to and from file store plugins can now accept backslashes as part of their name identifier. These are now automatically URL encoded. (SHARED-67632)
- Log message now displays the correct encoding used in the XML tag attribute when no encoding is found in the input file. (SHARED-70842)
- Allow the use of JSON arrays notation when **Retrieving Items** by ID. For example use `[189,1057,3002]` to retrieve those three specific items. (SHARED-71621)
- **SFTP plugin** can now create a folder on FTP server when outputting files. (SHARED-72503)
- Legacy version 4 configuration files can now be opened in Workflow. (SHARED-73396)
- SMTP Server ports were previously limited to 9999. This has been increased to allow a maximum port of 65,535. (SHARED-73458)
- **NodeJS Server Input**. Customizable constant DEFAULT_MULTIPART_MAXFIELDS added to default.js file. This can be changed to raise the maximum limit of allowed fields in a multipart/form-data post request. Note that the server must be restarted after the change. (SHARED-73613)
- **Secure Email Input/Output** plugins are now correctly translated when changing language. (SHARED-74053)
- Fixed an issue with scripted PDF section backgrounds not scaling correctly through Workflow. (SHARED-74431)

- Fixed minor GUI issues with custom plugins. These included:
    - Sometimes the plugin would pause and not get past the "Loading UI" screen, requiring rerunning the plugin. (SHARED-74518)
    - It was not possible to click the browse button a second time, as the background was disabled when file browser started. (SHARED-74560)
- Fixed **CreateJob** when using the "unselect unused content items in metadata" option. It now works even if only the content item IDs are provided in the metadata. There is no need for record ids to be present. (SHARED-74666)
- Fixed "respond on error" not working for legacy HTTP server plugin. (SHARED-75608

OL Connect Send Improvements

Authorization for printing

The Send client can be configured with an API key for all communications with the server receiving print jobs. This enables the server to reject any print jobs that are not submitted by an authorized client. By having different API keys for each subscriber, service providers can also use these API keys as an additional identification. API keys are widely used for this purpose in industry. Particularly by email service providers such as SendGrid, MailJet, and Tie Kinetix. (SHARED-74792)

Driver improvements

We have reviewed the printer driver's properties and default settings, so they better fit a virtual printer like OL Connect Send instead of an actual physical printer, and also ensure the driver is better behaved with applications that try to do special tricks. The net result of this is a driver that requires less tweaking of settings, and print output with better quality font information. (SHARED-71266)

General OLCS improvements

- Improved document handling in OLCS for documents with embedded elements. (SHARED-67755)
- The End point URL for interaction can now be different to the transfer point. (SHARED-75197)
- Fixed issue with printing to OLCS print driver from Adobe Reader, with "Choose paper source by PDF page size" enabled. (SHARED-75408)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2019.2

License Update Required for Upgrade to Connect 2019.x

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license will now contain an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading to 2019.2 from Connect versions pre-2019.1 to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Client, as version 1.2.40 of the Update Client can update your OL License to the required version, then install Connect 2019.2.

For further details on how to upgrade the Update Client and update your Connect License see the Update Client 1.2.40 Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Backup before Upgrading

It is recommended that you always backup your existing OL Connect preferences before upgrading to a new version. This will enable you to revert back to the previous version, in a worst case scenario in which the new version introduces issues with your existing production processes. Whilst the probability of such a worst case scenario is remote, it cannot hurt to take some simple precautions, just in case.

For instructions on how to do so, please see how to backup an existing Connect installation.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2019.2 and Workflow 2019.2, as well as some important installation information.

Installing OL Connect 2019.2 and Workflow 2019.2

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Workflow Installation requires Administrator rights

To reduce potential later problems, please note that the Workflow installation can only be run by users who have Administrator rights.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2019.2 Enhancements

Windows Server 2019 now officially supported

As of OL Connect 2019.2, Connect is now officially supported under Windows Server 2019. (SHARED-69995)

Performance Scheduling settings simplified and improved

Not only have the Performance Scheduling options been greatly simplified in Connect 2019.2, but the installer now dynamically sets the values such that fresh installations should perform a whole lot better out of the box. So good, in fact, that you might never have to bother with these preferences at all now.

But in case some tweaking is still required, the number of Connect Server settings has gone

down from 27 to 13, and those settings that remain have less inter-dependencies. There's now just a single Connect Designer setting, for example.

This results in a much cleaner and easier understood settings, now renamed as "**Parallel Processing**", to better reflect the reality that these settings impact upon jobs running in parallel.

We have also introduced Performance setting presets. Each preset caters for a different use case scenario. This vastly simplifies optimizing systems for the included scenarios.

Out of the box, OL Connect dynamically matches the setup of the engines with the hardware it is running on, typically giving better performance after a fresh installation. Only users who have a mixed workload with parallel running tasks will need to go beyond choosing a preset.

Connect Server now automatically determines how many engines should participate in **Content Creation**, and can also dynamically reassign engines to other tasks, if desired. This results in better utilization of computing power, keeping all engines busy when there is enough work and better ensuring large tasks do not block other tasks.

For **Output Creation**, the licensed speed limit is automatically distributed over multiple tasks when they run simultaneously, continually adjusting as tasks start and finish. The spread of the licensed speed across the job types can be controlled by setting of pages per minute (PPM) target speeds.

Saving Connect files for different versions

You can now save Connect templates and Datamapper configuration files in previous Connect version formats. (SHARED-71187)

Improved PDF Signing

We have overhauled OL Connect's functionality for digitally signing PDFs. Amongst other improvements, Connect now supports signing certificates on secure devices such as USB tokens, and HSM's. This enables applying digital signatures for use cases where regulations require secure storage of the certificate. (SHARED-71456)

The PDF signing improvements include:

- Support hardware security devices (USB tokens, smart cards, and Hardware Security Modules) for digitally signing a PDF. The settings for this can be found in the Preferences dialog.

- Storing PDF signing passwords encrypted in output preset files.
- More reliable handling of time stamp server requests and responses.
- Can now add a timestamp to all types of digital signatures.
- Support for invisible signatures.
- Better compliance for digitally signed PDF/A-1b, PDF/A3-b, PDF/X-4 or PDF/VT-1 output.

Named parameters in Job Presets

Connect 2019.2 introduces a way for Job Presets to specify which parameters they need from Workflow (or from any other source). Accordingly, the Create Job plugin in Workflow can now display a list of expected parameters that the user can fill with dynamic values. This makes it much easier to pass specific parameters from Workflow to the Connect Server without having to stuff all kinds of information in generic JobInfos.

In future versions, we are planning to make this same method available for data mapping, content creation and output creation.

Performance Improvements

- Improved output engine and DataMapper performance when processing common PDF files. (SHARED-70290)
- Improved default performance, for most users, courtesy of simplified and improved performance settings, and better initial assignment of Engines by the Connect installer.

Installer Improvements

- Issues with license importations have been fixed. (SHARED-67735)
- The Microsoft Access Database Engine has been updated to version 2016 (ACE 16.0) as the previously used engine (2010) is scheduled for end-of-life in 2020. (SHARED-68863)
- Under certain circumstances, some Connect Setups would fail when attempting to add the "*Enhance with Connect*" association with PDF files. This would then cause the setup to appear to fail. This issue has been fixed. (SHARED-69559)

Barcode Improvements

- Added support for the **Royal Mail Mailmark 4-State C** and **Mailmark 4-State L** barcodes. (SHARED-63467/69719)

- Several barcode names have been changed to better match how they are referred to in industry. (SHARED-72618)
  The biggest changes are:
  - OneCode has been renamed to USPS IMb
  - IMPB has been renamed to USPS IMpb
  - Royal Mail Mailmark has been renamed to Royal Mail 2D Mailmark
  - Royal Mail 4 State (RM4SCC) has been renamed to Royal Mail 4-state (CBC)

General Improvements

- Extra information has been added to the log files to aid in diagnosing problems. This information includes the Operating System, Regional settings as well as the available system memory and Hard Disk Drive space. (SHARED-70542/70729)

Connect 2019.2 Designer Improvements

Import resources from other Connect jobs

Introduced an option to import resources (such as scripts and snippets) and data models from other Connect template files. This makes sharing existing Connect resources much easier. (SHARED-26597)

Simplified use of dynamic background images

A new scripting wizard has been introduced, simplifying creation of documents with dynamic background images. These images could be stored in the template, on a local folder on disk or remotely on a web server. (SHARED-67948/71096)

Improved white-space behaviour

Designer adds an empty paragraph to each HTML editor, to allow easy (keyboard) editing. When this empty paragraph remains in a (non-remote) snippet, this can lead to undesired white-space in the section in which it gets included.

Designer now removes the empty paragraph from inserted HTML when the snippet is included in a section. (SHARED-68793)

NOTE: To prevent the removal of desired white-space lines (please do consider using styles instead), add a non-breakable space, or some other invisible code point.

Scripting Improvements

- When a master page editor is active in the Designer, the Post Pagination folder in the Designer Scripts panel is now decorated with an icon and a tooltip to clarify that post pagination scripts are not active in a master page editor. (SHARED-68543)
- Scripts can now differentiate between master pages and sections through an attribute selector. (SHARED-69811)
- Matches from cross-section queries are now highlighted when the mouse is hovering over a post pagination script. (SHARED-70156)
- Dragging and dropping a JSON or HTML snippet in the script editor automatically inserts the accompanying script commands (`loadhtml()` or `loadjson()`) and sets the path for the dropped resource.
  The `loadhtml()` and `loadjson()` functions now also have improved support for code completion. (SHARED-70687)
- The script editor and debugger now automatically mark all occurrences of a selected identifier, as well as marking whereabouts within the script they occur in. (SHARED-71453)

Script Debugger Improvements

- New hot keys added to the "*Find*" control in the Script debugger. They are **Enter** for "*Find Next*", and **Shift+Enter** for "*Find Previous*". (SHARED-69422)
- Improvements made to the information control that is displayed when hovering over a variable whilst a script is suspended. (SHARED-69861)
- The hover control in the script debugger now allows you to navigate to the children/parent of a results object. (SHARED-69949)
- Improved tooltips in variables panel. (SHARED-69986)
- Pressing **Ctrl+F** in the script debugger now starts an "*Incremental Find*" operation. (SHARED-70147)
- Fixed an issue whereby custom expressions were not always refreshed when stepping through the code. (SHARED-71276)
- Some cosmetic improvements made in the scripts overview within script debugger. (SHARED-72034)

General Designer Improvements

- To prevent issues it is no longer possible to replace absolute positioned direct child elements with inline elements in the body of Master Pages. (SHARED-68791)
- Formatting contained within a `<pre>` tag is no longer pretty-printed in the HTML source editor. (SHARED-69407)
- To prevent issues we no longer allow drag-drop operations in the outline tree if either the **Preview** or the **Live** tab is active. (SHARED-69640)
- We now generate a runtime error if a script tries to apply an unknown media or master name. A scripted master name with a NULL value or an empty string now resolves as "don't use a master". (SHARED-69683)
- When a standalone HTML file is opened in Designer, we now hide or disable the menu options that are not appropriate for standalone HTML files. (SHARED-69748)
- Added new `html(value)` function for Control Scripts that can be used to set the initial HTML of a section or master page. (SHARED-69782)
- Designer can now send more than 200 Emails in a single batch. (SHARED-70188)
- A **Collapse All** button has been added to the Data Model view toolbar. (SHARED-70451)
- Image file size restrictions removed. The only limit to images now is the memory available to Designer. (SHARED-70565)
- If a template is linked to a data model, code completion for `record.fields` and `record.tables` will now offer proposals even if no DataMapper config is open. (SHARED-70924)
- Can now drag remote stylesheet resources and remote javascript resources to a section or web context in order to add them to the section or context includes. (SHARED-70937)
- The **Send to Workflow** and **Package** dialogs will now only automatically add the latest job and output preset files when there is a template open with a print context. And only if those preset files where created or modified in the current session of the designer. (SHARED-71361)
- Saving a SASS partial file now auto compiles root level .scss files. (SHARED-72224)
- Modern, Roman and Script fonts are no longer available in the Designer Font ComboBox since these fonts are no longer supported. (SHARED-72391)

Connect 2019.2 DataMapping Improvements

Added option to "Stop data mapping"

New "*Stop data mapping*" option added to the **Action Step** to stop data processing at the specific record where the selected condition is met. As the data mapping operation then no longer needs to examine any further records, skipping those records can considerably improve performance. (SHARED-70320)

General DataMapper Improvements

- Improved field selection for **PDF split fields**. PDF split fields now have similar selection behaviour to simple fields. (SHARED-14120)
- **Boolean** fields can now be set as numerical values. A zero (0) equates to *false*, and any non-zero value equates to *true*. (SHARED-42848)
- Improved error messaging in the DataMapper. (SHARED-44404)
- **Duplicated fields names** in data files are now supported by Connect. (SHARED-49308)
- Improved performance of PDF Data Mapping when not doing any text extraction. (SHARED-67241)
- Significant improvements made to DataModel view when DataMapper is extracting a large number of data fields. (SHARED-67459)
- Improved compatibility with third-party PDF files. (SHARED-68651)
- Added a toolbar button to send the currently opened DataMapper configuration to Workflow server. (SHARED-70561)
- Improved Tab navigation in DataMapper workflow view. (SHARED-70738)
- You can now rename DataMapper fields directly. (SHARED-70940)
- Boundary Script changes now activate the '*Save*' button in DataMapper. (SHARED-72471)

Connect 2019.2 Output Improvements

Runtime parameters added to Job Creation

Added support for runtime parameters in Job Creation, so we can pass information from the actual runtime environment to compare against in conditions, or to use with external sorting programs. (SHARED-70085)

The runtime parameters will be available for use in Job Filtering and Finishing.

They can also be used as Meta Data fields, or in External Sorting.

The fields can be alphanumeric, numeric, Boolean, or even dates.

You will also be able to pass runtime parameters to the Connect Workflow **Job Creation** and **All In One** plugins.

Add customized PostScript commands to Dynamic PPD PostScript output

Added support for customized PostScript commands in Dynamic PPD PostScript printer output. The commands can include metadata values, so dynamic runtime information can be incorporated within the commands. (SHARED-72170)

Added bin selection to AFPDS, IPDS and PCL output

Added support for bin selection in AFPDS, IPDS and certain PCL printer definitions that support bins. (SHARED-71560)

Print Output Improvements

- When using a PDF file with a specific combination of page elements as background in the template ("*use PDF from datamapper input as section background*") and creating PostScript output, it was possible that some page elements could get positioned incorrectly. This has been fixed. (SHARED-68080)
- Fixed potential PDF/A or PDF/X compliance issue when outputting multiple channels or both variants simultaneously (as some custom OL Enhance workflows do). (SHARED-69642)
- AFP: Finishing now supported at DocumentSet and Segment Level. (SHARED-70047)
- AFP / IPDS: Multi-threaded option added to improve form ripping times. (SHARED-67154)
  Increasing the number of threads to use with form ripping could provide speed improvements for jobs that use a lot of graphics or resources imported from other print stream files.

- AFP / IPDS: Native support added for RGB images in FS45. The images are no longer converted to CMYK. (SHARED-70604)
- AFP / IPDS uses an improved RIP mechanism that results in cleaner, sharper lines and improved character alignment at 300 DPI. (SHARED-70631)
- PostScript: The output engine can now write tiling and shading patterns directly instead of first rasterizing them and writing them out as images. (SHARED-65125)
- PostScript (Dynamic PPD PostScript): Enable PCL Printer Job Language (PJL) comments to be selected and added to Postscript output as Job Control Language (JCL) blocks, if supported by the printer's PPD file. (SHARED-71504)
- PPML: Tray options now available for PPML print output. (SHARED-69314)
- Reduced memory usage requirements for large impositioned jobs. (SHARED-71177)
- Modern, Roman and Script fonts have been removed from the additional content font combo boxes since these fonts are no longer supported. (SHARED-72464)
- Ensured that PDF pass through is not active when the output PDF type is not standard PDF. (SHARED-72817)

Email Output Improvements

New Workflow Email plugins

Several new email related **Workflow plugins** have been created to improve the email sending performance by simplifying the integration with Email Service Providers (ESP). See "New email plugins" on the facing page section for further details.

General Email improvements

- Improved the scripting API for email headers. Connect now supports multiple headers with the same key. (SHARED-69927)
- Introduced the **Append plain-text copy of the HTML** option to email sections, to have a plain-text version of the HTML added to each email that is sent. This option is switched on by default for new templates. (SHARED-48902)

General Output Improvements

- Improved error handling when loading a template during output generation. (SHARED-67730)
- TrueType "OS/2" tables in embedded fonts are now passed through *as-is*, and the output engine will no longer abort if it does not recognize them (for example, a newer version of the format). (SHARED-69378)
- Fixed a problem with simulating font style (generating styled fonts). (SHARED-70471)

General Print Wizard Improvements

- Added support for processing tilde ('~') escape sequences in Additional Content barcodes. (SHARED-69388)
- Fixed handling of line end delimiters in Unix generated PPD files. (SHARED-70344)
- The Additional Content “EAN 128” barcode entry has been renamed to the international standard of "GS1-128". (SHARED-70798)
- Importation of PPDs improved. Some parsing issues were fixed, and the range of PPDs supported increased. (SHARED-71358/71450)

Print Manager Improvements

- Improved validation of page range values when resubmitting page ranges. (SHARED-58004)
- New **Export** and **Import** of Printer Configuration options added to Print Manager.

(SHARED-59828)

- Simplified interface for setting IPDS printer options. (SHARED-72238)

Workflow 2019.2 Improvements

New email plugins

Several new email related **Workflow plugins** have been created to improve the email sending performance by simplifying integration with external Email Service Providers (ESP). The original Create Email Content renders email messages and sends these over SMTP in a single run. This limits use to the slower SMTP protocol, which becomes a bottleneck when larger batches are processed.

The new **Render Email Content** plugin for Workflow prepares messages within the Connect File Store. This allows ESP specific plugins to later pickup the content and send it via their faster Web APIs. Sending email via ESPs also brings the advantage of open rate tracking, click through tracking and bounce handling all being presented in online dashboards.

> The **Render Email Content** plugin works only under OL Connect Workflows that are licensed for sending emails. It is not available for users with *Demo*, *Test* or *Reseller* licenses.

The associated **Download EML Messages** plugin retrieves the message body and attachments created by the *Render Email Content* plugin.

With Connect 2019.2 we also introduce connectors for the Email Service Providers ***SendGrid*** and ***MailJet***, to facilitate fast bulk processing. (SHARED-71120/71123)

These new connector plugins can be used to send pre-rendered email content (created by the new *Render Email Content* plugin) to either ***SendGrid*** or ***MailJet*** for bulk processing. The plugins will be made available via http://help.objectiflune.com.

New Documentation Management plugin category

A new **Documentation Management** category has been added to the Plugin Bar. This category is for plugins that integrate with Document Services/ECM/Document Management systems. Such as the **SharePoint** plugins (which have moved from the Connectors category) and the **Therefore** plugin. (SHARED-72133)

New M-Files connector plugins

New **M-File Upload** and **Download** plugins are being released in conjunction with Workflow 2019.2. (SHARED-71971)

These connectors can be used with both on-premise versions of M-Files, as well as with M-Files instances in the cloud. This makes it a lot easier to use M-Files in such use cases as:

- Proof Of Delivery - storing the delivery note in M-Files,
- Accounts Receivable - archiving invoices, and retrieving archived delivery notes to combine those with outgoing invoices.

These stand alone plugins will be available from http://help.objectiflune.com/en/#ppc-plugins.

New PDF to Bitmap plugin

A new **PDF to Bitmap** plugin provides functionality for converting page(s) of a PDF to either JPG or PNG image format. (SHARED-71217)

Updated PDF libraries

Workflow's PDF libraries have been updated to the latest available version, enabling the processing of larger PDF files while significantly improving overall stability, memory management and performance when merging documents or importing pages. The update fixes a number of issues reported by customers.

Workflow display improvements

Process branches and conditions can now be collapsed and expanded to allow a better overall view. (SHARED-63520)

CTRL-A now available

The CTRL-A (i.e. Select All) keyboard shortcut has been implemented in all edit/memo controls in Workflow, making it easier to select the entire contents of a field. (SHARED-47742)

Passing runtime parameters to Connect plugins

It is now possible to pass runtime parameters to the **All In One** plugin's Job Creation panel as well as to the **Job Creation** plugin itself. (SHARED-70097)

General Workflow Improvements

- **Create Email Content** now supports adding both the sender name and sender email address by expressing the email/name combination as "My Name <mye-mail@myhost.com>" in the Sender Address field. (SHARED-69901)
- Improved error reporting, as the actual error information is now included as part of the Error process.
  Whenever an error occurs in a plugin in Debug mode, the error message now appears in the Debug Information panel of the UI under the `error.errorlog` variable. (SHARED-53165)
- Added a new script function to the **Watch** service, for unpackaging resources synchronously to Workflow's documents folder . The new function is "`Watch.InstallResource("path")`". (SHARED-59908)
- **Execute Data Mapping** task now returns records in JSON format, to simplify determining boundary information. The output is identical to that provided by the **Retrieve Items** task. (SHARED-68256)
- The jobfile output of a **Create Web** output plugin is now `.html` by default, rather than `.dat`. (SHARED-69507)
- **XMLSplitter** plugin now displays the complete path of the selection instead of just the node name. (SHARED-69766)
- A new `getPreferences()` function has been added. It returns a JSON string containing Workflow preferences when used in Script/Custom tasks. (SHARED-70253)
- New option in Preferences/Default Config to set a Default Output task for Conditions and Branches. (SHARED-70377)
- When inserting the **NodeJS Server Input** task to a process, the initial value of the Form Data Encoding is now set to UTF-8. (SHARED-70529)
- A *Wait for completion* checkbox has been added to the **All in One** and **Create Print Output** tasks. The option will not be selected by default, unless the Output Management option is set to *Through Workflow*, in which case the option is ticked but disabled. (SHARED-70642)
- Minor bug fixes. (SHARED-69629/69631/69690)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2019.1

License Update Required for Upgrade to Connect 2019.1

From OL Connect 2019.1 onwards, only customers with a current OL Care subscription will be able to update Connect.

Every Connect license will now contain an end date that represents the last day of OL Care coverage for each customer. Upgrades and updates of Connect will be freely available up until that end date, but will not be available thereafter, unless the OL Care period is extended.

This new licensing model requires anyone upgrading to 2019.1 from earlier Connect versions to first update their OL Connect License.

A dialog box appears as part of the Connect upgrade process requesting you to do this.

It is heavily recommended that you first update the OL Update Client, as version 1.2.40 of the Update Client can update your OL License to the required version, then install Connect 2019.1.

For further details on how to upgrade the Update Client and update your Connect License see the Update Client 1.2.40 Upgrade Guide.

If issues arise, or if you need to renew your OL Care subscription, please contact your local Customer Care group.

Overview

This document provides an overview of the new features and enhancements in OL Connect 2019.1 and Workflow 2025.1, as well as some important installation information.

Installing OL Connect 2025.1 and Workflow 2025.1

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Connect Templates Used in Workflow

For improved performance we recommend re-saving any Connect templates used in Workflow to the current version of Connect templates.

Workflow Installation requires Administrator rights

To reduce potential later problems, please note that the Workflow installation can only be run by users who have Administrator rights.

Reduced Memory Version (*not* recommended for production)

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect 2019.1 Enhancements

Script Debugger

Connect 2019.1 introduces a new **Script Debugger**. This provides options to step through code (i.e. execute the code line by line) and to add breakpoints to pause execution at strategic points, as well as displaying the current state of variables, both local and global. (SHARED-66671)

The **Script Debugger** *simulates* an output run for all sections in the current context, including section clones. The output run is limited to running scripts and pagination for the current record. A print context will not generate actual output, and an email context will not generate attachments or send the email message.

The **Scripts** pane on the left hand side of the Script Debugger lists all scripts that are enabled to apply to the current context, in the order in which they are expected to be processed.

The **Source Code** pane shows the source of the suspended or selected script. Scripts based on a script wizard show the source of the expanded script. The current line in the script is highlighted, and the left margin of the source editor has a "pointer" marking the current line position within the greater script. You can ad breakpoints by clicking in the left margin of the source editor. The script will then pause at this line.

The **Variables** pane shows a hierarchical overview of the state of all local and global variables. It includes the special entry "(this)", which represents the JavaScript "this" object.

New Connect Project Wizards

OL Connect 2019.1 introduces a new suite of Project Wizards. (SHARED-67314)

These Project Wizards create building blocks to kick-start your projects as well as providing samples that explain Connect concepts and capabilities.

A Project Wizard generates a pre-configured Workflow configuration and associated Connect Templates, Data Mapper configuration, Job Presets and Output Presets. The Project Wizard creates a folder containing all the resources along with a workspace folder that is ready to be modified, tested and deployed.

Connect 2019.1 includes the following Project Wizards:

- **Print Promotional Jobs**, see how the *All In One* step is used to create a batch of personalized letters.
- **Print Transaction Jobs**, create print content once and use multiple *Create Output* tasks to generate various print output variants.
- **Serving a basic Webpage**, learn how to serve and personalize a simple web page using Workflow.
- **Submitting data with Webforms**, illustrates how to serve multiple web pages using a single Workflow process. In addition it stores submitted data in the Data Repository.
- **Basic email setup**, use a Workflow to send email messages. Attachments are added based on a static PDF stored with the template and pre-rendered PDFs stored on disk are attached based on the provided sample data.
- **Capture OnTheGo Timesheets**, a set of Workflow processes to deploy and serve COTG Timesheet forms. Additional processes capture the submitted data and illustrate how this information can be stored in JSON, XML and PDF formats. The PDF is generated by merging the data with a Connect template.

These new projects are also available from the Welcome screen. (SHARED-68777)

Improved Data Model view

- **Grouping/Ungrouping items**
  OL Connect 2019.1 introduces the ability to add and modify logical **Groups** within the Data Model view. This enhancement doesn't impact on the database structure itself, but it does allow you to manipulate the way the data is presented within Connect . Groups are a purely logical construct and can be nested within other groups, and

expanded or contracted as desired. This can be very handy for simplifying the display of complex databases within Connect. (SHARED-42643)

- **Ordering items**
  Data Model fields, tables and groups can now be re-ordered to create a fully customized Data Model view. Multiple fields can be selected together and drag & dropped either higher or lower in the Data Model pane, regardless of the order in which the data is actually being extracted. This does not affect the data at all, just the display of the data.
- **Grouping/Ordering selections made permanent**
  Grouping and Ordering information is saved within the data mapping configuration file. When one opens a configuration file, the Data Model view will display the same Grouping and Ordering information as was last saved.
- **Filtering the view**
  Data Model pane now allows filtering the displayed database items via a new **Filter** box. This narrows the list of displayed items to just those tables, groups or fields whose name contains the typed string.

New Translation feature

A multilingual Connect template can handle content and static text elements in more than one language. Traditionally this was done through creating a separate Template or Section for each of the language variants. A new **Translation** feature in Connect 2019.1 simplifies this process and introduces options to export translation strings in a standardized file format. (SHARED-21729)

OL Connect Designer lets you import translated string files and applies the translations based upon the current locale. The locale can be explicitly set for the template or dynamically defined by the value of a data field. The Translation feature even handles pluralization of strings.

The new translation workflow in a nutshell:

- **Tag page elements** for translation (for example paragraphs, headings, table cells, buttons, labels etc.). This adds the respective strings to the *Translations* view.
- **Export** the strings to a Portable Object Template (.pot) file. Send this file to translators.
- Translators **translate strings** and send the results back as a Portable Object (.po) file.
- **Add the .po file** to the *Translations* folder located in the *Resources* panel.

- **Set the desired locale** for the template either to a specific locale or set it dynamically based upon a data field.

Installer Improvements

- You can now modify your installation choices, when upgrading Connect from an earlier version. (SHARED-67956)
- The Database Connection Check now displays a progress monitor, which is useful when attempting to connect to remote databases. (SHARED-23309)
- Now able to connect to a non-Connect-installed instance of MySQL with an encrypted connection using SSL for main Connect application database interactions. (SHARED-54824)

Print Manager Improvements

- Improved paging speeds when viewing/processing large print spool files. (SHARED-67984)
- Fixed issues whereby some IPDS and PCL files not being displayed entirely correctly. (SHARED-54864)

Simplified access to MS SQL Server

Both the Connect Installer and the Designer/Server preference pages now allow selection of *MS SQL Server* by referencing instances through **Database Instance Name**. (SHARED-67994)

Connect 2019.1 Designer Updates

Scripting Improvements

- Regular JavaScript resources can now have *Defer* and *Async* properties set, as was already the case for remote JavaScript resources. (SHARED-67235)
- Scripts created by the *Make Conditional* option previously just used the ID and class of an element by default for the selector, but this can now be changed as desired from within the **Edit Script** dialog. (SHARED-68412)
- Text script wizards now validate whether fields exist in the active data model, and if they have the same types. The new **Refresh Types** button in the Edit Script dialog can be used to ensure that field types match the types in the active data model. (SHARED-65371)

Barcode Improvements

- Added support for the **Japan Post** barcode. (SHARED-59708 / 59711)
- Most barcodes except the 4 state barcodes (Australia Post, Japan Post, Kix and Royal Mail 4 State) and One Code now have an option to **enable\disable Tilde Processing**. (SHARED-68565)

  When this option is disabled, notations like "~d009" will no longer add a tab character to the barcode value. Rather the literal value "~d009" will be encoded.

  When this option is enabled you can escape a tilde by using another tilde (~~) for those data items which contain tildes that are meant to be used literally.
- The **KIX database** of the Dutch Postal Service PostNL has been updated to the 2019 version. (SHARED-68210)
- Fixed an issue whereby the barcode properties dialog would not appear for barcodes that didn't have an ID. (SHARED-50762)

General Designer Improvements and Fixes

- Added ability to copy the location of snippets and images, so the locations can be pasted into scripts to simplify using those resources in scripts. (SHARED-33976)
- We've added filter options to the *Package* and *Send To Worklow* dialogs, to make it easier to locate the correct presets. (SHARED-68033)
- The *Package* and *Send To Worklow* dialogs now automatically add the last modified **Output Creation** and **Job Creation** configurations to the package. (SHARED-68067)
- The "Database Connection" password now only needs to be entered once on the **preferences** page. (SHARED-66660)
- Added a *Shared Content Editing* option to the *View* menu with which you can disable/enable the editing of Shared Content in the template. (SHARED-60983)
- Default section clone names weren't always made unique when creating multiple clones. This could lead to lost clones, when naming conflicts arose. This issue has now been fixed. (SHARED-64156)

Connect 2019.1 DataMapping Updates

Stepping Improvements

In complex Data Mapping configurations, the Steps pane can contain hundreds of steps. This made the interface sluggish when the navigating from step to step. Accordingly, the

interface has been modified to redraw much more quickly than it used to.

Visually speaking, the dotted lines that used to connect each Step have been reworked to now use full lines that are drawn directly onto the display. To match the new look, all Step icons have been redesigned.

In addition to those visual elements, the background processes have also been overhauled to reduce the number of refresh operations occurring while working in the DataMapper.

Find functionality added to Steps

The Steps pane now includes a Find option that fetches a list of steps whose description contains the characters entered in the Find box (case insensitive).

The results are displayed next to the Find box, and one can click on the Next / Previous match buttons to immediately jump to the next/previous step within the list of matches.

XML Performance Improvements

The XML Viewer has been completely overhauled resulting in a much more responsive interface. It is now possible to scroll through extremely large records (i.e. thousands of elements) as easily as one would a text file. But the changes are not limited to the interface. Under the hood, the XML data mapping operation has also been greatly improved, with the result that XML data mapping operations should run significantly faster, especially for larger records.

XML boundaries

XML boundaries now can be split on variable element names. This solution adds the ability to specify an XPATH statement as the boundary, instead of a fixed element name. The XPATH statement supports dynamic values, allowing the boundaries to adjust dynamically to values passed by Workflow. (SHARED-65739)

General DataMapping Improvements

- Improved compatibility with third-party PDF files. (SHARED-68651)
- AFP input now logs a more meaningful error when no proper AFP input license is present. (SHARED-66804)
- Minor interface/behaviour issues fixed. Such as having the focus going missing after dragging and dropping a step. (SHARED-61252)

- Changed XML Datamapper font, to make zero character more easily distinguished from capital "O". (SHARED-69199)

Connect 2019.1 Output updates

Font handling

- Further improvements have been made to font handling within Connect. This should permanently resolve several of the issues previously experienced with Adobe font formats (Type 1, CFF, OpenType containing CFF). (SHARED-66561)
- Fixed problems with Type 3 font bounding boxes. (SHARED-62893)

Print Output Improvements

- Improved processing of Softmask graphics, to better prevent the flattening of files (which could lead to some images disappearing in printed output). (SHARED-61466)
- Enhanced print job summary in Server logs. A summary of print jobs will now be displayed in the Server logs when either printing to the Server from Designer, using an All-In-One plug-in from Workflow, or by submitting a request to the All-In-One Process. (SHARED-66793)
- Improved page breaking. This might impact upon some existing templates. (SHARED-68453)
- Extended the range of allowable Media weights. Media weight now ranges from 25 to 1000 gsm. (SHARED-69065)
- **AFP output**
    - An AFP Rules Editor has been introduced, to support dynamically created metadata in AFP output. (SHARED-65537)
    - Added support for creating and editing NOP and TLE records, in conjunction with the new Rules option. (SHARED-65545/65548)
    - Connect now supports the following types of Containers: (SHARED-67629)
        - PDF and PDF Transparent
        - PNG (RGB only; supports Masks)
        - TIF-AFPC (supports Masks)
        - TIF- OPAQUE
        - TIF-TRANSPARENT

- **AFP and IPDS output** - Connect now defaults to "*Scale to Fit*" for both IPDS and AFP output. (SHARED-68389).
  NOTE: This may have an impact when printing pre-existing Connect jobs on older printer models that do not support "*Scale to Fit*".
- **PCL output** - PCL uses a newer RIP mechanism that results in cleaner, sharper and improved alignment of characters at 300 DPI. (SHARED-33148)
- **PDF output**
  - Added support for e-Invoice PDF (**PDF/A-3b**) output. (SHARED-68209)
  - Redesigned the Digital Signatures section of the PDF output options to use a tree display which better reflects the structure of digital signatures. (SHARED-68364)
  - Added support for internal hyperlinks within a PDF document. (SHARED-52674)
  - XMP data now output in Canonical format for **PDF/VT** output. (SHARED-68483)
  - It's now possible to control the metadata in PDF output. The values for the fields can be static values or dynamic values (such as values obtained from datamapping). The PDF metadata fields that can be configured are Author, Title, Description and Keywords, which are the metadata entries displayed in the Adobe Reader Document Properties dialog. (SHARED-67501/68369)

General Output Improvements

- The PDF Preview function and PDF attachments created for Email messages now have the same producer name as standard Connect PDF output. (SHARED-56002)
- Improved memory footprint of the output engine. (SHARED-67225)
- Several improvements have been made to the output engine to ensure that output creation terminates even in exceptional circumstances. For example when handling faulty input files or in tight memory situations. Further, any exception that occurs in these circumstances will be reported immediately (improving feedback on what caused the error) and output creation will be terminated as soon as possible (fail fast). (SHARED-67401)

Print Wizard and Preset Wizard Improvements

Dynamic Print Control Improvements

Improvements have been made to the creation of Dynamic Print Controls.

- When working with PPD files to dynamically drive PostScript printers, some printers have fragments called *JobPatchFile* in their PPD, which are required for the printer to function properly. These are now supported and automatically added to the output if the PPD includes such. (SHARED-68025)
- Certain printers require some options to be in very specific locations in order for them to work properly. To support this, the Dynamic PPD rules now has two groups relating specifically to two of these locations: *Prolog* and *Setup*. These rather technical features allow us to support more PostScript printers than before. (SHARED-68025)
- When adding Media in Dynamic PPD we now validate the user defined Printer Options against the Rule Details. If the rules matches a Media in the template or Presets, it is highlighted in green. (SHARED-68401)
- An option has been added to rotate landscape pages. When this option is selected, any page detected as being in Landscape orientation will be rotated to match the standard PaperSizes (which are always in Portrait). (SHARED-65191)

Import multiple Presets

You can now import more than one preset at a time in the Job Preset and Output Preset wizards, as well as in the Print dialog. (SHARED-66727)

Improved Data Filtering Options

The Data Filtering page in the Print Wizard has been totally revamped. It has been enhanced with extra features, and the user interface overhauled. The user interface is now much more consistent with other rule builders available within the Print Wizard.

New functionality includes:

- Filter content items by number of **sheets** (rather than pages). This is better suited for filtering what fits within an envelope or inserter machine than using the number of pages,
- The use of content item **Property Rules**. This allows great flexibility because properties can be set during content creation, or from Workflow, or after data mapping has taken place.

General Print Wizard Improvements

- The **Additional Content** dialog now uses the selected Designer language's decimal and thousand separators in entry boxes. (SHARED-60881)
- Sample size preferences for standard and booklet imposition setups are now saved to the output preset. (SHARED-66484)

- In addition to being able to create multiple print files based on the number of sheets per file ("*Sheet Count Splitting*"), it's now possible to create multiple print files based on either the number of documents, documents sets or job segments per file. (SHARED-61932)
- Replaced the global timeout value (previously available for external sorting with a job preset specific timeout value. (SHARED-67982)

Workflow 2019.1 Updates

PDF/A-3 e-Invoices

A number of national standards now rely on PDF/A-3 as a standard for e-Invoicing. Workflow can now add invoice information and set Conformance levels for both the German (ZUGFeRD) and French (Factur-X) PDF/A-3 outputs via the new PDF/A-3 plug-in.

More e-Invoice standards will be added in future releases. (SHARED-69195)

NodeJS

NodeJS now accepts raw content types. It is possible to send a raw JSON, XML, HTML, CSV, CSS, JS, TXT and XHTML body. By specifying the content-type, Workflow will receive a file with an extension based on that content-type. If Node detects that the file sent is of another format, the extension will revert to .DAT. (SHARED-67199)

Capture/Send emails with SSL support

You may now retrieve emails from and send emails through secure online mail servers like Office365 or GMail. Different encryption methods can be specified, with the default (TLS v1.2) working out of the box with Office365 and Gmail.

The new **Secure Email Input** task allows capturing POP3 or IMAP emails with SSL support. This means Workflow can now process emails from multiple modern email servers. (SHARED-68489)

The new **Secure Email Output** task provides an option for sending emails with SSL support. (SHARED-68517)

Data Repository Manager Improvements

- Added new keyboard shortcuts for **Add Group** (Ctrl+G), **Add Key** (Ctrl+N) and **Add KeySet** (insert) functions. (SHARED-60155)
- Dialog now maintains focus when deleting or renaming Groups and Keys. (SHARED-61416)

General Workflow Improvements

- You can now specify the color of the highlight for any task to allow color coding of tasks or sections to make them stand out better. (SHARED-61963)

  This enables you to define personalised color schemes. For instance Scripts could always be highlighted in red, or each *Send To Folder* task could have a different color, based upon its destination folder.
- Connect resources which have been used are now logged when a Connect plugin finishes. (SHARED-64300)
- Licensing activation now works correctly on Amazon AWS platform M5 servers. (SHARED-65828)
- Logging for **Text Condition** task improved. When used for numeric comparisons you can select whether a numeric error triggers the condition to return *True*, *False* or a failure.
  If the Return *True* or *False*is selected, then the message will still be logged but with log level of 3 (Information) instead of a log level of 1 (Error). If the selection is to *Return the error*, the error is reported as such. (SHARED-xxxxx)

- New Workflow specific configuration Preference option introduced to Connect Designer to cater for instances where the Workflow is running on a different machine with a different *Windows codepage*. (SHARED-69053)
- New option added to the Workflow preferences to determine whether the current Workflow Configuration should be automatically re-opened when the Workflow application is next started. (SHARED-65133)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2018.2

Overview

This document provides an overview of the new features and enhancements in OL Connect 2018.2.1 and Workflow 2018.2, as well as some important installation information.

Installing OL Connect and Workflow

- OL Connect is released as a 64 Bit version only, although the Workflow, Fax, Search and Imaging modules are 32 Bit..
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Updating stand-alone Workflow Messenger installations

If Workflow Messenger was installed stand alone with no other Workflow components installed, the Update Client will be unable to find the Messenger component and thus it will not automatically update to the Workflow 2018.2 version. To get around this, download and run the Workflow 2018.2 installer manually.

Templates Used in Workflow

For improved performance we recommend re-saving Workflow templates set up in the previous versions to run with OL Connect 2018.2\Workflow 2018.2.

Upgrading from OL Connect 1.7

It is highly recommended that you update the **Objectif Lune Update Client** before upgrading OL Connect from version 1.7 to version 2018.2.

If you do not update the Update Client, an unexpected error might occur whilst updating Connect. This error does ***not*** prevent the successful upgrade of Connect to 2018.2, even though it appears as if it might have. To avoid potential confusion, we recommend that you first update the **Objectif Lune Update Client** before attempting to upgrade Connect from version 1.7.

The Update Client will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem to run the update while the Client itself is still open. It automatically updates itself.

Reduced Memory Version

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

Note: This is *not* recommended for production.

Connect 2018.2.1 Enhancements/Fixes

Backwards Compatibility Issues

Connect 2018.2 introduced some issues which have now been addressed in 2018.2.1. These issues were:

- Backwards compatibility issues with Workflow/REST API processes that incorporate Connect DataMapper configurations connecting to databases. (SHARED-67683)
- External sort metadata fields were not being catered for properly in the metadata validation code in the Output Preset and Print Wizards. (SHARED-67687)
- Fixed a problem affecting scripted Section configurations containing different scripted values for different records. The merge engine would sometimes use previously applied scripted values. (SHARED-67699)
- Fixed a problem with cloning template Sections that did not successfully clone all fields. (SHARED-67702)

Sassy CSS

OL Connect 2018.2 introduces **Sass** CSS preprocessor functionality to the Designer. (SHARED-64625)

Sass (*Syntactically Awesome Style Sheets*) is an extension of CSS that enables you to use things like variables, nested rules, inline imports and more.

Maintenance of CSS styles for templates requiring many different style rules can become a tedious undertaking. Stylesheets files keep getting larger and more complex and thus harder to maintain. Some common challenges include:

- How to group/organize rules that belong together? (e.g. styles for an address block, footer or dynamic table)
- How to update a color value or font family across multiple rules and even across stylesheet files?

A CSS preprocessor like Sass solves these problems and helps you write maintainable code.

In Connect Designer options have been added to the Stylesheets folder to create and compile Sass files (.scss file written using Sass 3 syntax). For more information on the Sass language see: www.sass-lang.com.

Compiling a .scss file outputs a normal CSS stylesheet file which is automatically linked to the current section.

Alternatively use drag and drop to link the result to other sections or use the Includes dialog of a specific section.

New ERP Template Wizards

ERP templates have been added to the Template Wizards. (SHARED-52963)

The new wizard allows you to create seven different transactional print documents, all with a uniform style. They all include a datamodel with appropriate scripting embedded. Basically, all you need to do is use the wizard to customize company information, logo, and color, and then create a data mapping that fits the template's model.

The seven document types provided are those often used in ERP environments: Sales Invoice, Sales Packing Slip, Sales Confirmation, Purchase Order, Project Invoice, Collection

Letter, and a Free Text invoice.

The current wizard is named after the style it uses: “Microperspective”. Other wizards with different styles may follow in future versions.

Dynamic Sheet Configuration Settings

A common requirement for dynamic print templates is the ability to switch between simplex and duplex. Typically this is solved by duplicating a Print Section, enabling Duplex in the Sheet Configuration dialog and then switching between the simplex and duplex sections by Control Script (or through the Conditional Print Section wizard).

In Connect 2018.2 new scripting commands have been added to dynamically set Sheet Configuration options from within a Control Script. This allows you to set the Duplex mode dynamically, or do things such as applying a different Master Pages based upon the data, or omitting empty sides from the last sheet, or single sheets.

The following Control Script sample enables Duplex mode for "*Section 1*":

```
let section = merge.template.contexts.PRINT.sections["Section 1"];
section.sheetConfig.duplex = true;
```

The following snippet sets the media and the master pages for the all sheet postions of "*Section 1*":

```
let section = merge.template.contexts.PRINT.sections["Section 1"];
section.sheetConfig.positions.all.media = "My Media";
section.sheetConfig.positions.all.allowContent = AllowContent.ALL_SIDES;
section.sheetConfig.positions.all.masterFront = "My Master Page";
section.sheetConfig.positions.all.masterBack = null;
```

Performance Improvements

- **Job Creation:** Improved job creation speeds. This is particularly the case with any job creation that involves any processing, including filtering, grouping or sorting. (SHARED-50328)
- **DataMapper:** The default amount of memory allocated to each DataMapper engine has been increased to improve both performance and memory management.
- **Output:** Improved memory handling, leading to even better stability.

Installer Improvements

- Support added for using **non-Latin-alphabet characters** in the User Name for connecting to external databases. This applies to both the Setup and the Preferences (Designer and Server Configuration) dialogs. (SHARED-65243)
- **Microsoft Visual C++ Runtime** now only installed when really required. (SHARED-63909)
- Several third party libraries have been updated. (SHARED-57163)
- A warning is now displayed if the **Windows Management Instrumentation** service is not enabled and started. (SHARED-61548)
- Installer now logs more helpful messages when failing to connect to an existing MySQL server. (SHARED-62836)

REST API Improvements

REST API calls that take resource IDs, whether on the URL or in a JSON payload, have been improved in two ways.

1. The existence of the resource represented by the ID is now verified. If it doesn't exist, a HTTP status code of either 403 or 404 is returned. Previously the result was undefined and varied depending upon the call made.

   Note: Calls that accept a variable number of resource IDs are not subject to this change for performance reasons, as this would require all resources be loaded upfront for validation.

2. Many errors are now reported using a standard JSON format.

   Note: Most such error messages are not translated.

Logging Improvements

- Improved error/warning logging in Live mode. (SHARED-49046)
- Extra preferences added to allow greater control over logging levels. (SHARED-63923) One change is the introduction of an option to set the **Overall Logging Level**. By default the logging level is set at the midpoint of *Info*, but it can be set higher to include more logging (*All*, *Trace*, *Debug*) or lower to reduce the logging (*Warning*, *Error*).
  NOTE: Higher logging settings will have an impact upon Connect production speeds, as well as leading to substantially larger log files.

- We have also introduced **Advanced Logging** preferences. The Advanced settings override the Overall logging settings, and provide a greater level of logging granularity. The Advanced Log Settings should only be set in conjunction with advice from OL support, to ensure that only the most relevant settings are set to the higher logging levels.
- The logging format default has changed from dd MMM yyyy (31 May 2018) to yyyy-MM-dd (2018-05-31) to make it easier to sort log files when using tools to merge and view multiple log files together. (SHARED-64233)
- Connect will now log a warning message in the event that there are insufficient engines currently available for a specific job, leading to the scheduler having to wait. (SHARED-61642)
- The location of the logging configuration file has moved from the Users directory to ProgramData to restrict the configuration to a per-machine basis. (SHARED-64147)
- Should a fatal error occur in the Java Virtual Machine running Connect the logs generated are now controlled and will appear in "*C:\ProgramData\Objectif Lune\OL Connect\ErrorLogs*" folder. (SHARED-64597)

Connect 2018.2 Designer Updates

JSON support added to the Data Model Panel

The Designer now allows you to add **JSON sample data** straight into the Data Model for debugging purposes. (SHARED-61655)

This can be done via a new **File menu option** or from the Data Model Panel itself, using the new **JSON Sample Data** button.

Invoking this option launches a wizard which allows you to either select a JSON file or paste JSON data copied from a Workflow variable. When loading data from a disk file the file contents are automatically placed in the JSON editor.

If JSON sample data is active invoking the **JSON Sample Data...** option will show the current data allowing you to edit its structure and values.

You can also now import or export a Data Model in JSON format. (SHARED-64620)

Minimum Pages option added to Print Sections

Connect 2018.2 introduces a new section property for controlling the minimum number of pages for a Print Section. This can be used to generate pages with only absolute positioned content, simplifying workflows where the number of pages are known up front and you want

to draw boxes (absolute positioned divs) on specific pages.

The property can be set when the new Template is being created or through the Print Section properties. (SHARED-10236)

Color Output option added to Print Contexts

For certain full color print jobs, it is important to be able to set black text on top of color areas to overprint. This is of particular importance for small text, because mis-registration problems are more visible with small text. To support this a new Color Output option has been added to the Print Context. (SHARED-62265)

The default encoding for the color black has also changed from RGB to CMYK. Thus the "*Keep RGB black in output*" option was added for backwards compatibility.

Breadcrumb Improvements

The breadcrumbs at the top of the workspace will now always end with a ~contents breadcrumb. When clicked this will select the *contents* of the last element rather than selecting the last element. (SHARED-17339)

Send To Workflow Improved

The Send To Workflow option has been simplified and streamlined, and a new Package option has been added. (SHARED-60392)

Paste as Plain Text option added

Option added to paste content copied from an external editor as plain text. This ensures any unwanted formatting and invisible elements are removed. (SHARED-63025)

Scripting Improvements

- An overview ruler has been added to the Script Editor. The ruler shows annotations concerning the entire script. These annotations are shown relative to their position in the script and do not move as the user scrolls the script source. Script errors are highlighted by a red icon, and warnings in yellow. The topmost icon will display red if any errors exist in the script at all.
  This allows you to immediately see if a script has warnings or errors and easily jump to any warning/error location by clicking on the annotations in the overview ruler. This is particularly handy when a script has many lines of code. (SHARED-64717)
- Personalization scripts can now use the "`loadtext`" function to load text content from files. (SHARED-61512)

- Option added to Scripting Preferences to set whether expanded scripts use single or double quotes. (SHARED-60370)
- Scripts can now conditionally trigger a fatal error by calling the new `fatalError(message)` function. This will abort content creation. (SHARED-64789)
- Scripts in Print Context jobs can now attach arbitrary properties to *content items* generated by Content Creation. The properties are retained in the database, allowing the information to be utilized in Workflow for further processing along with the base record information when retrieved by the Retrieve Items action. (SHARED-65239)
- Introduced "`width`" and "`height`" script functions with the same behaviour as jQuery's "`outerWidth`" and "`outerHeight`". (SHARED-65513)
- Fixed an issue whereby the `resource()` function did not work properly when passing a UNC path. (SHARED-65523)

Post Pagination Improvements

- We have introduced a new **Post Pagination** script type, and have updated the Scripts panel so that it now groups scripts by type. The groups are Control, Standard (which contains any pre-existing scripts) and Post Pagination. (SHARED-64634)
  These script types are grouped to emphasize their execution order. Control scripts are executed before the merge process and therefore run before the Standard and Pagination scripts.
- Several new script functions have been introduced for use with the new Post Pagination scripts. Scripts can now access elements in any section as well as retrieve information that is only available after pagination. This will allow for the creation of post content elements like a **Table of Contents**. (SHARED-64637)
- Margins for Sections and Master Pages are now scriptable, through the new `margins` object. (SHARED-62173)
- The sheet configuration of a Print Section is now scriptable, via the new `sheetConfig` object. (SHARED-63061)

Barcode Improvements

- Added **human readable text** and checksum controls to Royal Mail 4 State barcodes. (SHARED-47422)
- Added support for Grayscale “Colors”. (SHARED-50614)

Business Graphics Improvements

- **Color picker** options added throughout the Chart Properties dialog. (SHARED-62564)
- The "*Values (y)*" table in both the **Insert Graph** Wizard and the **Graph Script** dialog can now be filtered to show only numeric fields. This will then limit the display to only Integer, Float and Currency fields in the Data Model. (SHARED-64875)

General Designer Improvements

- **Download Remote Resources** (CSS stylesheets, JavaScript files, JSON and HTML snippets) to your template through a new context menu in the Resources view. This allows you to keep a link to a centralized file but quickly download a copy to your template without having to separately copy and paste.(SHARED-58949)
- **New Media** now defaults to the same paper size as the current Print Section. (SHARED-34012)
- Added an "**Omit Master Page Back in case of an empty back page**" option when configuring all sheets at once in the Print Section Properties. (SHARED-56383)
- **Media Properties** dialog behaviour improved. When any of the media in the Resources tree properties are changed, the "Page Setup" button gets activated and Designer prompts to save the media properties changes. (SHARED-62353)
- Disabled **irrelevant menu items** in the View menu when a Snippet editor is active. (SHARED-62816)
- When selecting content in the main editor, and selecting the context menu option of Create Shared Content or Create Snippet, you are now presented with a dialog in which to specify the name of the snippet resource. (SHARED-63217)
- Improved support for editing shorthand **!important** rules in the Edit Rule dialog. (SHARED-62863)
- **Preflight**: If a fatal error occurs during Data Mapping, the reported error will now also include the current record number. (SHARED-63089)
- **Help button** added to both the *Profile Scripts* and *Preflight* dialogs. (SHARED-63091)
- Template Wizard file comboboxes now add "Empty" value if not set. (SHARED-63294)
- Improved behaviour for when the **Auto Save** directory no longer exists. (SHARED-64428)
- Improved the way *Create Snippet* and *Create Shared Content* work. Creating a snippet for a selection of multiple elements in the Outline pane now creates a single snippet containing all the elements. (SHARED-64445)

- Improve support for UNC paths in Remote CSS. Connect now attempts to auto-correct UNC paths with the incorrect number of slashes. (SHARED-65308)
- **Zoom** options improved. (SHARED-64880/61294/61296)
- Apply button added to the Stylesheet **New Rule** dialog. (SHARED-60852)
- Fixed an issue whereby formatting could be lost after invoking undo or redo when the source tab is active. (SHARED-64734)

Connect 2018.2 DataMapping Updates

Ignore Blank Lines

Option added to the DataMapper Settings to **ignore any blank lines** in the data. (SHARED-57790)

Dynamic SQL Queries

Custom SQL queries now support the use of dynamic elements which can be built using JavaScript syntax. This allows an SQL query to use information passed on by Workflow to dynamically customize the query. (SHARED-34301)

For instance, a custom query could contain the following code:

```
=SELECT {automation.variables.FieldList} FROM {automation.jobInfo.JobInfo9}
```

This would get converted dynamically to something like `SELECT id, name FROM MyTable` based upon the values of Workflow process variables *FieldList* and *JobInfo(9)*.

Set Data Fields as Not Required

Individual data model fields can now be set as **required** or **not required**.
By default all data base fields are initially set to being required. This new option provides the ability to set individual fields as not required. Non-required fields are indicated in the Data Model panel by a green asterisk ('*') symbol after the field type. (SHARED-60067)

Landscape PDF Input

Rotated PDF pages are now fully supported in PDF input files. The DataMapper displays the pages in a human-friendly way that allows you to select and extract data according to the orientation. This allows the DataMapper to process PDFs that are either completely landscape or which contain a mix of portrait and landscape pages. (SHARED-37879)

Dynamic XPATH values

The Repeat Step, in XML mode, can now use dynamic XPATH syntax. This allows the Repeat step to loop through a subset of the entire record without having to use conditions to determine whether each element meets a specific condition. (SHARED-60081)

For instance, the XPATH statement for a Repeat step could contain something like: `=./address[@state="{automation.jobInfo.JobInfo9}"]`

This would get converted dynamically to something like `./address [@state="California"]` based on the value of *JobInfo(9)*.

General DataMapping Improvements:

- Can now set default values to **ExtraData** fields, in the same way as other fields. (SHARED-61348)
- Improvements in the way engines communicate in a **multithreaded environment** which should see a reduction in the number of times an engine fails to register with the server. (SHARED-63036)
- Added detection of **duplicated layered text** (where characters are painted in the exact same location multiple times) during text extraction to allow ignoring the duplicates. (SHARED-63547)
- Fixed issues with the `data.find()` function in TXT and PDF files. (SHARED-65582)

Connect 2018.2 Server Enhancements

Engine Setup

The settings for the number of Connect Server engines to launch has been removed from the "Scheduling" settings page, which controls how the engines are actually used. They have been moved into a new "Engine Setup" page that controls how many engines of each type are to be launched. This also simplifies the Scheduling preferences page.

Automatic Engine Restarts

To safeguard server availability restarting the Connect engines periodically is recommended. Previously Connect engines would restart every 8 hours by default, and all would restart simultaneously. Connect 2018.1 added the ability to have engines restart in case of very bad memory usage as well. Connect 2018.2 has now introduced several improvements to allow further control over engine restarts:

- **Rolling restart**
  Causes engines to restart one by one, instead of all at once. On systems with many engines this reduces the impact of restarts and increases engine availability.
- **Schedule restarts in quiet hours**
  Instead of restarting engines after a certain amount of time, they can now be restarted during a specified time period. This allows engine restarts to be scheduled outside of business hours.

Improvements for high workload situations

On busy systems, with many requests to the OL Connect Server (such as hundreds of simultaneous PDF preview requests), the internal communication between the Connect Server and the engines can now be set to use a different port from the primary connection port. This prevents the engines from being starved from connections to the Connect Server if the external connections are not closed fast enough.

These settings are located on the new "Connections" page of the Connect Server Configuration. The "Security" preferences page for the Connect Server connection is now a sub page of this new page.

The new settings are as follows:

- Allow inter-engine communication to occur on alternate ports, to reduce the chance of inter-engine communication being starved of connections when running on systems with a high numbers of HTTP connections. (SHARED-64035/65449)
  See highlighted in red boxes in following image.
- Set the maximum number of simultaneous asynchronous jobs. (SHARED-64043)

Connect 2018.2 Output updates

Important Information

The Output Engine (Weaver) memory footprint has increased in Connect 2018.2. This means some users might need to increase their Weaver memory allocation in the **Engine Setup** preferences.

This is especially the case for PReS Connect users running cluster configurations.

Note: When running really large jobs, it often pays to increase Weaver memory allocation, even if only for the duration of the production job(s).

Font handling

- **Additional Content Text improvements**
  When adding text as Additional Content in Output Presets, TrueType Collections (TTC) font files were not supported. TTC fonts allow a single file to hold multiple fonts types. Since TTC fonts can be important for Asian language fonts, we have now added support for these.

  Previously, Additional Content text allowed selecting Bold and Italic for any font, even if the bold or italic version was not present. We now only display the font variation options for a font, if that font variation is actually available.

- **PDF Output font selection**
  Added selection for either simple mixed byte font encoding (for smaller size) or fixed 2 byte CID fonts with identity encoding (for greatest compatibility) for PDF output. (SHARED-65462)

  Connect always used simple fonts in its output, but dynamically switched to composite fonts when the number of characters in a font grew too large. This approach best reduced output file size. But in some cases the dynamic font creation caused viewing issues with certain PDF viewers, even though the font method is compliant with PDF standards.

  To accommodate such, we have provided an alternative font creation method which always creates CID fonts with identity encoding. This results in larger output files, but provides better compatibility with PDF viewers.

  When generating PDF for non-Western languages, this new option may be the recommended choice.

- Asian language PDF output made smaller through improved glyph handling. (SHARED-63549)
- Further improvements made in handling simulated style fonts for OTF fonts. (SHARED-65402)
- Fixed an issue with output of TrueType font encoding information that could cause special characters to be missing or incorrectly substituted. (SHARED-62332)
- When a TrueType font "glyph count" information is incorrect Connect will now attempt to continue processing, rather than immediately throwing an exception. (SHARED-64382)

Email Output Improvements

- **Meta information added to the Email Section Properties**
  This allows viewpoint meta information to be added and configured in the email. This meta information will not be visible to the receiver, but can have an effect on how the email is represented in the email client. (SHARED-64761)
- **Email Script dialog improvements**
  The email Script dialogs have been streamlined and simplified.
  Improvements were made to the **To**, **CC**, **BCC**, **Reply-To**, **From**, **Subject** and **PDF Password** email scripts. (SHARED-61005)
- **Improved support for graphs in emails**
  As email clients don't usually support SVG images, graphs added to an email context are now rasterized by default. There is also a preflight message for emails containing graphs set without rasterize options. (SHARED-60546)

Print Output Improvements

- **AFP output speed improvement** for image intensive jobs containing lots of pages. (SHARED-55831)
- If a template containing **linked annotations** (either URL links or email links) was printed to PDF as PDF/A-1b, PDF/X4 or PDF/VT, the resulting output file was not compliant to the specified conformance level. This has been fixed. (SHARED-61714)
- We introduced a way to create **custom TLE** and **NOP records** in AFP output, through a custom printer definition. These records can have variable content, and they can be created conditionally. At the moment there is no GUI option to create these custom records and their rules, but this is intended for a future version. (SHARED-65527)
- In PDF/A-1b, PDF/X4 and PDF/VT output the document title in the PDF metadata is now derived from the template name, rather than the full path of the output file name. (SHARED-61168)

General Output Improvements

- Output engine now warns instead of crashing if it encounters text with a non-invertible transformation matrix (such as a zero point size). (SHARED-59744)
- Output engine now logs a warning if an image in an encrypted PDF file cannot pass-through untouched. (SHARED-59802)
- Fixed uncontrolled appearance of optional content (such as virtual stationery) in documents that are processed by the transparency flattener. (SHARED-60923)

- Fixed handling of output paths with one or more dollar signs ($) in them, such as certain network shares. (SHARED-63872)
- Several miscellaneous improvements to memory management.

Print Wizard and Preset Wizard Improvements

Size grouping

The ability to group documents by their size has been extended. In addition to grouping documents, you can now also group *document sets* and *job segments* by the number of pages. And instead of just using the page count, it is also now possible to use the sheet count for the size of the documents, sets, or segments. This makes size grouping useful for jobs that might contain both simplex and duplex content. And size grouping can now be used in combination with commingling when mail pieces can have multiple documents. (SHARED-61192)
This was previously referred to as "page break grouping", and is found on the same Grouping Options Wizard page.

The settings used in the screenshot above groups documents into document sets by their customer number. Size grouping then creates three different jobs segments: the first with sets of only 1 sheet, the second with sets of 2 to 5 sheets, and a third with anything of 6 sheets or greater.

> In Output Creation, you could create an output file per job segment, and use `${segment.metadata.ItemSize}` in the output file mask to automatically name these files appropriately.

Impositioning

We have made a number of improvements to Impositioning, both by improving usability and by adding new features. These changes led to a redesign of the layout of the imposition wizard page, with some options being moved and others renamed.

The major improvements are as follows.

Visual warning
The page layout diagram now displays in red when the selected imposition options do not fit the selected sheet.

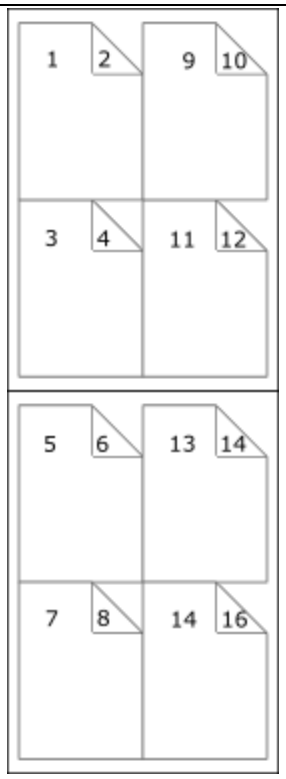
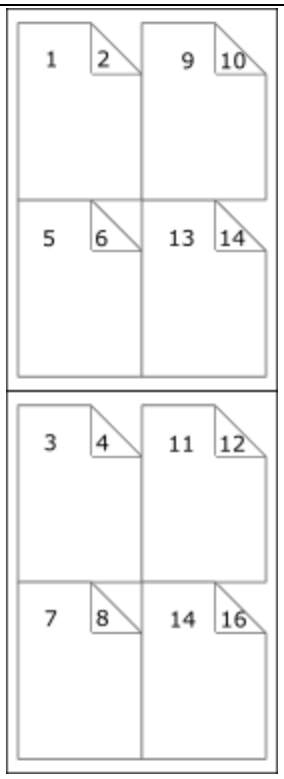
Sample page for imposition
The imposition settings are not always meant to work with just your current template. So we have added a sample page setting that will be used for both the preview and the validation of the settings. This sample page now defaults to the size and orientation of the first section of a template, instead of its first media.

Stack by Column

Alongside the existing cut and stack impositioning order, we have introduced an imposition order that is more suited for continuous feed printing when the imposition has multiple rows. We call this new ordering “Stack by Column”, and it is very suitable for roll fed label printing.

It works like this: instead of always positioning the subsequent page on the next sheet (until the stack depth is reached), stack by column first goes to the next row down on the sheet, and then down in the stack. The effect is that if all sheets are layed out top to bottom consecutively, the impositioned pages are ordered by column. If the imposition has only 1 row, both methods are identical.

Comparison of Stack by Column and Cut and Stack with 16 duplex pages in a 2×2 imposition.

Stack by Column (top to bottom, stack down, left to right)	**Cut and Stack** (stack down, top to bottom, left to right)
	

Reverse impositioning

In roll fed applications, reversing the imposition order can be very important, because, when done properly, this causes the roll output by the printer to have the first page on the outside of the roll, allowing the roll to start with the first page.

Our existing option to reverse the order of impositioning is reversing the order of the resulting imposition sheets by starting with the last side of the last sheet, and then going backwards. This effectively turns the stack up side down. It also means that, in case of

duplex impositioning, all back sides become front sides and vice versa.

To have an effective way of reversing a Stack by Column imposition that leaves the "by column" order in tact, but that starts at the last pages in the job, we have added a new option to reverse the imposition, that basically reverses the incoming sheet order before impositioning. The effect is that not only is the order of the imposition sheets reversed, but the order of the pages on the sheet is also reversed.

To properly distinguish between these two ways of reversing, we have renamed our existing reverse to "**Stack upside down**", and our new way of reversing is called "**Inverse page order**".

Rotate imposition

In addition to our existing options of having your imposition either upright or rotated 180 degrees upside down, we can now also rotate your imposition by 90 or 270 degrees.

Selective Inserts

Selective inserts in the *Inserter Mark Options* page now support metadata menus for inserting metadata field names in selective insert conditions. (SHARED-61152)

New AFP Options

We have introduced a new **AFP Options** page, to provide greater control over AFP output. (SHARED-63234)

Connect already had the ability to add metadata properties from Job Creation as TLEs to the AFP output. However, the earlier implementation did not support creating TLEs for specific pages, or for adding TLEs at the page level (which allows different values for each page).

TLEs are often used to mark the start of documents or mail pieces, to indicate media types, locations for inserts (including the name of the insert), and other properties required for print production. To accommodate this, it is now possible to conditionally create both TLEs and NOPs in AFP output by using inserts in custom printer definitions.

In addition to this, we have also added settings to control which job elements page groups will be created in the output. Earlier Connect versions would always create page groups for all job segments, document sets and documents, which would lead to pages being nested in three levels of page groups. Now you can choose to have just one level of page groups, or even no page groups at all.

To make sure that metadata properties from Job Creation do not interfere with dynamic TLEs from the printer definition inserts, this page also has a setting to turn off TLE creation for metadata properties.

Connect 2018.2 Print Manager updates

Job Cleanup Improvements

Option added to have finished Print Manager jobs automatically deleted after a set period of time. (SHARED-62981)

General Print Manager Improvements

- **Date formats** made consistent in the *History* tab, to assist in date searching. (SHARED-59679)
- **Text files** (*.txt) are now treated the same as comma separated text files (*.csv) in Print Manager. (SHARED-64666)
- Fixed issue with **IPDS Preview** of GOCA content sometimes displaying random lines across the page. (SHARED-63331)
- Fixed issue with printer connection not always closing after querying printer capabilities. This sometimes led to failures in subsequent printer capability queries. (SHARED-64498)

Workflow 2018.2 Updates

New HTTP Server Input - NodeJS

An alternative HTTP server option now ships with OL Connect Workflow: **NodeJS Sever Input**. (SHARED-62247)

The new implementation is backward-compatible with the existing HTTP Server Input (except for PlanetPress Capture requests), but also introduces several new capabilities. These include:

- The ability to serve static resources from several folders.
- Redirect requests to other servers using a Proxy list.
- Authenticate users using ActiveDirectory via LDAP.

Process Grouping

Support has been added for nested Process grouping in the Workflow Configuration Components. (SHARED-65457)

Groups can be added to existing groups, moved from one group to another or ungrouped within another group.

In addition, multiple startup processes can now be defined. (SHARED-65454)

This allows you to create startup processes that are specific to certain projects (or groups, or subgroups) and to specify the order in which the processes should run at launch time.

Combined with the multi-level process grouping feature, this allows you to design full projects of related processes that are initialised with their own specific startup process. With this capability, you no longer need to modify an existing, tried and tested startup process when introducing new functionality that requires some level of initialisation. Instead, you can simply create a new, distinct startup process that takes care of initialising values for the new set of processes.

Connect All-in-One Improvements

- *DataSet ID* now available in Workflow metadata after the All-In-One plugin has run. (SHARED-60116)
- When the last All-In-One operation is a Content Creation, proper record IDs are now available in the metadata. If the data comes from JSON, the record IDs are set to zero. (SHARED-63676)
- All-in-One can now accept a **Job Preset configuration**. The settings for this are contained in the new Job Creation tab. (SHARED-33510)

Create Job Improvements

- New option has been added to show only those Documents which are included in the job. (SHARED-53893)
- All Content Set IDs are now stored in the metadata returned by the Create Job task. (SHARED-61056)

DataMapper Validation

Option added to **Execute Data Mapping** to validate the records without actually extracting them. This provides an elegant option for determining potential data errors prior to actually running the job. (SHARED-57946)

When running with this option, the Data Mapping task returns a summary of the data mapping job as well as a list of records that would generate an error, if any. Since no data is

extracted, the operation is very quick and may save you a considerable amount of time by allowing you to check for errors prior to actually running the job.

Improved Logging

All tasks now include their Task Index (relative to their process or sub-process) in a fixed location on every entry they log.
The Task Index can be found between square brackets immediately after the time-stamp for each entry. (SHARED-64725)

Connect Workflow Improvements:

- The **Update Data Records** task now supports JSON data with detail tables, as well as Metadata. This allows the task to modify not only records, but also the information in any detail table for any record. (SHARED-55419)
- Improved error handling when invalid JSON data is passed to a **Create Content** task in Workflow. (SHARED-63584)
- **Retrieve items** with Record ID list now trims white space from end of Record. (SHARED-63603)
- When a **Retrieve Items** query returns multiple job sets, all the Job Set IDs are now stored in the Workflow Metadata at the Job level. (SHARED-64730)
- Improved **Create Web Content** task support for anchors. (SHARED-65445)

General Workflow Fixes and Improvements:

- Workflow will now *retry Connect commands only once*, and only when the configuration files are no longer valid (as determined by a 404 return code in the REST API). (SHARED-59821)
- Deleting processes and subprocesses now prompts for deletion confirmation. (SHARED-61378)
- The **Comments** tab in the **Branch** plugin now has a "*Use as task description*" option, to allow the comment to be used as the description in the Workflow process. (SHARED-61056)
- Optionally log only successful start and end times of any process, in the **Workflow process options** dialog. (SHARED-64698)
- Fixed potential security threats in **HTTP Service** static resources serving. (SHARED-61144)
- Timeout in Connect Send *Get Job Data* plugin now works as expected. (SHARED-61308)

- *GetKeySets()* command made backwards compatible. Command now retrieves all keys when parameter list is left empty or if the select all character '*' is used. (SHARED-64160)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 2018.1

Overview

This document provides an overview of the new features and enhancements in OL Connect 2018.1 and Workflow 2018.1, as well as some important installation information.

Installing OL Connect 2018.1 and Workflow 2018.1

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Updating stand-alone Workflow Messenger installations

If Workflow Messenger was installed stand alone with no other Workflow components installed, the Update Client will be unable to find the Messenger component and thus it will not automatically update to the Workflow 2018.1 version. To get around this, download and run the Workflow 2018.1 installer manually.

Templates Used in Workflow

For improved performance we recommend re-saving Workflow templates set up in the previous versions to run with OL Connect2018.1\Workflow 2018.1.

Upgrading from OL Connect 1.7

It is highly recommended that you update the **Objectif Lune Update Client** before upgrading OL Connect from version 1.7 to version 2018.1.

If you do not update the Update Client, an unexpected error might occur whilst updating Connect. This error does ***not*** prevent the successful upgrade of Connect to 2018.1, even

though it appears as if it might have. To avoid potential confusion, we recommend that you first update the **Objectif Lune Update Client** before attempting to upgrade Connect from version 1.7 to version 2018.1.

The Update Client will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem in running the update while the Client itself is still open. It will automatically update itself.

Reduced Memory Version

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

NOTE: This is *not* recommended for production.

Connect 2018.1.6 Enhancements/Fixes

Improved Weaver Engine memory usage

Weaver Engine memory usage has been considerably improved. These changes have resulted in the Weaver Engines not only requiring fewer system resources but have also improved stability and reduced the need for periodic restarts. (SHARED-65330)

Improved font handling

- Improved font style simulation, for when the full font family is not available. **Note** that this might cause some very minor output differences in comparison to earlier Connect 2018.1 font style simulations. (SHARED-65336)
- Incompatibilities between the Windows API and some custom fonts that required font style simulation could lead to Merge Engine failures. These issues have now been addressed. (SHARED-65334)

Blank pages encountered in jobs with PDF backgrounds of 30 pages or more

An issue was discovered in jobs in which the PDF background input exceeded 30 pages. In this scenario some pages would output blank after the 30 page limit was reached. This error was due to the internal image cache being exhausted and has now been fixed. (SHARED-65209)

Dynamic Printer Control not applying some rules correctly

Dynamic PPD rules that were supposed to apply to Job/File level were actually being applied at the Page/Sheet level. This error has now been fixed.
(SHARED-65328)

Connect 2018.1.5 Enhancements/Fixes

Blank page inserted in some multi-page transactional tables

Issues were encountered with some transactional print jobs, whereby a blank page would be inserted towards the end of some multi-page transactional records. This problem would only be encountered under very specific circumstances. Such as in duplex transactional documents with records that exceeded 15 pages, or where two documents with page numbers in the mid teens followed each other. These issues have now been fixed. (SHARED-64556)

Connect 2018.1.4 Enhancements/Fixes

PDF/A-1b compliance broken with Polish text

An issue was discovered with Polish text in PDF/A-1b compliant output, whereby the output was missing "*CIDSet*" font descriptor entries for TrueType-based CID fonts.
This error has now been fixed. (SHARED-64536/64816)

Issues with text extraction from PDF output

Problems were encountered in extracting non-latin and multilingual text from PDF output. These problems have now been addressed. (SHARED-64936)

Connect 2018.1.3 Enhancements/Fixes

Hyperlinks broken in PDF output

Hyperlinks in PDF output were broken in 2018.1. This issue has been fixed, and hyperlinks now work as expected in 2018.1.3. (SHARED-64353)

Connect 2018.1.2 Enhancements/Fixes

Issues running Connect jobs through Enhance

- When running multiple parallel output channels via Enhance configurations it was possible that some output channels could attempt to write the exact same content at the same time, thus using the same resources. This could lead to conflicts and resulted in errors and inconsistencies between outputs. This issue has now been fixed. (SHARED-63372)

- When creating PostScript output using the REST API to execute an Enhance configuration, the resulting PostScript file contained incorrect DSC comments if the name of the media used on a page contained one or more space characters. This has been fixed. (SHARED-64334)
- The PostNL KIX database has been updated to the latest 2018 version.
  Note: The KIX database is only available when using the REST API to execute Enhance configurations using the KIX functionality. Standard Connect usage does not allow use of the KIX database. (SHARED-64358)

OutOfMemory Errors in PostScript Output

When creating PostScript output or using the REST API to run a custom Enhance configuration containing one or more PostScript output channels, "*OutOfMemoryError: Metaspace*" errors could be encountered. This was due to JavaScript source code in the PostScript printer definition being compiled too often. This issue has now been fixed. (SHARED-64350)

Issue with background image URL paths

Fixed issues with URL path encoding. This issue applied to Section background image paths (\\host\path\image.png) containing special characters, such as spaces and hash signs ('#'). (SHARED-64528)

Special characters displayed incorrectly in Designer Snippet editor

Fixed an issue with the encoding of non-Latin-1 characters in HTML snippet editors. (SHARED-64530)

Connect 2018.1.1 Enhancements/Fixes

Section cloning issues

Two separate Section cloning issues were fixed. (SHARED-64087)
The issues were:

- Cloned content was being based upon the section for which *addBefore* or *addAfter* was invoked, rather than the section the clone was originally based on.
- The rendered page count could be incorrect. This was due to the page count of a section bundle (consisting of sections with continuous page numbering) mistakenly overwriting the page count of the previous section bundle.

Weaver Engine memory leak

Improvements made to how the Weaver Engine reclaims memory, in order to reduce any memory leakage. (SHARED-64205)

DataMapper issue with PDF input

In OL Connect 1.8 and previous versions, the DataMapper's `boundaries.find()` function returned the region searched within PDF files, whereas for text files it returned the exact region where the text was found. In 2018.1 this was changed so that `boundaries.find()` on PDFs would return the exact region where the text was found, the same as for text files. However, it was subsequently found that this could cause issues with previously created templates using the function on PDF files. Consequently, this change was reverted in 2018.1.1. (SHARED-64201)

Workflow - Updated *GetKeySets()* implementation now made backwards compatible

The Workflow function to retrieve key values *GetKeySets()* was updated in Workflow 2018.1 to use the more formerly correct asterisk '*' parameter for retrieving all key sets. This change broke existing jobs that used the previously allowable blank entry to retrieve all Key Sets.
In Workflow 2018.1.1 the *GetKeySets()* function has been changed to support both the more correct asterisk '*' parameter as well as the deprecated blank entry. (SHARED-64162)

Issue with Type 3 fonts in PDF Output

Type 3 fonts not embedded in PDF Output are replaced with font substitutions. These could become unmanageable in some circumstances. This issue has now been fixed. (SHARED-64214)

Connect 2018.1 General Enhancements

Dynamic Print Control for PostScript printers

Connect 2018.1 introduces a new way to drive PostScript printers. Individual PostScript printers can now be selected in the Printer Wizard and Output Preset via their associated PostScript Printer Definition (PPD) files:

This allows:

- The use of printer features that are not yet supported through the Connect user interface, such as:

    - switching between color and greyscale printing
    - selecting output bins
- Direct control of supported PostScript functionality, such as:
    - Duplex/Simplex
    - Print Media selection
    - Finishing options (e.g. Stapling, Binding and Folding)

NOTE: Case sensitivity/insensitivity has not been added as an option in this first cut of the Dynamic Print Control rules editor. This will be added in a subsequent version.

Improved Business Graphics

Business Graphics have been greatly enhanced and now display a preview of the data and graph as part of the Pie, Line and Bar chart wizards. The updated wizards simplify data selection and allows you to set and preview graph options dynamically, prior to inserting the graph object. This same functionality has been added to the scripting wizard, which can be launched from any Business Graphic script in the Scripts panel:

The Properties dialog in Connect lets you configure the graph via the User Interface and stores the configuration according to the specifications for this library. A Source tab has been added to the Graph Properties dialog so that you can view and edit the JSON configuration.

The implementation now supports the full Line, Bar and Pie chart API of the underlaying amChart library. This allows you to enable amCharts graph features that are not available in the user interface of Connect.

It is also possible to copy/paste Line, Bar and Pie chart configuration settings from the online amChart Live Editor directly to this tab.

Refer to the amChart documentation for more information on the numerous configuration options available.

NOTE: As a consequence of changes in both the user interface and the underlying technology, Business Graphics made with a version prior to OL Connect 2018.1 may not display correctly when opened in version 2018.1. See the ***Business Graphics: Backwards Compatibility*** section on the Known Issues page for full details.

Welcome Screen improvements

The Welcome screen layout has been redesigned and streamlined to simplify navigation and improve access to Connect resources, including current license details. (SHARED-61377)

Connect and Workflow version numbering changed

We have changed the version numbering system. From here on in, the version will be year based, with the second number showing the major release and subsequent numbers showing minor releases. The intention is to have two major releases a year, with minor releases in between to address specific customer issues.

2018.1 is the first of the two planned major releases for 2018. The second will have the version number 2018.2. Any minor releases between the two major releases will be labelled 2018.1.n, where "n" is a numeric counter. Thus the first minor release would be 2018.1.1 and the second would be 2018.1.2, and so forth. (SHARED-60069)

Windows Server 2016 now officially supported

As of OL Connect 2018.1, Connect is now officially supported under Windows Server 2016.

NOTE: The **Update Client** application might be blocked by enhanced security settings in Windows Server 2016. To fix this, add http://updates.ca.objectiflune.com to the list of trusted web sites on that machine, or lower the internet access rules.

Improved Previews

The Preview now smooths text, images and line art when rasterizing a page. This improves the quality of the resulting image, especially when rasterizing a page with lower resolutions. This applies to both Designer and DataMapper Previews. (SHARED-59724)

Localization improvements

Improved localization and translations throughout OL Connect and Workflow.

Improved Data handling:

- Improvements to the internal handling of data field names to support data fields that do not conform to a standard SQL query syntax. This means that fields with characters such as #, $, % and ^ can now be processed. (SHARED-59126)
- Performance improvements made to the Connect MySQL back-end database, through changes made to internal database indexing. (SHARED-59393)

Improved support for Microsoft SQL Server back-end

A number of enhancements have been made to improve the performance and support for Microsoft SQL Server (MS-SQL) as the Connect back-end database. These include:

- Database Table Partitioning, significantly speeding up the clean-up service. (SHARED-59589)
- Internal Connect SQL statements re-factored to work more efficiently with MS-SQL. Improving support for large jobs, in particular. (SHARED-57575)
- MS-SQL driver updated. (SHARED-60464)

New Preferences options:

- Scheduling preferences for Merge and Weaver engines now include a preference for specifying the memory (RAM) to be used per engine. (SHARED-44547)
- Merge and Weaver engine scheduling preferences now include an option for scheduling an engine restart if the total amount of memory (RAM) used by the engine exceeds the specified entry. This is in addition to the time limit setting. (SHARED-63101)
- Log file settings can now be adjusted within the Designer, Print Manager and Server Configuration preference windows. The settings are global to all Connect applications. Settings include log file rollover options (time or size based), and the format of the log messages generated. (SHARED-60043)

Installer Improvements:

- Prerequisites now installed silently when installing Connect using silent installer. (SHARED-59286)
- Messaging improved for when the Connect installer cannot safely commence uninstallation. (SHARED-59565)
- Installation folders cleaned up. Those that were no longer necessary have been removed, and others have been renamed or moved to make them more logically consistent.

Back-end Improvements:

- Better handling of non-responsive engines. (SHARED-55305)
- Improved response to Merge engine failures. (SHARED-46180)
- Improved handling of Master server restarts. (SHARED-57742)

- Reduced memory footprint when running the same job multiple times. (SHARED-57905)

Logging Improvements:

- Improvements made to Weaver logging for small output jobs and improved throughput. (SHARED-58570)
- Connect version number added to certain log files, to streamline issue identification and confirmation. (SHARED-60256)
- When using the REST API to execute a +PReS Enhance configuration that logged to a custom log file, the log file could not be deleted at the end of the execution. Repeating the execution of the +PReS Enhance configuration also caused unreadable NUL characters to appear in the log file and log messages were appended to it. This has been fixed and the custom log file now closes at the completion of the Enhance configuration execution. (SHARED-59656)
- Logging infrastructure has been updated. Logging can now be configured via an XML file located in the user's home directory under *Connect\workspace\configurations\logback.xml*. (SHARED-61913)

Connect 2018.1 Designer Enhancements/Fixes

Updated internal browser component

The browser component (Mozilla Gecko) used in the WYSIWYG editor of the Designer has been updated. This allows you to use new CSS properties like flexbox. Connect 2018.1 now uses Mozilla Gecko 38 as its HTML rendering engine. (SHARED-42286)

NOTE: The update to the Mozilla Gecko engine could lead to increased output file sizes for some PCL, AFPDS and IPDS jobs. This is generally not a cause for concern, however there might be some associated increase in processing times, as well as some minor differences in the output. For example, table line widths and font spacings might differ slightly (particularly for SMALL CAPS text), which could lead to slightly different word-wrapping in some cases.

Improved attachment options for Email output

New features have been introduced to simplify the setting of email attachments within Designer.

- **Dynamic file attachments** can be added to email sections via a new *Email Scripts* option. (SHARED-56602)
  This Script wizard allows you to construct a selection using the Prefix, Field and Suffix

values to dynamically specify a file from disk or a remote location.

- **Static file attachments** can be added to email sections via the *Attachments* tab in the *Email Section Properties* dialog. (SHARED-28343)
  You can also select whether the Print Context should be added as a PDF attachment or not, using the "**Attach Print context as PDF**" check-box.

Rotate and Scale Background Images

Section Background images can now be rotated or scaled, either via the Section Properties dialog or via a control script. (SHARED-60244/60855)

Align multiple objects

You can now select and align several absolute positioned objects at once with the new **Align Objects** context menu. (SHARED-60263)

Preflight improvements

Double-clicking a script warning or error in the Preflight report will now open the offending script within a script editor. (SHARED-60273)

Improved language support

Disable certain **CTRL + ALT + <?>** shortcuts in source editors, including source tab, HTML snippet editors and CSS editors.

This allows adding square brackets in a source editor when using a French keyboard layout shortcut (`CTRL + ALT + [` or `CTRL + ALT + ]`), as well as semi-colons (`CTRL + ALT + ;`) on Hungarian keyboard layouts. (SHARED-60308/62056)

Data Model pane improvements

The context menu in the Data Model view now allows you to modify the data model while the template editor is active. (SHARED-60234)

Scripting improvements:

- Option added to allow Case Sensitivity to be set on or off for any conditional string comparison scripts. (SHARED-56535)
- Improved support in Text Script Wizards for the custom formatting of Date, Currency, and Numeric fields. (SHARED-60261)
  Documentation for these patterns can be found on the following pages:

    - Date and Time patterns in Connect
    - Number patterns in Connect
- You can now inspect Scripted elements in the Outline tree when in Preview mode. (SHARED-60231)
- New sub-folders now inherit the same execution scope as their parent. (SHARED-56964)
- You can now drag and drop a data field to a script folder even when a script of that name already exists within a different folder (or root) of the Scripts panel. (SHARED-57027)
- Improved Email Context script selection. All the Email Context scripts are now bundled together in their own sub-menu. (SHARED-56602)
- Improvements made to the default data field selection logic when adding new Text Scripts. (SHARED-57286)
- Double clicking a script in the Script Panel with the script editor open will now prompt you to save your changes. (SHARED-47908)
- Any open Script Editor will now close when another template is opened. (SHARED-32289)

General Designer improvements:

- Style rule changes can now be saved and applied instantly from within the Edit Rule dialog, via a new Apply button. (SHARED-58595)
- A new **Send to Workflow** icon has been added to Designer. It has been added as a new option in the icon toolbar and also to the File menu option. (SHARED-59322)
- You can now copy/paste or drag/drop images files from the file system or *Images* folder to the *Stylesheets* folder. (SHARED-49551)
- The **New Email** and **New Print** template wizards now open using the last used Template settings. Selections such as page size, page margins and the like will be retained. (SHARED-33982)
- Improved **element insertion** rules, to make context menus and dialogs more accurate. For example, the option to add Absolute elements in emails has been removed, as the email format does not support this functionality. (SHARED-60947/61117)
- Improved support for **remote images** (http-based URLs). (SHARED-58592)
- The *resource()* script function has been extended to return information about PDF permissions. (SHARED-58026)

- The attributes panel now has a "*Reset Size*" button that reverts a resized image back to its the original size. (SHARED-57025)
- You can now clear the "*recent files*" list via new **Window > Clear Recent Files Lists** menu option. (SHARED-42712)
- You can now toggle between the front and back of the selected Media entry in Master Pages. (SHARED-62038)
- You can now use cursor keys to navigate after or <article> elements that are the last element in the <body>. (SHARED-41710)
- Dialog box resizing improved and made more consistent. (SHARED-16164)

Connect 2018.1 DataMapping Enhancements/Fixes

Standalone DataMapper Engine

The DataMapper Engine has been stripped into a separate stand-alone server. This is to make it more efficient and configurable as well as more adaptable in the future. The new DataMapper Engine options can be set via the **Connect Server Configuration** tool. (SHARED-59696)

Improved date field support for Excel datafiles

Formatting for date fields in Excel data files has been improved. (SHARED-60041)

Ignore CR/LF in Text records

New option added to allow ignore end-of-line CR/LF characters in Text datafiles. This prevents empty trailing records being added to the dataset. (SHARED-62139)

Create unique GUID in scripts

New DataMapper Script function `createGUID()` added to the Boundaries and Steps scripts. This function returns a unique GUID. (SHARED-61246)

New "No Data" warning

A visual warning has been added to the DataMapper when an open table contains no data. (SHARED-2716)

General DataMapping improvements:

- Improved support for transactional data records containing thousands of transactional items per record. (SHARED-50430)

- Data records can now be duplicated. To support this, a **record.copies** property has been added to the record object. (SHARED-53766)
- It is now possible to add a CSV input file containing just a single column. (SHARED-59814)
- Improved DataMapping performance for XML files containing large numbers of end-of-line CR/LF characters. (SHARED-60940)

Connect 2018.1 Output Enhancements/Fixes

Redesigned Additional Content page in the Print Wizard

The Additional Content page in the Print Wizard has been re-designed to present all the Additional Content entries in a single table, simplifying Additional Content management. (SHARED-56444)

Improved Tray Mapping in Printer Wizard

Media attributes can now be imported directly into the output config tray mapping from either the currently open template or from a saved template file. (SHARED-48818)

New Conditional option

During Output Creation, it is now possible to determine the template section to which a sheet belongs to using the new `sheet.sectionName` property. This property has been made available as a conditional option in the Printer Wizard. (SHARED-61275)

NOTE: The introduction of this property introduces a slight risk of compatibility issues, but only for cases where an existing custom printer definition or Enhance configuration already has a user-defined property with the *exact same name*.

Font handling made more robust

Connect will now generate **bold**, *italic* and ***bold-italic*** variations of most fonts on machines that do not have those variants of the font installed. (SHARED-59975/60670)

Improved PDF throughput processing:

- Improvements were made to the processing and optimization of barcodes in imported PDFs. This has resulted in speed improvements when processing certain PDF files for print output. This is particularly the case for AFPDS, IPDS and PCL output. (SHARED-58782/59955)

- Enhancements made in the caching of PDF input files has led to speed improvements when processing large PDF inputs. (SHARED-58503)
- Improved memory usage with PDF pass-through. (SHARED-58782)
- PDF pass-through is now enabled by default for PDF containers in AFP and IPDS output. (SHARED-61494)

Print Output Improvements:

- Transparency support for PDF output. (SHARED-61492)
- Soft mask improvement in PDF. The soft mask is now set by the CTM ("Current Transformation Matrix") when the soft mask is created. (SHARED-58893)
- Improvements made to generic PCL output for some printers. (SHARED-60019)
- Booklet Impositioning for mix-plex jobs has been improved, reducing the likelihood of empty pages. (SHARED-60054)

Connect 2018.1 Print Manager Enhancements/Fixes

Several improvements have been made to the Print Manager. Some to better cater for continuous IPDS printing, and others to improve useability.

Hot Folder improvements

The Hot Folder dialog has been updated and improved. Improvements include:

- Simplified selection of Hot Folders
- Option added to put Hot Folders 'On Hold'.
  If selected, jobs will be assigned to the target printer but put 'On Hold' and not be taken from the queue until released. (SHARED-61224)
- Files are now moved from the Hot Folders to a processing sub folder and processed from there. (SHARED-60180)

General Print Manager improvements and fixes:

- Improved handling of duplicated job names. (SHARED-34710)
- Job status added to the job filename, to make it easier to identify the current job state. (SHARED-60182)
- Improvements made to file status updating. (SHARED-58122/58172)

Workflow 2018.1 Enhancements/Fixes

Improved Performance

The performance of the **Execute Data Mapping** and **Retrieve Items** tasks have been improved when setting the options to "Output Records to Metadata". (SHARED-56678)

NOTE: The performance update to **Retrieve Item** is currently only implemented for Data Records. The other entity types will be added in a subsequent release.

Retrieve Items by their IDs

Option added to the **Retrieve Items** task to support retrieving items via their associated entity IDs. (SHARED-51101)

New option in Digital Action and Output Image tasks

A checkbox has been added to allow the PDI FormName to be replaced by the *Title* defined in the Job Options tab when the input file type is PDF/VT. (SHARED-61165)

New option in Create Web Content task

An option to "Do not alter HTML" was added to the **Create Web Content** task. When selected the task shall not add the BASE element to the HTML as it usually does, nor shall it modify the local anchors with JavaScript code. (SHARED-62521)

New option in FTP Input task

An option to "Search in subfolders" has been added to the **FTP Input** task. (SHARED-54684)

New Workflow Preferences:

- You can now set the default Workflow scripting language. This defaults to VB Script for backwards comparability. (SHARED-60000)
- Improved PHP like Array options are now available for the **HTTP Service** plug-in. (SHARED-50770)

Number of email attachments now available

The number of email attachments is now available to the **Email Input** task. (SHARED-59294)

GUID variable introduced

A new **%U** system variable has been introduced to support full GUID values. This new variable is an addition to the pre-existing *%u* (a unique 15-char string). The new %U system variable is available in the same locations that the %u system variable is, including in the context menus. (SHARED-61244)

Improved support for special characters

When retrieving data/strings from the data repository manager through the scripting API command *GetKeySets()*, special characters like 'é è ê ë æ' got corrupted in the output.

GetKeySets() now converts the results coming from the Repository from UTF8 to ANSI for compatibility with scripting. *GetKeySetsW()* can still be used to obtain the UTF8 value without conversion. (SHARED-62189)

Workflow Service Console improvements:

- Option added to browse either the Connect or Workflow logs for the current user on this machine. (SHARED-60021)
- Text can now be copied from the service console logs using the new **File>Copy** menu option. (SHARED-60025)
- Options for setting the logging refresh speed have been added. (SHARED-60029)

Workflow Data Repository Manager improvements:

- Improved Data Repository Key Set editor. (SHARED-59137/60165)
- Improved speed when writing to a Data Repository. (SHARED-51564)
- Instead of throwing an error the `lookup()` function now returns `NODATA` when an invalid group and/or key is passed. (SHARED-48508)

- A new `ConnectionString` property was added to the Repository API to allow the selection of a custom Data Repository. (SHARED-59335)
- The Repository API `GetKeySets()` function can now select *all* keys . (SHARED-47769)

General Workflow fixes and improvements:

- Special characters which may conflict when present in a URL (such as '`@`' or '`/`') can now be used in the "Server Connection Settings" username and password. (SHARED-58820)
- Improved logging of REST API calls to Connect. (SHARED-60241)
- Workflow can now use Connect Output Presets that use fonts for additional text (and also for human readable text of barcodes), provided that each font used is installed on every machine involved in the output creation. (SHARED-61903)
- Plugins that retrieve data using the `region(...)` command would sometimes fail intermittently. These failures have now been fixed. (SHARED-61345)
- In Alambic-intensive Workflow configurations instances of `ppalmbic.exe` would sometimes remain open even though they should have stopped. This has now been fixed, and the `ppalmbic.exe` instances close as expected. (SHARED-59435)
- Fixed issue whereby Workflow would stop processing PostScript and PDF jobs on some *Windows 10 Fall Creators Update* machines. (SHARED-62516)

Known Issues

As of OL Connect 2018.1 the Known Issues have moved from the Release Notes to the Connect online Help.

They can be found here.

OL Connect Release Notes 1.8

Overview

This document provides an overview of the new features and enhancements in OL Connect 1.8 and Workflow 8.8, as well as some important installation information.

Upgrading from OL Connect 1.7

It is highly recommended that you update the **Objectif Lune Update Client** before upgrading OL Connect from version 1.7 to version 1.8.

If you do not update the Update Client, an unexpected error might occur whilst updating Connect. This error does ***not*** prevent the successful upgrade of Connect to 1.8, even though

it appears as if it might have. To avoid potential confusion, we recommend that you first update the **Objectif Lune Update Client** before attempting to upgrade Connect from version 1.7 to version 1.8.

The Update Client will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem in running the update while the Client itself is still open. It will automatically update itself.

Upgrading from OL Connect 1.1

In order to upgrade from Connect Version 1.1 to Version 1.8 via the Update Manager, it is necessary to install a newer version of the Objectif Lune Update Client. The next time you run your current Update Client it will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem in running the update while the Client itself is still open. It will automatically update itself.

Once you have done this, OL Connect 1.8 will become available for download.

From Connect Version 1.2 onwards, the newer version of the Update Client was included with the Connect installation by default.

Installing OL Connect 1.8 and Workflow 8.8

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help, which can be accessed from the software and the installer.
- Note that the software comes with a 30 day trial license by default.

Updating stand-alone Workflow Messenger installations

If Workflow Messenger was installed stand alone with no other Workflow components installed, the Update Client will be unable to find the Messenger component and thus it will not automatically update to the Workflow 8.8 version. To get around this, download and run the Workflow 8.8 installer manually.

Templates Used in Workflow

For improved performance we recommend re-saving Workflow templates set up in the previous versions to run with OL Connect 1.8\Workflow 8.8.

Reduced Memory Version

It is possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

NOTE: This is *not* recommended for production.

Connect 1.8 General Enhancements and Fixes

Native support for Microsoft Excel spreadsheet files

The Connect DataMapper can now handle Microsoft Excel files natively. The CSV data type has been enhanced to automatically recognize *.XLS and *.XLSX files and use them directly without any additional steps. Both the *CSV Wizard* and the *Add Data* options now allow you to pick these file types. You can even use different types of sample files regardless of their extension, as long as their formats match the data mapper configuration. (SHARED-58610)

Conditional Print Sections Wizard

A wizard has been created to simplify the process of creating conditional print sections. Simply right mouse click a print section in the *Resource* panel and choose *Make Conditional...*
The wizard allows you to enter a basic condition based on a data field value and either *skip* or *print* the section. When the specified condition is *true* the selected *Action* is applied (which can be Print or Skip) otherwise the opposite is in effect. The icon of the section in the *Resources* panel identifies if the section is printed or skipped. Use the *Expand* option to reveal the underlying *Control* Script. (SHARED-47661)

Simplified Web Font support

Online font resources (such as Google Fonts) allow you to use their fonts for both commercial and non-commercial projects. Using these online fonts in your documents is a matter of linking to the Stylesheet file hosted by these services. Simply create a *Remote Stylesheet* entry and paste the location of Stylesheet file in the *URL* field. This will make the font available to the application. It will be added to the Fonts menu and can be used in your custom CSS Stylesheets, via the `font-family` property. (SHARED-56637).

Dynamically set Media Background Images

Support has been added for dynamically setting the path of media backgrounds at run time (aka Virtual Stationery). This is achieved via the Control Script API. The path can be

set to an image in the *Images* folder but also to a file on disk (the http:// and https:// protocols are not currently supported). This greatly simplifies template management in situations where a design is shared between different brands. This technique can also be used to dynamically set the stationery image for the preview template in Connect Send environments. (SHARED-53522)

```
var myMedia = merge.template.media["Media 1"];
myMedia.stationery.front.enabled = true;
myMedia.stationery.front.url = "file://C:/letterhead.pdf";
```

Page Breaks inside Lists

Support has been added to allow the splitting of lists across pages. This includes *Widows* and *Orphans* control for ordered (`<ol>`) and unordered (`<ul>`) lists.
The *Orphans* CSS property specifies the minimum number of lines at the bottom of a page and the *Widows* property specifies the minimum number of lines after the page break. To prevent page breaks inside these elements simply add `page-break-inside: avoid;` to your stylesheets. A formatting dialog for these elements will be added in a future version. (SHARED-14092)

Installer improvements

- Improved error capture, handling and messaging. (SHARED-40209)
- Significantly improved logging of Server Service installation. (SHARED-50796)

Korean Language Support

OL Connect 1.8 is now available in Korean, in addition to the other languages already supported. Korean is not yet available in Workflow, however. (SHARED-40161)

Context Sensitive Help

Context Sensitive Help has been added to OL Connect. Selecting Help (via button, or `F1`) in dialog boxes or screens will now generally take you to the Help page most closely associated with the calling dialog box or screen. Context sensitivity will continue to be incrementally introduced and improved hereafter. (SHARED-45766)

Database connection deadlocks resolved

We have added a timeout period to all Connect back-end database connectors, as well as increasing the amount of database connection threads to better match the capabilities of the hardware. This greatly reduces the chances of database deadlocks or bottlenecks when

processing jobs through either Connect directly or through Workflow. (SHARED-57240/57252)

Improved error handling of Merge Engine errors

In some rare circumstances XPCOM initializations would fail in Merge Engines. Thereafter the Merge Engine would continue to run but would be unable to process any further requests. Additional Merge engines would then be launched but the originals did not shut down, eventually leading to resource shortages and subsequent job failures. This issue has been addressed and XPCOM initialization errors in Merge Engines now cause the Merge engine to terminate and restart cleanly. (SHARED-57270)

Anchored positioned boxes losing style attributes

Absolute positioned (Anchored) elements would lose some style attributes under certain circumstances. These issues would only occur when the absolute positioned element had multiple style attributes that ended with the text "*top*" or "*left*". Such as is the case with "*padding-top*" and "*top*". If both those attributes were set, then only one of the attributes would be retained. (SHARED-57361)

- Customers upgrading from 1.6.1 to 1.8 will not experience any issues with their templates.
- Customers upgrading from 1.7.1 or 1.7.2 to 1.8 will experience problems only if they have saved their templates within 1.7.1. or 1.7.2, and only if those templates contained absolute positioned objects with specific inline CSS styles that end with **top** or **left**, such as **padding-top**, **padding-left**, **border-top** etc.
  In that case those specific styles will be gone and they will either need to restore a backup from before 1.7.x of those templates or manually set the styles again in 1.8 and save the templates.

IPDS Data exception

In some instances IPDS output would fail, due to data records exceeding maximum limits and continuing in the subsequent record. This scenario is now catered for and the error is no longer encountered. (SHARED-57723)

Issues with 2D Barcodes on Nipson N7000 printers

2D Barcodes (such as DataMatrix) were corrupted when printing on Nipson N7000 printers. This error has been fixed. (SHARED-57767)

Changes made to Output Speeds in Connect 1.8

A speed throttling issue was discovered that allowed some users to exceed license limitations. This issue has been corrected, and output speeds will now more accurately reflect license speeds.

Connect 1.8 Performance Related Enhancements and Fixes

Faster Performance Tweaking in Server Configuration

When tweaking performance it can be hard to figure out the right settings for the number of Merge engines, Weaver engines, dividing speed units, etc. Having to restart the Connect Server to apply the changes every time a setting was changed also made tweaking performance harder than it should have been. So we've made some improvements in order that changing scheduling settings no longer requires a Server restart.

When you hit *Apply* or *OK*, the Connect Server will pick up your new settings. Engines are only stopped or started as needed, so the impact is minimal. Changes can even be made while jobs are running, without the jobs being interrupted.

This greatly reduces the time and effort required for optimizing performance. (SHARED-53222)

> In the case that all engines are occupied, some changes will only take effect when the job finishes.

Connect 1.8 Designer Enhancements and Fixes

Automatically Fit Text to Container (Copy Fit)

The Designer can now automatically scale text content to fit the boundaries of a box (inline or absolute positioned `<div>`). Scaling text to fit a container is a very popular feature when creating personalized post cards and the like.

The option is found in the *Content* tab of the *Box* properties dialog and can be set to scale all text or a specific element in that box by entering a CSS selector. (SHARED-37702)

Rotate Print Sections individually

An option has been added to rotate individual *Print Section* orientations via the *Sheet Configuration* dialog (SHARED-46086)

Snap Guidelines to Ruler

Dragging a guide whilst holding the Shift key will snap the guideline to the closest mark on the ruler. (SHARED-54465)

Toggle Comments On/Off via Shortcut Keys

Toggle comments off or on in HTML, CSS and JavaScript editors via a keyboard combination. Use `Ctrl + /` to comment out a single line and `Ctrl + Shift + /` to comment out multiple lines. (SHARED-56440)

Refresh View button added

You can now refresh the contents of both Design and Preview views via the new Refresh button or new Refresh selection in the Menu. (SHARED-55616)

Specify Page Range for Preflight

You can now optionally perform Preflights on a range of records. (SHARED-35076)

Select and adjust multiple Box elements simultaneously

Multiple Box elements can now be selected at the same time.

Once selected you can either move or resize the selected boxes as a group. You can move them either via the mouse, or by nudging them around a single pixel at a time with the arrow keys. When nudging, the boxes will not snap to guides. Otherwise, when moving or resizing multiple boxes, the box that was originally clicked (the reference box) will snap to guides if *Snap to Guides* is enabled. (SHARED-55636)

Specify/change name of Email Attachments

Ability added to overwrite the file name for email attachments through scripting. (SHARED-57120)

Set Template Locale

Previously Connect always assigned the *System Locale* to new templates. A new option has been added to the preferences to allow the selection of a specific locale. This selection will then apply to all new templates thereafter. It applies to Date and Time fields plus numeric and currency data fields. For example, a monetary data value in France (locale fr-FR) would apply the € (Euro) currency symbol and use the ',' (comma) as the decimal separator, whilst the same monetary data value in the US locale would apply the $ (Dollar) symbol, and use the '.' (full stop) as the decimal separator. (SHARED-56791)

Locales can still be changed in individual templates by the **Edit > Locale ...** option.

General Designer improvements

- **GS1 Datamatrix** barcode now supported. (SHARED-55999)
- Section, Media, and Master resources can now be duplicated by copy-pasting. (SHARED-52261)
- **Reopening a template** will put the focus on the section that was active when the template was closed. (SHARED-53199)
- **Keyboard shortcuts** to increase (`Ctrl + Shift + >`) or decrease (`Ctrl + Shift + <`) text size now work as expected. (SHARED-11660)
- "Problem" view renamed more accurately as "**Preflight Result**". (SHARED-56343)
- Added "**folding**" support for CSS editors and source editors for standalone HTML files. (SHARED-10717)
- Multiple values now allowed in the **Make Conditional** script wizard. (SHARED-57029)
- **Default colour** swatches can now be edited or overwritten. (SHARED-53670)
- Minor Designer interface inconsistencies fixed. (SHARED-54114)

Connect 1.8 DataMapping Enhancements and Fixes

Extracting Variables without using JavaScript

Variables are frequently used in data mapping configurations as counters or as a way to concatenate values before extracting the final result to a data model field. Until Connect 1.8, the only way to extract those variables would be to create a Javascript-mode field and to use the appropriate API syntax (e.g. `automation.jobinfo.jobinfo1`, or `sourceRecord.properties.MyVariable`). We've now made it much easier by providing a pick list of all available properties in the **Extract Step**. No JavaScript required!. (SHARED-54139)

"Extract all from here" feature added

In TXT mode, we've often had the request to be able to extract everything after a specific location, regardless of the record length. This would be useful, for instance, when extracting the body of an email which is never fixed-length. Until OL Connect 1.8, the customary way of proceeding would be to use a Loop step that stores all lines in a variable and an Extract step after the loop to store that variable into a field. Not the most elegant method! A new feature allows you to specify that a field should extract everything from the current location down to the end of the current record. No loops, no variables required. (SHARED-54137)

Improved Management of Sample Data

Support added for copy-and-pasting files from Windows Explorer to the Data Samples list, and vice-versa. This allows the quick exporting of those files without having to open the data mapping configuration. It also allows you to quickly add files to the configuration without having to use the Browse button.

In addition, you can now multi-select files in the Data Samples list and delete them at once, which saves you time when you want to remove all these test files from the configuration before sending it to production. (SHARED-43317/51800)

Improvements made for Transactional Style Datafiles.

- The DataMapper and Output have both been enhanced to better cater for extremely large data records (where thousands of details might be associated with a single record). (SHARED-51691).
- The Designer has been updated to allow a limit to the maximum number of records to display in preview.
- Tooltip now displays a warning if detail table exceeds the preview limit. (SHARED-58212)

Minor DataMapper Enhancements and Fixes

- Several other small improvements made to DataMapper interface. (SHARED-57347/33912)

Connect 1.8 Output Enhancements and Fixes

Dynamic Finishing

Print Output Finishing has been improved considerably, and is now much more powerful and flexible. In Templates, you could already set finishing for documents and sections. Job Creation would allow you to specify a different kind of Finishing for your documents and

Templates. This has been extended to allow Finishing settings on all levels of Job Creation: **Document Sets**, **Job Segments** and **Jobs**. (SHARED-53277)

So now you can staple document sets and punch holes in your segments. You can also have multiple finishing settings at the same level.

The reason we call this feature ***dynamic*** finishing, is that it includes a brand new rules editor to allow you to choose when to apply a finishing setting:

PDF Pass-through

Connect's output creation (Weaver engine) tries to write content the best way possible, depending on the chosen output format and optimization settings. However, there are cases when this might not be desired, such as when the graphics have already been optimized for the device and you do not want the software to change them.

It is now possible to instruct output creation to include PDF resources in the output file *as-is*. When used, it guarantees that the fidelity of PDF graphics used in a template is retained in the output. The resulting output will be less optimized, typically producing somewhat larger files. This option can also be useful when the output is showing unexpected results or to prevent rasterization of PDF output. Will this feature trigger visible differences in the output? No, in most cases not, but when printing highly optimized graphics, expect to see a slight difference in the printed output. This is only possible when PDF content is allowed in the output, meaning it can be used with PDF, IPDS and AFP output with PDF containers. For PDF output, this feature can be found in the Output Preset or Print Wizard page under **PDF Options**. (SHARED-56412)

For AFP and IPDS with PDF containers, this option needs to be set via a Printer Definition file. Printer Definition files for AFP and IPDS that have PDF pass-through enabled will be made available for download, and will be accessible from the Connect Welcome page.

Please note that when using a PDF from a Data Mapping in combination with Virtual Stationery, the PDF is not passed through when selecting this option in Connect 1.8. This is a known issue and will be addressed in a later release of OL Connect.

A second issue when using a PDF background via the Datamapper is that the resultant PDF output file *may* contain invalid font resources. Whilst the output can be viewed in Adobe Acrobat Reader without issue and will print correctly on many printers, it will prompt warnings in Adobe Acrobat Professional's Preflight report and it should not be used as input for Connect Data Mapping. We recommend testing the output on your specific printer(s) to

best determine whether this will be an issue on your specific printer(s) or not. This issue will also be addressed in a later release.

Overprint for Spot Colours

Overprinting certain content on top of other content is sometimes required. (SHARED-56743) For example:

- To deal with special print applications, such as applying special (invisible) inks that are intended to go on top of coloured areas, for instance printing UV ink or applying varnish to a certain area.
- To avoid mis-registration when printing black on top of coloured areas.

To support scenarios like these, Connect now supports Overprint for Spot Colours.

Note: Overprinting does not show on-screen in the Designer.

Print Output

- **Mixplex support added**
  An option has been introduced to omit empty back sides for the *Single* and *Last* sheet positions when Duplex is enabled, resulting in mixplex output. This helps in reducing costs in printing environments where page count or click-charging is applied. (SHARED-46965\55459)
- **Improved logging** of output generation. (SHARED-53367)
- **Continuous Printing on IPDS printers now supported**
  Option added to the Add/Edit Printer dialog, to provide support for continuous printing on IPDS printers.
  For further details, please see **"Continuous Printing on IPDS Printers" on the facing page**.

Connect 1.8 Print Manager Enhancements and Fixes

Continuous Printing on IPDS Printers

When printing on high volume IPDS printers, it is best if the printer does not pause often. On continuous feed printers in particular, a pause can lead to malfunctions as well as cause delays. It's not just the pause itself, but also the additional time spent whilst the printer slows down and then speeds up again after the pause. The faster the printer, the bigger an issue this becomes.

The Connect Print Manager itself was increasing the pause time as it had to make sure that the currently printing job was completed before sending a new job.

To reduce the need for such pauses, support for **IPDS output incorporating multipage PDF containers** was added in Connect 1.8. Most (if not all) modern IPDS printers support Multipage PDF containers, which can be used in such a way to as remove pauses between print jobs. This means high-end printers can now be prevented from pausing.

To cater for IPDS output with Multipage PDF containers, the Print Manager has been significantly updated.

For further information about this, please see the Print Manager section in the online Help. (SHARED-56881)

General Print Manager improvements and fixes

- The entire row is now highlighted when selected in the SNMP printer status view of the Print Manager. (SHARED-41114)
- Can now drag print jobs back and forth between the Printer Queue and the Job Queue.
- A multitude of small GUI improvements were made, including catering for new continuous IPDS printing.

Capture OnTheGo (COTG) Enhancements and Fixes

New and improved COTG library

A new **jQuery** plugin variant of the COTG.js library introduces *events* and *options* for COTG widgets. These concepts greatly simplify event based programming. For example: it allows your code to set a date for a date field and retrieve the geolocation automatically on the drawing of a signature.

The COTG jQuery plugin is the successor of the *cotg-1.x.js* JavaScript library found in COTG forms based on the COTG Starter templates (v1.x). The main purpose of the COTG library is to link the COTG specific form controls with the hardware features of your mobile device. It also implements special controls like the **Fields Table** and makes field names unique for each row in a detail table. (SHARED-44058)

Other COTG Improvements

- COTG Server selection added to **Send COTG Test** dialog. (SHARED-30002)
- **Unchecked checkboxes** now automatically submit a value of '0' (zero). (SHARED-50655)
- Add support for inserting form inputs and COTG inputs in the **Snippet editor** even when a <form> element is missing. (SHARED-55885)
- The Send COTG Test dialog now allows submission of **blank forms** for testing purposes. (SHARED-56001)
- The Camera Properties dialog has a new option to **enable/disable time stamping**. (SHARED-56991)
- The Geolocation Properties dialog now includes an option to **enable/disable the "High Accuracy" flag**. (SHARED-57004)

Workflow 8.8 Enhancements and Fixes

Deleting documents from the COTG repository

Many COTG customers create documents for which they don't want to set an expiry date (or a lifespan). They therefore set an expiry date far into the future to make sure the document remains in the repository. However, once the document has been filled they would like to remove it from the repository as soon as possible so it doesn't clutter the view of the repository. Accordingly, a new Workflow task now allows deletion of a document from the COTG repository, using just the ID. The ID is obtained when the document is first sent to the COTG server and could conceivably be stored in Workflow's Data Repository until you need it to delete the document. The task can be used either as a standard Action Task or as a Condition Task that is set to True when the deletion is successful. (SHARED-55313)

Accessing the Connect Managed File Store

Sometimes you need a central location for storing and retrieving files. Some of those files might only be required for a brief period of time while others may have to be stored for longer periods. The usual approach is to create a folder somewhere on the Workflow system and use it as a file repository. But this solution is neither portable nor completely

centralized since it relies on the characteristics of the specific system on which it is implemented.

The Connect Server has its own File Store which it uses for transient files. This File Store is managed by the Cleanup service who takes care of removing obsolete files when those files are not marked as permanent. This greatly reduces the amount of administration required to manage the files. We figured that since there is already a File Store and REST API calls to connect to it, why not turn that into an accessible feature for customer implementations?

With Connect 1.8, the Workflow 8.8 tool implements three new tasks that allow you to **Upload**, **Download** and **Delete** files in the Connect File Store.

Files can be marked as permanent, in which case the Cleanup service won't touch them. The files can still be accessed through the REST API, which means web portals could potentially access the files directly without having to go through a Workflow process. The feature is therefore centralized and portable as it does not rely on any specific location or folder. (SHARED-57617)

Generating all contents using JSON instead of data records

A number of customer workflows involve generating new versions of documents based on the original one created by Connect. For instance, a Delivery Note might come back with adjusted quantities on certain line items. Or a web-based status page might need to get updated with additional data. However in many cases, the original data must still be left untouched (for reprints, for instance).

The traditional approach to recreating the document would be to generate a new single data record with the new data and perform the usual data mapping/content creation/output creation operations. However, that means the data records are duplicated in the database and the entire operation takes slightly longer than it could.

As of Connect Workflow 8.8 all *Connect Create Content* tasks have the option of using a JSON string as their data source instead of relying on the metadata generated from a data mapping operation.

This feature builds on the same kind of improvement that was implemented in Connect Workflow 8.7 with the *Create PDF Preview* task that provided lightning fast PDF Creation. The new feature makes it much easier to create Web or Email status pages that are delivered almost instantly. (SHARED-51086/55414)

Converting XML to JSON and JSON to XML

XML is already one of the most popular data formats used with Connect. In many implementations, most notably web-based ones, XML is used to transfer or update information back and forth between Connect and Workflow. However, using XML means the DataMapper must be involved each time the data changes, which impacts performance, especially when all you want is to update a Status web page. With the new functionalities added to all *Content Creation* tasks allowing them to receive JSON as the data container, it is only natural that Workflow provide an easy way of converting data between JSON and XML formats.

The new task is simple to use: pick a destination format (XML or JSON) and the job file is immediately converted to that format. No scripting required! (SHARED-51089)

> The JSON data can be used in other settings as well, not just for creating Connect content. For instance, converting a HTTP Request to JSON would allow a script to easily turn that JSON into an object and manipulate its properties with much more readable syntax than using ExpandString() on data selections.

Full Timestamp entries added to Data Repository

Added full timestamps to the Repository to allow for more precise information on each key set. The *DateC* and *DateM* keys now both contain a full time stamp in the form of: *YYYY-MM-DDThh:mm:ss.sZ*.
Please consult the online Help for a full description of the Timestamp. (SHARED-52160)

General Workflow fixes and enhancements

- Additional **Tooltips** added to interface. (SHARED-54981)
- Improved support for multiple IDs in **Retrieve items** task. (SHARED-53740)
- Repository access improvement and memory leaks fixed. (SHARED-56424)

Known Issues

Memory Usage in Clustered Environments

In high speed clustered environments, OL Connect will fill the database faster than the Clean-up service can clear it (the ratio is approximately 3:1). Therefore if the system needs to run for some time under a considerable load, a larger database should be used to prevent production from being interrupted.

We also recommend only the Master Server performing the database clean-up, or the local

Server Extension if the database is not installed with the Master, to maximize performance. For **all** Designers, Server and Server Extensions except one, set the cleanup of the database entities to run every 100 months and disable the cleanup for any Designer which also has a Server or Server Extension on that machine.
Note that if the Designer is a stand-alone then it is not connected to the central database for clean-up purposes.

Issues with Microsoft Edge browser

The Microsoft Edge browser fails to display web pages when the Workflow's CORS option (in the HTTP Server Input 2 section) is set to "*". This issue will be resolved in a future release.

Worklfow - "Execute Data Mapping" - Issues with mutliple PDFs

If a process has a *Folder Capture* step (but not the first input step) to capture multiple PDFs within a folder, followed by a *Execute Data Mapping* step to extract XML files for each PDF, only the first PDF file will be processed. The workaround is to put the Extract Datamapper in a Branch, just after the Folder Input. This issue will be fixed in a later release. (SHARED-59752)

Installation Paths with Multi-Byte Characters

When installing the Chinese (Traditional or Simplified) or Japanese versions of Connect, if the user specifies an alternative installation path containing multi-byte/wide-char characters it can break some of the links to the Connect-related shortcuts in the Start Menu and cause an error to appear at the end of the installer. The workaround for the moment is to use the default installation path. The problem will be addressed in a later release.

Switching Languages

Changing the language using the **Window>Preferences > Language Setting** menu option does not currently change all of the strings in the application to the selected language. This is a known issue and will be fixed in a later release.

In the meantime we offer the following workaround for anyone who needs to change the language:

1. Go to the .ini files for the Designer and Server Config:
    - C:\Program Files\Objectif Lune\OL Connect\Connect Designer\Designer.ini
    - C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.ini

2. Change the language parameter to the required one under Duser.language=en | es | de | fr | it | ja | ko | pt | tw | zh

Only one of the above language tags should be selected. Once saved, Connect will appear in the selected language at next start-up.

GoDaddy Certificates

When installing Connect offline, dialogs allow installing the GoDaddy certificates. Most users should use the default settings and click **Next**. In some cases, however, this may not work correctly. For this reason those users should activate **Place all certificates in the following store** and then select the **Trusted Root Certification Authorities** as the target certificate store.

MySQL Compatibility

After installing Connect 1.8 a downgrade to a Connect version earlier than Connect 1.3 or to a MySQL version earlier than 5.6.25 is not seamlessly possible. This is because the database model used in Connect 1.3 and later (MySQL 5.6) is different to that used in earlier versions. If you need to switch to an older version of Connect / MySQL, it is first necessary to remove the Connect MySQL Database folder from "%ProgramData%\Connect\MySQL\data" before installing the older version.

PostScript Print Presets

The print presets for PostScript were changed from Version 1.1 onwards meaning that some presets created in Version 1.0 or 1.0.1 may no longer work.

Any PostScript print preset from Version 1.0 that contains the following will not work in Version 1.8: *.all[0].*

Any preset containing this code will need to be recreated in Version 1.8.

Available Printer Models

Note that only the single Printer Model (Generic PDF) will appear on the **Advanced** page of the **Print Wizard** by default.

To add additional printer models click on the settings button next to the Model selection entry box.

External Resources in Connect

There are certain limitations on how external resources can be used in Connect. For example if you want to link a file (e.g., CSS, image, JavaScript etc.) from a location on the network but you do not want to have a copy of the file saved with the template you need to do the following:

1. The resource must be located where it can be accessed by all Servers/Clients run as users. Failure to do this will cause the image to appear as a Red X in the output for all documents which were merged by engines which could not access the file. The job will terminate normally and the error will be logged.
2. The file must be referenced via a UNC path e.g.,

   file:////w2k8r2envan/z%20images/Picture/Supported/JPG/AB004763.jpg

   - UNC paths are required because the services will be unable to access mapped network drives (Windows security feature).
   - The engine processing the job will look on the local file system for the direct file path leading to the "resource not found" issue mentioned above.

Important Note: The Designer itself and Proof Print do not use processes that run as services and they may find local files with non-UNC paths which can lead to the false impression that the resources are correct.

Using Capture After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, Capture can no longer be used within Workflow 7. The plugins are now registered uniquely to Workflow 8 and the Messenger for Workflow 7 is taken offline. It is only possible to use Capture from OL Connect Workflow 8 thereafter.

Capturing Spool Files After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, the PlanetPress Suite 7 option to capture spool files from printer queues will no longer function. The solution is to use OL Connect Workflow 8 to capture spool files from printer queues.

Colour Model in Stylesheets

The colour model of colours defined in a stylesheet can sometimes change after editing the stylesheet. This is a known issue and will be addressed in a subsequent release.

Image Preview in Designer

If in the Windows Internet settings (**Connection Settings > LAN configuration**) a proxy is enabled, but "Bypass proxy settings for local addresses" is not checked, the image preview service, conversion service and live preview tab in the Designer will not work and exhibit the following issues:

- Images will be shown as 0 size boxes (no red 'X' is displayed)
- Live preview does not progress, and when re-activated reports "browsers is busy"

To fix the issue you must check the "Bypass proxy settings for local addresses" option.

Merge\Weaver Engines when Printing

The print operation in the Designer will automatically detect whether the Merge\Weaver engines are available and display a message for the user to retry or cancel if not. Once the Merge\Weaver engine becomes available and the user presses retry the print operation will proceed as normal. This message can also occur in the following circumstances:

- If the server is offline and you are not using Proof Print
- On some occasions before the Print Wizard opens

REST Calls for Remote Services

The Server will now accept REST calls for all remote services and will make commands wait indefinitely until the required engines become available. The Server will log when it is waiting for an engine and when it becomes available. Note that there is no way to cancel any commands other than stopping the Server.

Print Content and Email Content in Workflow

In Workflow's Print Content and Email Content tasks, the option to Update Records from Metadata will only work for fields whose data type is set to String in the data model. Fields of other types will not be updated in the database and no error will be raised. This will be fixed in a later release.

Print Limitations when the Output Server is located on a different machine

The following limitation may occur when using the Print options from a Designer located on a different machine to the Output Server:

- The file path for the prompt and directory output modes is evaluated on both the client AND server side. When printing to a network share it must be available to BOTH

the Designer and Server for the job to terminate successfully.

- The Windows printer must be installed on both the Server and Designer machines.
- When printing via the Server from a remote Designer, the output file remains on the Server machine. This is remedied by selecting "Output Local" in the Output Creation configuration.

VIPP Output

Some templates set up with landscape orientation are being produced as portrait in VIPP. It can also sometimes be the case that text and images can be slightly displaced. These are known issues and will be addressed in a later release of Connect.

OL Connect Release Notes 1.7.1

Overview

This document provides an overview of the new features and enhancements in OL Connect 1.7.1 and Workflow 8.7.

Installing OL Connect 1.7.1 and Workflow 8.7

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help in the installer.
- Note that the software comes with a 30 day trial license by default.

Upgrading from OL Connect 1.1

In order to upgrade from Connect Version 1.1 to Version 1.7.1 via the Update Manager, it is necessary to install a new version of the Objectif Lune Update Client. The next time you run your current Update Client it will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem to run the update while the Client is open. It will automatically update itself.

Once you have done this, OL Connect 1.7.1 will become available for download.

From Connect Version 1.2 onwards, the newer version of the Update Client was included with the Connect installation.

Updating stand-alone Workflow Messenger installations

If Workflow Messenger were installed stand alone, with no other Workflow components installed, the Update Client will be unable to find the Messenger component and thus it will not automatically update to the Workflow 8.7 version. To get around this, download and run the Workflow 8.7 installer manually.

Templates Used in Workflow

For improved performance we recommend re-saving Workflow templates set up in the previous versions to run with OL Connect 1.7.1\Workflow 8.7.

Reduced Memory Version

> This is **not** recommended for production.

It is now possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

Connect 1.7.1 General Enhancements and Fixes

Template Reports added to Connect

Generate a report in PDF format containing the most important information about your template. The report lists contexts, sections, master pages, scripts, the data model, graphic files, and any other resources used, along with their properties. This report can be added as part of your project documentation.

The report is created using Connect technology, and it generates an XML file and thumbnails, allowing you to create your own custom report structure and corporate styling. This can be achieved by altering the underlying Connect Template and DataMapper configuration.

Document Properties

Document Properties can now be added to both Templates and DataMapper Configurations. This allows you to specify properties such as the document author, the customer name and other important references. You can also add custom key/value pairs. The respective properties can be retrieved in scripting and are thus available as content in your documents. The information is also included in the *Template Report* feature. (SHARED-47780)

Stability improvements

- Improvements made to the Clean-up service. In some production environments the database Clean-up could not keep pace with database growth, leading to the database gradually filling up. This has been fixed through an improved internal database structure and more efficient queries and deletions. (SHARED-46465/52345)
- The OL Connect server was experiencing communication problems with the engines in some circumstances, after data mapping or content creation errors were encountered. These issues have now been resolved. (SHARED-55165)

"Enhance with Connect" option added for PDF files in Windows Explorer

A Windows Explorer context menu entry "**Enhance with Connect**" has been added for PDF files. When a user selects this context menu entry, OL Connect Designer opens with a prefabricated template, that uses the selected PDF file as the background. (SHARED-15350/47156)

Support added for Remote HTML and JSON Snippets

In OL Connect 1.7.1 we introduce the concept of remote snippet resources. These snippet entries have a Name and URL property (e.g., the hyperlink to the endpoint) and reside in the Snippets folder located in the Resources panel.

In scripts these snippet entries are referenced just like regular snippets, e.g., `loadhtml ('snippets/my_content.rhtml')` or `loadjson('snippets/posts.rjson')`. Note the "r" in the file extension.

Having the snippet entry in the **Snippets** folder within the **Resources** folder allows us the simplest overview of the resources used. In previous versions this behaviour would have had to be captured in script and therefore would not have been directly visible as part of the resources. This new approach greatly simplifies maintenance of the URL, as it can now simply be updated in the Resources panel rather than by browsing through all the scripts. (SHARED-42314/52591)

Handling Nested Detail Data

Simplifying the handling nested detail data has been on our agenda for some time. As part of our research into this we have looked at an approach that repeats table rows for nested detail data. This doesn't create HTML tables in HTML tables but rather clones a base row specified for each level.

At this stage there is no user interface to configure this type of dynamic table but in a separate Technical article. It can be achieved by setting some HTML attributes in the *Source* view and adding scripting to populate the cells.

A user interface (table wizard) to set things more elegantly will be introduced in a future version, along with additional functionality, such as subtotal calculations.

As we are rather excited about this approach we wanted to share the current state with you in OL Connect 1.7.1. (SHARED-43047)

Installer improvements

- The OL Connect 1.7.0 installation did not work on machines running Windows 10 build 1703 (i.e. the "Creators Update", released March 2017). This has been fixed for OL Connect 1.7.1. (SHARED-56800)

- The **silent installation** process has been enhanced, and now supports the following:
  - **Setting the repository**. This can be configured via the "*product.repository*" entry in "*install.properties*". (SHARED-17841)
  - **Selecting a dedicated locale** (language and country code) for the Connect applications. These can be configured by the "*user.language*" and "*user.country*" entries. (SHARED-18381)
  - Improved reporting of Silent Installation success or failure. (SHARED-17723)
- Microsoft SQL server connection settings added to Connect Installer. (SHARED-36866)
- The Update Client will now run the installation in the same language as the original installation. (SHARED-37868)
- Installer has been made more robust, and will now continue (with warning messages, if applicable) when it encounters any of the following scenarios:
  - If Server start-up was unsuccessful during installation. (SHARED-39398/46837)
  - If no Database connection could be established. (SHARED-39400)
  - The Installer now checks if the OL Connect MySQL service is in the proper state and resident in the expected folder. If is not, instructional warning messages are now displayed. (SHARED-40309/45431)
  - If Connect folders that should have been deleted were found upon re-installation. (SHARED-41420)
  - Passwords for running the Connect Service/s (Server and Server Extension), which are entered as part of the installation process, would cause errors in the installer if the passwords contained angled brackets ("<" or ">"). This error has been fixed and those characters are now correctly supported. (SHARED-42546)

Print Manager improvements

- A document preview has been added to the Print Manager, for IPDS, PCL and PostScript documents. (SHARED-43959)

  NOTE: The Postscript viewer requires manual installation of the free Ghostscript application.

Connect 1.7.1 Designer Enhancements and Fixes

Edit and Save CSS, HTML, JavaScript and JSON files within the Designer

Ever needed to quickly edit an external CSS, HTML, JavaScript or JSON file? The OL Connect 1.7.1 Designer now allows you to open and save these file types via the **File** menu.

(SHARED-42094)

Data Model Panel Enhancements

Various enhancements have been made to the **Data Model** panel. The browse options of the main record are now *sticky* and do not move out of view when working with a large number of data fields. An eye icon has been added to the toolbar, and is used to toggle the visibility of the *ExtraData* field. In addition, you can select and group multiple fields in order to collapse them out of view (and expand them back, obviously), which is particularly useful when dealing with large data models that force you to constantly scroll up and down to bring a specific field into view. (SHARED-45370/54106)

The Data Model panel has also been enhanced to allow alphabetical sorting of detail tables. (SHARED-47169)

Simpler Invocation of Email Script Wizards

Invoke email related script wizards simply by clicking the labels in the email information bar. (SHARED-47329)

Simplified Email fields User Interface

Create *To*, *CC*, *BCC*, *From* and *Reply To* email scripts by dragging and dropping a data field to the respective input field or type a static address directly in the input fields. (SHARED-9178)

Type the subject in the Subject email field and drag and drop data field(s) to positions in that string to make a personalized email subject without any scripting. (SHARED-51475)

Improved Customization of the Designer interface

Customize your interface by selecting your own colours for object edges, margins, guidelines, etc. (SHARED-49841)

Guideline behaviour improved

Along with visible/invisible settings, Guidelines can now be locked in place or set to snap to objects, using the new **Guides** option in the View menu. (SHARED-47159).

Warning now displayed when opening templates created in an older version

When OL Connect opens an older template file it is automatically migrated to the template structure of the current version. Saving the file in the new version would thus update the file format and prevent the document opening in an older version.

A warning is now shown when opening a Template created in an older OL Connect version, allowing you the chance to save the Template to a new file, leaving the original intact. (SHARED-51912)

Turn Warning dialogs Off/On

A "**Do not show this warning again**" check-box option has been added to many OL Connect Warning dialog boxes:

These Warnings can be switched on again at any time thereafter, via the "**Reset All Warning Dialogs**" button in the General Preferences dialog. (SHARED-16962)

Option to automatically Delete a dynamic table when the table is empty

An option has been added to allow you to automatically delete a dynamic table when the data table is empty. To do so, select the entire table, and then tick the "**Hide when empty**" checkbox in the Attributes panel. (SHARED-43537)

Replace elements with data-insert-location when inserting HTML elements

When inserting an element from an **Insert dialog**, Connect now checks the *data-insert-placeholder* attribute. The value of the attribute is then used to set the default value for the *Insert Location* option within the Insert dialog. If the attribute is not found, things behave as in previous versions.

This ticket also introduces the Replace option for the Insert Location drop down. When selected the to-be-inserted element(s) will replace the currently selected element(s). (SHARED-52369)

Scripting improvements

- Context menu added to the **Edit Script** dialog. (SHARED-45381)
- **Find and Replace** functionality has been added to Script editors. (SHARED-48424)
- New menu option to **rename** Scripts or Folders has been added to the Context Menu within the Script panel. (SHARED-48607)

- Support added for **copy and paste** of folders and scripts within the Scripts panel. (SHARED-49299)
- The JavaScript **parseInt()** method now defaults to using base 10 arithmetic rather than base 8, as defined in the ECMASCRIPT specifications. (SHARED-49010)

General Designer improvements

- **Duplicate and Delete line(s)** using shortcuts in the Stylesheet, JavaScript and HTML editors. Use `Ctrl+D` to **duplicate** and `Ctrl+Shift+D` to **delete** the currently selected lines. (SHARED-46928)
- Entering **geometry values** without stating a specific unit type will now automatically assign the default unit type to the entry. (SHARED-50656)
- When **deleting an element** (such as a Barcode or a Chart) on a page, a check will now be made for associated scripts. If any are found, the deletion step will provide an option to delete those scripts as well. (SHARED-45675)
- An "**All Files (*.*)**" filter was added to Save/Save-as dialogs. (SHARED-28237)
- **Icons** have been added for JS and CSS files in the Includes dialog to make it easier to distinguish between local and remote resources. (SHARED-47936)
- **Abs box** grippers and borders now display at a consistent thickness regardless of zoom level. (SHARED-50175)
- Support added for **dynamically setting the media background image** and its options via a Control Script. This only works for PDF files residing in the template at the moment. (SHARED-53524)

Web form improvements

- Includes (JavaScript and CSS) can now be set for the entire Web context, rather than just per single web pages. These files are automatically linked to all web page sections and act as **global includes**. This is ideal when working with framework and library files that are used by all web page sections (for example jQuery or Foundation).
  One can still add additional includes on a per web page basis, if desired. (SHARED-48708)

Capture OnTheGo (COTG) improvements

- **Two new form inputs** have been introduced to facilitate the retrieval of the document ID and the store ID. (SHARED-53987/54054)

- Improvements made to **updating the COTG library** within existing templates. The user will now be prompted as to whether they wish to switch to the new version or not. (SHARED-46920)
- Ability added to **insert dummy data** for the form inputs (including special COTG inputs such as the signature) upon retrieving the Job Data file from within the Designer. (SHARED-48676)
- Workflow "**Output to COTG**" task can now be run as an action and return Document IDs. (SHARED-48951)
- **Deskew (straighten) pictures.**
  A new option is added to the Camera Properties allowing the user of the COTG app to deskew or straighten images taken with their mobile device. Deskewing optimizes the image for further processing of the image, such as through an OCR process. (SHARED-53982)

Connect 1.7.1 DataMapping Enhancements and Fixes

DataMapper can now fetch or update data from remote sources

New in OL Connect 1.7.1 is the ability to create an XMLHTTPRequest object (aka XHR) in DataMapper scripts in order to issue REST/AJAX calls to external servers. This feature allows the datamapping process to complement the extraction process with external data, including data that could be provided by a HTTP process in Workflow.

For instance, the DataMapper could issue calls to a Workflow process that retrieves certain values from the Workflow's Data Repository.

Also, one could imagine a DataMapper postprocessor that writes the results of the extraction process to a file and then uploads that file to a Workflow process. (SHARED-43502)

> As always with powerful features like this one, you need to be careful how you design your solution. For example it is possible to create an endless circular process, whereby a Workflow process calls a data mapping process, which in turn calls the same Workflow process, which calls the DataMapper again, and so on.

Skip over Source Data Records

Another very important new feature implemented in the DataMapper is the ability to skip over some of the source data records without writing anything to the database. This allows

you to quickly filter out some records from the data source without having to extract them to the database first, while still extracting others. (SHARED-24548)

Let's say for instance that your data source contains postal addresses from many countries, but you only want to extract the data for Portugal. You can now create a condition that examines the country field to determine if the source record is for a country *other* than Portugal. If the condition is *False* (i.e. the country IS indeed Portugal), you can extract the data as per usual. But in the *True* branch of the condition (i.e. the country *is anything but* Portugal), you can now add an Action Step and specify the new "**Stop processing record**" action type. This basically discards this data record and instructs the DataMapper to immediately skip to the next source record.

This yields two immediate and major benefits:

- Data Extraction is much faster since you are only extracting the records you actually want
- The database will not be cluttered with useless records (potentially numbering in the thousands) that you were not going to use anyway. As a consequence, the automated clean-up process will have much less work to perform when the time comes to delete obsolete entries from the database. This should result in a lighter workload and better overall performance.

If you stop processing any record *after* you've already extracted some data from it, then the record will still be stored in the database, with un-extracted fields being assigned whatever default value (if any) you defined for them. So if your goal is to completely prevent unwanted records from being stored in the database, you should make sure to implement your filtering conditions early in the data mapping process.

Improved naming of default fields

The default field naming scheme has been enhanced to allow duplicate field names to be generated automatically as long as they are on different levels (or in different detail tables). This is especially useful with XML data sources where field names (e.g., ID, NAME) are often re-used through different levels of the structure.
With the new, more flexible naming scheme, the DataMapper checks for duplicates at the same level before deciding whether or not to create or increment a numeric suffix that is appended to the field name. (SHARED-42645)

Improvements made to XML File Processing

Issues were encountered with repeated nodes in XML datasets. In XML when adding an Extraction step, the XPath was not generated with an index, which resulted in only the first node being returned. This has been fixed, and repeated nodes are now catered for. (SHARED-28107/32238)

Improved support for Multi-Byte Encoding

Added support for byte based positioning in addition to the existing character based positioning for MultiByte (Big5, GB18030, UTF-8 and Shift-JIS) text files. (SHARED-53174)

General DataMapper Enhancements and Fixes

- **XML Wizard**: option added to extract Attributes and to set boundaries on Attribute changes. (SHARED-42251)
- Improved support for **UNC paths** to image files. (SHARED-44316)
- The **Extradata** fields are now available in the DataMapper to more easily allow setting of field default values. The display of the Extradata fields can be toggled on or off directly from the Data Model panel. (SHARED-51426)
- New **data.findRegExp()** function added. This function is similar to the existing data.find() function, but with Regular Expression support added. (SHARED-51694)
- Repeat Step conditions can now be set to **evaluate operands as integers or as strings** to make it easier to compare numeric values without having to cast them first. (SHARED-49786)

Connect 1.7.1 Output Enhancements and Fixes

Grouping With and Without Sorting

Sometimes the data used for generating documents is already pre-sorted, but you may still need to group documents into sets or segments. In those cases, the grouping process should not reorder the documents. This has now been implemented in OL Connect 1.7.1.

Consider the following example: data has been pre-sorted for postal sorting, which means that documents for the same customer will also be in consecutive order in the job (assuming a customer has a single postal address). If we want documents for the same customer to go into the same job, we can use grouping to create document sets and we might use the customer number for this grouping. When the customer number changes, we want a new document set to begin. If grouping by customer number also sorts by customer number, our pre-sorted order will get messed up.

The Job Creation settings have been improved to allow this kind of grouping. Sorting ascending, descending or not sorting at all can be set per field used for grouping. (SHARED-45125)

This means that, apart from straightforward cases where we are grouping with or without sorting, it is also possible to create combinations where some fields do alter the sort order and others have no effect.

Please note that grouping without sorting also means that any documents that have the same value for the same grouping field (i.e., customer number in the example above), but which are not consecutive in the input data, will not end up in the same group.

The settings for **Grouping** are available both in the **Job Creation Settings** dialog and the **Advanced** mode of the **Print Wizard**.

Progress of External Sort now displayed

When using an external sort, there was no feedback about how the external sort program was progressing. A new dialog control has now been added to Connect which displays the progress of the external sort in real time. (SHARED-53601)

Improvements made to Imposition Options dialog

The settings page for Cut and Stack Impositioning has been improved to show a sample of the chosen imposition settings.

Additionally, some settings on the Imposition Options page affect the way that booklets are created. These settings are now editable, so settings such as the gap between pages can now also be set for booklets. (SHARED-31097)

Additional Postal Services Barcodes added to Output Creation

Barcodes for postal services are excellent candidates for adding during the Output Creation steps, rather than during Content Creation.

Reasons for this include:

- They often cannot be added during Content Creation because they depend on document size (or weight) and on a sort order that is determined during Job Creation.
- They need to go in a fixed position, dependent upon the envelope window, rather than document design.

- It can be desirable to have templates independent from the postal service doing the delivery, in cases where there is a choice between postal services. This makes it relatively easy to switch to whichever service is offering the better rates.

To support these scenarios better, a number of postal service specific barcodes have been added to Output Creation, in addition to Content Creation. (SHARED-54755/54962/55046)

The new barcodes include:

- Australia Post 4 State
- KIX Code (Dutch postal service - Post.NL)
- Royal Mail (UK)
- Royal Mail 2D (UK)
- USPS IMB (US)
- USPS IMPB (US)

Some of these barcodes have specific requirements in order for them to be usable. The respective postal services provide specifications and sometimes also the tools for generating the content of these barcodes.

The checksums needed for Australia Post 4 State and IMPB are calculated automatically.

Fixed issue with Merge Engine memory usage

The Merge Engine would slow down when running some jobs that used external JavaScript files. These memory issues have now been resolved. (SHARED-47242)

Job Output Mask improvements, to simplify working with output file names

We have improved the way that output file names can be specified. A new dialog box has been added to the Print Wizard, to simplify the creation of Job Output Masks. While it is still possible to directly type a file name with placeholders in the Output File Mask box, it is now also possible to use the dialog to pick the metadata fields and other variables that can be used to create dynamic file names. (SHARED-12173)

A typical use case for using place holders in an Output File Mask is while generating PDF's for archiving purposes. This can require generating one PDF per document and often the files have to be named in a meaningful manner, by using an invoice number in the file name for instance. This requires one to define the invoice number as metadata in a Job Preset and then this metadata field can be used in the Output File Mask of the output preset. In addition, the Separation setting of the Output Preset has to be set to separate at the

document level.

So the next time you need a dynamically generated output file name like `inv-${document.metadata.InvNumber}.pdf` or `${document.metadata.ID}-${system.time,'yyyy-MMM-dd'}.${template.ext}`, you can use this dialog to help you get what you need.

For the Output Preset to know what metadata is available, you can select a Job Preset when creating or modifying an Output Preset:

In the Advanced mode of the Print Wizard this new dialog works a bit different, because the metadata can be directly edited in the same wizard instead of having to refer to a Job Preset.

Tray Mapping for Multiple Templates

For printing to a cutsheet printer, the Output Preset allows mapping of media defined in a template to trays and media known by the printer. To make it easier to use an Output Preset for multiple templates, the list of media shown on the Tray Mapping page is no longer fixed. So now it is possible to easily define a tray mapping for all media used on a certain printer. (SHARED-49357)

This doesn't mean that all these media have to be used in every job, so one might even map multiple media types to the same tray. In such cases, a Job Preset could be used to filter jobs in such a way that no conflicting tray mappings can occur within a job, as Job Presets allow filtering by media type.

Print Output

- **Improvements made to the Print Wizard**
  These include:
  - Improved usability in Inserter dialog. (SHARED-38279)
  - Data Filtering dialog usability improved. (SHARED-38281/38283)
  - Support added for manually setting both the horizontal and vertical gutter settings in Booklets (SHARED-53769)
  - The Additional Text and Barcode dialogs did not allow many of the available system fonts to be used. This issue has been fixed. (SHARED-46825)
- **Improved PDF comparison** implementation has improved output creation times. (SHARED-44097)

- **PostScript** Tray mapping configurations can now be made independent of the loaded template. (SHARED-49357)
- **IPDS and AFPDS** improvements:
    - Improved colour support. (SHARED-49554)
    - Extra image compression options added. (SHARED-51222)
- Improve output creation speed when **outputting with separation**. (SHARED-52088)
- **Soft masked images** were not handled correctly when writing to PDF/VT, causing errors. This issue has now been fixed. (SHARED-32335)

Workflow 8.7 Enhancements and Fixes

Custom Task descriptions

The Comments section of each Workflow Task can now be used as the task's description in the Workflow Configuration tool, allowing users to better document the process without having to resort to numerous Comment Tasks. (SHARED-39120)

Workflow processes can sometimes become rather complex and thus they require some documentation in order to allow subsequent users to know why they were implemented in one fashion or another. *Comment tasks* are already available but they use up some valuable real estate in Workflow processes, which sometimes adds to the clutter, rather than making things clearer.

Each task in a process also has a *Comments* section that allows you to properly document what you want, but requires you to click each task in order to view the comments associated with each (and you have to remember to display the *Task Comments* panel, which also robs you of valuable on-screen real estate).

To cure these issues in Workflow 8.7 we introduced a new checkbox located below the Comments field: *Use as step description*. Ticking this box instructs the Configuration tool to use the task's Comments as the description for the task in the Process panel, which allows you to put in more descriptive text than the default value without having to click on each Task to visualize it:

Option to bypass Record Persistence added to plugins

A new checkbox option has been added to both the **Create PDF Preview** task and the DataMapper tab of the **All-In-One** task, allowing you to specify that the data should not be stored in the database. This feature is specifically tailored for one-off jobs, to prevent data

from being written needlessly to the database. Instead, records are streamed directly into the Content Creation process for immediate merging. Turning the feature on in the All-In-Task can improve data mapping performance significantly, as well as the time required for the clean-up process.

Note that checking this option in the All-In-Task means that if you ever need the data for any reason (reprints, produce additional jobs, etc.), you will have to perform the data mapping configuration from scratch. So make sure you only tick the box for true one-off jobs. (SHARED-48956/56420)

Retrieve Items task now has a JSON Output option

The **Retrieve Items** task can now output the results of its query as a JSON string instead of storing them within the metadata. This allows easy handling of the results either through Workflow Scripting or directly in the Designer.

The Retrieve Items task is often used in circular Workflows where data that has already been extracted, output and presented to the user is sent back by the user for further processing. Think, for instance, of a Proof-Of-Delivery Workflow where the delivery person makes the recipient sign a **Capture OnTheGo** document and dynamically makes adjustments to the quantities that were actually shipped out. Both the signature and the modified values are sent back to a Workflow process whose purpose is to generate a PDF version of the document with those modified values.

With the new option to retrieve existing items in JSON format (including the detail tables), you can now retrieve the original record as a JSON object and use a simple script to update all the detail line values using those posted by the end-user. You can then provide that JSON object directly to the Create PDF Preview task without having to perform the data mapping operation once again, thus saving some valuable resources (both in terms of time and database space). (SHARED-50426)

New Create Preview PDF plugin added

A new **Create Preview PDF** plugin has been added to the Connect Workflow. The Plugin retrieves the resulting PDF from the file store and makes it available to the process as the job data file. (SHARED-47860)

Create Email Content task now has an option to test the SMTP settings

The **Create Email Content** task now has an option to test the SMTP settings entered before saving the configuration. (SHARED-44332/46165)

The "Test SMTP Settings" does not work when using TLS. This limitation will be addressed in a later release.

General Workflow fixes and enhancements

- **Improved datamapping speed** when outputting records in metadata. (SHARED-38455)
- **Improved performance** when creating metadata after Content Creation. (SHARED-47150)
- Processing a **Secure PDF** *as passthrough* through CreatePDF will retain the Security options. (SHARED-47951)
- The **Create Email Content** task now validates the format of the sender's address to prevent typos/mistakes from being saved to the configuration. (SHARED-47947)
- The **OL Connect Send Plug-In** now stores user name and IP along with Job Info. (SHARED-49421)
- The **Create Web Content** task can now be added to processes as an Output task. This helps make processes leaner and easier to understand. (SHARED-50083)
- Drop down list added to the **HTTP Server Input** task to set the application/json mime type. (SHARED-50085)
- **OL Connect Send related plugins** now all grouped together. (SHARED-50126)
- An issue has been fixed where the **wrong document** was attached to email output. This occurred when generating email output with a PDF attachment based on a print context and generating print output of that same template in a single Workflow process. (SHARED-51396)
- The **Output to SharePoint** task was hard coded to use MSXML 4, which was installed along with Workflow. This hard-coded dependency has now be removed from any task that uses MSXML so that they can now automatically use the latest version installed on the PC. (SHARED-53831)

Known Issues

Memory Usage in Clustered Environments

In high speed clustered environments, OL Connect will fill the database faster than the Clean-up service can clear it (the ratio is approximately 3:1). Therefore if the system needs to run for some time under a considerable load, a larger database should be used to prevent production from being interrupted.

We also recommend only the Master Server performing the database clean-up, or the local

Server Extension if the database is not installed with the Master, to maximize performance. For **all** Designers, Server and Server Extensions except one, set the cleanup of the database entities to run every 100 months and disable the cleanup for any Designer which also has a Server or Server Extension on that machine.
Note that if the Designer is a stand-alone then it is not connected to the central database for clean-up purposes.

Issues with Microsoft Edge browser

The Microsoft Edge browser fails to display web pages when the Workflow's CORS option (in the HTTP Server Input 2 section) is set to "*". This issue will be resolved in a future release.

Installation Paths with Multi-Byte Characters

When installing the Chinese (Traditional or Simplified) or Japanese versions of Connect, if the user specifies an alternative installation path containing multi-byte/wide-char characters it can break some of the links to the Connect-related shortcuts in the Start Menu and cause an error to appear at the end of the installer. The workaround for the moment is to use the default installation path. The problem will be addressed in a later release.

Switching Languages

Changing the language using the **Window>Preferences>Language Setting** menu option does not currently change all of the strings in the application to the selected language. This is a known issue and will be fixed in a later release.

In the meantime we offer the following workaround for anyone who needs to change the language:

1. Go to the .ini files for the Designer and Server Config:
    - C:\Program Files\Objectif Lune\OL Connect\Connect Designer\Designer.ini
    - C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.ini
2. Change the language parameter to the required one under Duser.language=en | es | de | fr | it | ja | pt | tw | zh

Only one of the above language tags should be selected. Once saved, Connect will appear in the selected language at next start-up.

GoDaddy Certificates

When installing Connect offline, dialogs allow installing the GoDaddy certificates. Most users should use the default settings and click **Next**. In some cases, however, this may not work correctly. For this reason those users should activate **Place all certificates in the following store** and then select the **Trusted Root Certification Authorities** as the target certificate store.

MySQL Compatibility

After installing Connect 1.7.1 a downgrade to a Connect version earlier than Connect 1.3 or to a MySQL version earlier than 5.6.25 is not seamlessly possible. This is because the database model used in Connect 1.3 and later (MySQL 5.6) is different to that used in earlier versions. If you need to switch to an older version of Connect / MySQL, it is first necessary to remove the Connect MySQL Database folder from "%ProgramData%\Connect\MySQL\data" before installing the older version.

PostScript Print Presets

The print presets for PostScript were changed from Version 1.1 onwards meaning that some presets created in Version 1.0 or 1.0.1 may no longer work.

Any PostScript print preset from Version 1.0 that contains the following will not work in Version 1.7.1: *.all[0].*

Any preset containing this code will need to be recreated in Version 1.7.1.

Available Printer Models

Note that only the single Printer Model (Generic PDF) will appear on the **Advanced** page of the **Print Wizard** by default.

To add additional printer models click on the settings button next to the Model selection entry box.

External Resources in Connect

There are certain limitations on how external resources can be used in Connect. For example if you want to link a file (e.g., CSS, image, JavaScript etc.) from a location on the network but you do not want to have a copy of the file saved with the template you need to do the following:

1. The resource must be located where it can be accessed by all Servers/Clients run as users. Failure to do this will cause the image to appear as a Red X in the output for all documents which were merged by engines which could not access the file. The job will terminate normally and the error will be logged.
2. The file must be referenced via a UNC path e.g.,

   file:////w2k8r2envan/z%20images/Picture/Supported/JPG/AB004763.jpg

   - UNC paths are required because the services will be unable to access mapped network drives (Windows security feature).
   - The engine processing the job will look on the local file system for the direct file path leading to the "resource not found" issue mentioned above.

Important Note: The Designer itself and Proof Print do not use processes that run as services and they may find local files with non-UNC paths which can lead to the false impression that the resources are correct.

Using Capture After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, Capture can no longer be used within Workflow 7. The plugins are now registered uniquely to Workflow 8 and the Messenger for Workflow 7 is taken offline. It is only possible to use Capture from OL Connect Workflow 8 thereafter.

Capturing Spool Files After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, the PlanetPress Suite 7 option to capture spool files from printer queues will no longer function. The solution is to use OL Connect Workflow 8 to capture spool files from printer queues.

Colour Model in Stylesheets

The colour model of colours defined in a stylesheet can sometimes change after editing the stylesheet. This is a known issue and will be addressed in a subsequent release.

Online Help Links Point to Introductory Page

Context sensitivity for the online help is not yet enabled in Connect. All links and F1 calls point to the introductory page, where you can Search on keywords to bring up Help pages relating to the topic.

Context sensitivity will be introduced in a subsequent release of Connect.

Image Preview in Designer

If in the Windows Internet settings (**Connection Settings > LAN configuration**) a proxy is enabled, but "Bypass proxy settings for local addresses" is not checked, the image preview service, conversion service and live preview tab in the Designer will not work and exhibit the following issues:

- Images will be shown as 0 size boxes (no red 'X' is displayed)
- Live preview does not progress, and when re-activated reports "browsers is busy"

To fix the issue you must check the "Bypass proxy settings for local addresses" option.

Merge\Weaver Engines when Printing

The print operation in the Designer will automatically detect whether the Merge\Weaver engines are available and display a message for the user to retry or cancel if not. Once the Merge\Weaver engine becomes available and the user presses retry the print operation will proceed as normal. This message can also occur in the following circumstances:

- If the server is offline and you are not using Proof Print
- On some occasions before the Print Wizard opens

REST Calls for Remote Services

The Server will now accept REST calls for all remote services and will make commands wait indefinitely until the required engines become available. The Server will log when it is waiting for an engine and when it becomes available. Note that there is no way to cancel any commands other than stopping the Server.

Print Content and Email Content in Workflow

In Workflow's Print Content and Email Content tasks, the option to Update Records from Metadata will only work for fields whose data type is set to String in the data model. Fields of other types will not be updated in the database and no error will be raised. This will be fixed in a later release.

Print Limitations when the Output Server is located on a different machine

The following limitation may occur when using the Print options from a Designer located on a different machine to the Output Server:

- The file path for the prompt and directory output modes is evaluated on both the client AND server side. When printing to a network share it must be available to BOTH

the Designer and Server for the job to terminate successfully.

- The Windows printer must be installed on both the Server and Designer machines.
- When printing via the Server from a remote Designer, the output file remains on the Server machine. This is remedied by selecting “Output Local” in the Output Creation configuration.

VIPP Output

Some templates set up with landscape orientation are being produced as portrait in VIPP. It can also sometimes be the case that text and images can be slightly displaced. These are known issues and will be addressed in a later release of Connect.

OL Connect Release Notes 1.6.1

Overview

This document provides an overview of the new features and enhancements in OL Connect 1.6.1 and Workflow 8.6.

The major focus of Connect 1.6 has been to improve performance and increase stability, as well as launching a new option for the PReS Connect and PlanetPress Connect brands called **OL Connect Send**.
A description of the the new **OL Connect Send** can be found here: "OL Connect Send" on the facing page.

Installing OL Connect 1.6.1 and Workflow 8.6

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help in the installer.
- Note that the software comes with a 30 day trial license by default.

Updating from OL Connect 1.1

In order to upgrade from Connect Version 1.1 to Version 1.6.1 via the Update Manager it is necessary to install a new version of the Objectif Lune Update Client. The next time you run your current Update Client it will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem to run the update while the Client is open. It will automatically update itself.

Once you have done this, OL Connect 1.6.1 will become available for download.

From Connect Version 1.2.0 onwards, the newer version of the Update Client was included with the Connect installation.

Updating stand-alone Workflow Messenger installations

If Workflow Messenger were installed stand alone, with no other Workflow components installed, the Update Client cannot find the Messenger component and thus it will not automatically update the component to the Workflow 8.6 version of Messenger. To get around this, download and run the Workflow 8.6 installer manually.

Templates Used in Workflow

For improved performance we recommend re-saving Workflow templates set up in the previous versions to run with OL Connect1.6.1\Workflow 8.6.

Updating Connect 1.5 installations using Microsoft SQL Server as back-end

If a Microsoft Server and MySQL were installed with Connect 1.5, and the Server Configuration Tool used to switch the back-end database to Microsoft SQL, then an extra step is required in the Update to Connect 1.6.1. The procedure is documented in the installation guide.

Reduced Memory Version

> This is **not** recommended for production.

It is now possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

OL Connect Send

OL Connect Send is an application of two components. The first is a Windows printer driver and the second is a set of Workflow plug-ins.

In its most basic form, OL Connect Send allows the transmission of print files over the Internet from any Windows Desktop application.

OL Connect Send flavors

OL Connect Send comes in three flavors. These are:

- **Free of charge**: No license required; any user; any domain; no usage limits; no web interaction.
- **User mode**: user-domain license required for each client domain; no usage limits, web interaction. OL Care at POP sold separately. Packages are available for 10, 25, 50, 100, 200 or Enterprise (Unlimited) users.
- **Credit mode**: credit license required; any user; any client (allows all domains) ; limited by credits only, web interaction. Includes OL Care at POP. Credits packages available for 20,000, 50,000, 100,000, 200,000 credits, 400,000 and 1,000,000 credits.

With the licensed version, OL Connect Send has the ability to requests a web page, displayed in the user's browser that allows them to enter job specific information. The information from this web page tells Workflow what to do next. OL Connect Send can be used to create custom interactive workflows from a centralized location, yet has the ability to be deployed and installed very easily.

OL Connect Send Verticals and applications

OL Connect Send has many applications. These include:

- **Print-for-Pay market**: Enables the consolidation of incoming jobs for further processing, printing and mailing.
- **Insurances and Financial market**: Desktop capture over the Internet of print jobs, (remote workers and branches), for centralized processing, printing and mail.
- **Law firms**: Desktop capture over the Internet of print jobs requiring complex mail merge and stationery management.
- **Supply Chain**: Inbound document processing, such as capturing inbound invoices or POs for publication in an ECM. Print to EDI for outbound documents such as invoices.

For further information on Connect Send, please refer to the OL Connect Send website and standalone User Guide.

Connect 1.6.1 General Enhancements and Fixes

Performance improvements

- Changes to the handling of **transparency in PDF backgrounds** has not only cured some job failures, but has also led to substantial improvements to both output speeds and filesizes. (49680)
- Improved processing speed for **multiple large detail table** documents. (47252/48537)
- Improvements made to the **clean-up** processes, improving overall production speed.

- Some **memory leaks plugged**, improving overall production speed.
- Improved reliability when using **MS-SQL** as back-end database.

Further Performance improvements can be found detailed in the Output Enhancements and Fixes section.

Documentation improvements

- Help layout changed to allow **easier navigation**, and the content improved.
- Broken links within the **Welcome Screen** have been fixed. (39077/47964)

Installer improvements

- **Microsoft SQL Server** can now be setup as the back-end database during the installation process. (47546)

Print Manager improvements

- Support added for **IPDS printer status (SNMP)**. (51629)
- Improvements with **handling and deleting of overlays** for some IPDS printers. (48961/49508)
- Disable printer capabilities check when the printer is busy, to reduce connection problems with **older printers**. (51589)
- Improved support for **Xerox IPDS 495** printers. (48774)
- Improved error handling when **IPDS structural errors are encountered** in IPDS resources. The Print Manager now reports the error and ceases sending the job to the printer. (50138)
- Print Manager would hang when **unexpected errors encountered on IPDS printers**. The Print Manager now checks for fatal errors and exits the running job when one is encountered. (50135)
- Improved robustness when encountering **Communications Errors**. For example, the errors previously encountered when removing a job whilst Print Manager was retrying a connection. (49757)

Connect 1.6.1 Designer Enhancements and Fixes

General Designer improvements

- **Interface improvements** such as inclusion of icons for different types of files (js and CSS).
- Provided option to configure the **script timeout period**. (48639)

- Minor issues with non-English language **translations** fixed.
- Display issues that were sometimes encountered when changing **section background images** have been fixed.
- Issue with Virtual Stationery not being enabled when adding VS PDF in **Formal Letter Template** wizard now fixed. (47268)
- Fixed an issue in which **resizing an abs box** in some scenarios would result in an additional box being displayed. (47858)
- When **typing in a non-English input language** (such as Hebrew), the keyboard input would periodically switch to English. This issue has been fixed. (49566)
- Saving a template when in Source view would erroneously **show snippet content** within the Source view thereafter. This has been fixed. (50131)
- HEX encoding option added to **DataMatrix** barcodes. (47897)

Capture On The Go (COTG) and Web form improvements

- **Dummy data** can now be added, to help when designing COTG forms. (47835)
- Improvements made to the COTG **Kitchen Sink** template Wizard. Added Image Annotation and Fields Table controls. (47817)
- Visible container added to **Geolocation** objects.
- Added support for **blank forms**, for multiple submissions. (50483)
- Workflow "**Output to COTG**" task now returns Document ID. (50487)
- Fixed picture issues with **iOS** sometimes causing image to be displayed as a red "X". (45231)
- Fixed Geolocation coordinates on **Android** devices. (46831)

Connect 1.6.1 DataMapping Enhancements and Fixes

- Support for **Regular Expressions** added to database searches. (51694)
- Improved Datamapping process **reliability**.
- Improved data record reliability when handling **large jobs** (those in excess of 50,000 records).
- Improved **PDF extraction** avoids character duplication.
- Improved **marking of data fields** in extraction steps.
- New option added to **support multibyte** (variable length) encoded data such as Big5, GBK, UTF-8 and Shift-JIS. (46813)

- Fixed issue whereby post-function for **extracted fields containing several lines** would only replace characters on the first line. (47949)
- Fixed issue with **Identically named datafields** in different detail tables causing issues with data extraction.
- Fixed issues with **Right-To-Left text concatenation**. (48712)
- Fixed issues with **PlanetPress Suite PDFs** not working consistently within Connect. (50355)

Connect 1.6.1 Output Enhancements and Fixes

General

- Merge Engine **memory leaks** fixed. (50188)
- Improved creation of **Metadata** after content creation.
- Improved conditional content for **Email and Web output**.
- Fixed issue with PDF Pass-through jobs, whereby **hidden features of the PDF** input were being included in the output. (49373)
- Some templates containing JPG or Bitmap graphics could trigger the **flatten.exe** program (used for flattening external files) to repeatedly open, causing performance degradation or even failure. (51251)

Email Output

- Fixed issue with the **email Subject field** not being encoded properly when using characters other than Latin characters. (48781)
- To **improve privacy** certain Meta tags that were embedded in the output email HTML have been removed. These include "email-reply-to", "email-from", "sender-name" and "sender-address". (49864)
- The **Date field** was not always included in the email header. This has been fixed and the Date field should now be present in all email headers. (48706)

Print Output

- Improved processing speed for jobs involving **separation** that create multiple output files. (49167)
- Improvements made in the **conversion of PDFs** to other formats have made such conversions significantly faster and the output smaller, in PCL, AFP and IPDS output. (50140)
- Improved **page range handling** on last page.

- Improved **N-Up** positioning.
- **Improved color support** on older AFPDS and IPDS printers.
- **PDF and TIF objects** are now marked for optimization in AFPDS output
- **Vector graphic** processing improved, resulting in much faster output, and smaller file sizes. (50140)
- Issues with **AFPDS Highlight colors** have been resolved. (48397)
- Issues resolved with **missing non-Latin characters** (such as the dyet "Đ" character) in IPDS PDF Pass-through jobs. (51509)
- Errors were encountered in **Connect Master/Client environments**, when printing to a directory via a Job Output Mask that contained a directory structure. (49479)
- Fixed issue with **missing PDF backgrounds** in output from Master/Client (Server Clustered) environments. These errors were due to merge engine concurrency issues. (51551)
- If an image file were replaced on disk between runs then a subsequent **Proof printing** would display the old image, . This has been fixed. (47567)
- Merge engine no longer slows down when using **external JavaScript** files in print sections. (48447)
- Fixed "**ApplicationException: Null**" errors encountered in some PCL outputs. (50868)
- Added support for **shaded bitmap fonts** to allow printing of grey text on supported IPDS and AFPDS printers. (49702)
- Fixed issues with output to IPDS Printers that do not support **FS45**. (48967)
- Fixed issues with imported PDF graphics that were **inverted in IPDS output**. (51504)
- Conversion of certain **FS10 G4 encoded images** in IPDS output was causing Xerox 495 Printers to crash. This printer specific issue has now been fixed. (51308)
- In IPDS and AFPDS output **lines and boxes** would be drawn using "Graphics Object Content Architecture" (GOCA) processing, rather than the more efficient Write Text (WT) processing. This has now been fixed, and the more efficient process is now used. (51127)
- New PrinterDef designed for **Nipson 7000** IPDS printer, which does not support Extended OVL. (50154)
- When converting a PDF in IPDS processing, if the **final item** could not be converted it was lost. This has now been fixed. (51325)

Connect Workflow 8.6 Enhancements and Fixes

General improvements

- A multitude of changes and enhancements made to support new **OL Connect Send** functionality within Workflow.
- Support for **password protection** added to "Create PDF" task. (48380)
- **Generic Data Repository** field length extended beyond previous limit of 32 characters. (47734)
- Added new "**Create PDF Preview**" task for lightning fast creation of single record PDF. (49497)
- Minor issues with some **language translations** fixed.
- Fixed issue with **control script tags** not being set correctly in email headers, in emails sent via Workflow. (51708)
- Fixed issue with the HTTP Server Input plug-in not receiving any data when a **webform contains the *enctype="multipart/form-data"* instruction** and no files were attached to the form. (35751)
- When running a process that produces an email with a PDF attachment along with a print output in the same process, the **email attachment would be incorrect** (that of the previous run). This issue has been fixed. (51879)

Performance and Stability improvements

- Improved **Print Content Creation** speed in both stand-alone and "All-In-One" tasks. (49589)
- Improved reliability in **simultaneous (multi-threaded)** COTG uploading. (47146)
- Option added to "***All-In-One***" plug-in to disable the storing of temporary data if not needed.
  This speeds up both output and the cleanup afterward. (50713)
- Fixed issues with **inconsistent PDF output** when calling the *Get Result* method (REST API) in the *All-In-One* process. (49413)
- Fixed issue with Connect (Merge Engines) losing performance when **large numbers of smaller jobs** were processed through Workflow. (50863)
- Fixed potential out-of-memory error with **very large Workflow configuration** files. (51621)

- Fixed crashes encountered when **large numbers of data selection calls** were issued in highly threaded processes. (50569)

HTTP and SMTP Server improvements

- Support added for **cross-origin HTTP** (CORS HTTP) requests, to facilitate the development and testing of web templates. (47014)
- Added option to **specify SMTP port number** in "Create Email Content" task. (49887)
- Documentation on **how to retrieve attachments from SMTP input** improved. (49473)
- Fixed corruption of **large uploaded files**. (47901)
- Fixed the timeouts encountered when the option "**Do not include XML envelope**" were selected. (49186)
- Fixed the **HTTP 501 error** when receiving Authorization headers other than Basic. (48387)

Capture On The Go (COTG) improvements

- "*Output to COTG*" task now returns **Document ID**. (50487)
- Fixed issue with "*Output to COTG*" task that could sometimes cause memory corruption, requiring a restart of the services. (47804)

Known Issues

Installing OL Connect Send on a machine with Connect installed.

When OL Connect Send Plug-Ins are installed (either standalone or via a Workflow installation) on the same machine as Connect, an interference between OL Connect Send's internal Database and that of Connect may occur, which will block a browser popup on that same machine.
This issue can be fixed by applying a startup wait to the Connect Server Service.
If the issue occurs during runtime, restarting the Connect Server Service will fix the issue.

OL Connect Send issues under Microsoft Edge browser.

- Issues with using the built-in Windows 10 default "Administrator" account and the OL Connect Send Client. This is due to Windows disallowing the opening an Edge browser whilst running under this account. This blocks interaction with the printer driver.
- The Microsoft Edge browser fails to display web pages when Workflow's CORS option (in the HTTP Server Input 2 section) is set to "*". This issue will be resolved in a future release.

Disk Drive Usage in Clustered Environments

In high speed clustered environments, OL Connect will fill the database faster than the clean-up service can clear it (the ratio is approximately 3:1). Therefore if the system needs to run for some time under a considerable load, a larger database should be used to prevent production from being interrupted.

We also recommend only the Master Server performing the database clean-up or the local Server Extension if the database is not installed with the Master to improve performance. For ALL Designers, Server and Server Extensions except one, set the clean-up of the database entities to run every 100 months and disable the clean-up for any Designer which also has a Server or Server Extension on that machine.

Note that if the Designer is a stand-alone then it is not connected to the central DB for clean-up purposes.

Note that this is not an issue with fresh Connect installations, which have the correct settings by default.

Installation Paths with Multi-Byte Characters

When installing the Chinese (Traditional or Simplified) or Japanese versions of Connect, if the user specifies an alternative installation path containing multi-byte/wide-char characters it can break some of the links to the Connect-related shortcuts in the Start Menu and cause an error to appear at the end of the installer. The workaround for the moment is to use the default installation path. The problem will be addressed in a later release.

Switching Languages

Changing the language using the **Window>Preferences>Language Setting** menu option does not currently change all of the strings in the application to the selected language. This is a known issue and will be fixed in a later release.

In the meantime we offer the following workaround for anyone who needs to change the language:

1. Go to the .ini files for the Designer and Server Config:
    - C:\Program Files\Objectif Lune\OL Connect\Connect Designer\Designer.ini
    - C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.ini

2. Change the language parameter to the required one under Duser.language=en | es | de | fr | it | ja | pt | tw | zh

Only one of the above language tags should be selected. Once saved, Connect will appear in the selected language at next start-up.

GoDaddy Certificates

When installing Connect offline, dialogs allow installing the GoDaddy certificates. Most users should use the default settings and click **Next**. In some cases, however, this may not work correctly. For this reason those users should activate **Place all certificates in the following store** and then select the **Trusted Root Certification Authorities** as the target certificate store.

MySQL Compatibility

After installing Connect 1.6.1 a downgrade to a Connect version earlier than Connect 1.3 or to a MySQL version earlier than 5.6.25 is not seamlessly possible. This is because the database model used in Connect 1.3 and later (MySQL 5.6) is different to that used in earlier versions. If you need to switch to an older version of Connect / MySQL, it is first necessary to remove the Connect MySQL Database folder from "%ProgramData%\Connect\MySQL\data" before installing the older version.

PostScript Print Presets

The print presets for PostScript were changed from Version 1.1 onwards meaning that some presets created in Version 1.0 or 1.0.1 may no longer work.

Any PostScript print preset from Version 1.0 that contains the following will not work in Version 1.6.1: *.all[0].*

Any preset containing this code will need to be recreated in Version 1.6.1.

Available Printer Models

Note that only the single Printer Model (Generic PDF) will appear on the **Advanced** page of the **Print Wizard** by default.

To add additional printer models click on the settings button next to the Model selection entry box.

External Resources in Connect

There are certain limitations on how external resources can be used in Connect. For example if you want to link a file (e.g., CSS, image, JavaScript etc.) from a location on the network but you do not want to have a copy of the file saved with the template you need to do the following:

1. The resource must be located where it can be accessed by all Servers/Clients run as users. Failure to do this will cause the image to appear as a Red X in the output for all documents which were merged by engines which could not access the file. The job will terminate normally and the error will be logged.
2. The file must be referenced via a UNC path e.g.,

   file:////w2k8r2envan/z%20images/Picture/Supported/JPG/AB004763.jpg

   - UNC paths are required because the services will be unable to access mapped network drives (Windows security feature).
   - The engine processing the job will look on the local file system for the direct file path leading to the "resource not found" issue mentioned above.

Important Note: The Designer itself and Proof Print do not use processes that run as services and they may find local files with non-UNC paths which can lead to the false impression that the resources are correct.

Using Capture After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, Capture can no longer be used within Workflow 7. The plugins are now registered uniquely to Workflow 8 and the messenger for Workflow 7 is taken offline. It is only possible to use Capture from OL Connect Workflow 8 thereafter.

Capturing Spool Files After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, the PlanetPress Suite 7 option to capture spool files from printer queues will no longer function. The solution is to use OL Connect Workflow 8 to capture spool files from printer queues.

Colour Model in Stylesheets

The colour model of colours defined in a stylesheet can sometimes change after editing the stylesheet. This is a known issue and will be addressed in a subsequent release.

Online Help Links Point to Introductory Page

Context sensitivity for the online help is not yet enabled in Connect. All links and F1 calls point to the introductory page, where you can Search on keywords to bring up Help pages relating to the topic.

Context sensitivity will be introduced in a subsequent release of Connect.

Image Preview in Designer

If in the Windows Internet settings (**Connection Settings > LAN configuration**) a proxy is enabled, but "Bypass proxy settings for local addresses" is not checked, the image preview service, conversion service and live preview tab in the Designer will not work and exhibit the following issues:

- Images will be shows as 0 size boxes (no red 'X' is displayed)
- Live preview does not progress, and when re-activated reports "browsers is busy"

To fix the issue you must check the "Bypass proxy settings for local addresses" option.

Merge\Weaver Engines when Printing

The print operation in the Designer will automatically detect whether the Merge\Weaver engines are available and display a message for the user to retry or cancel if not. Once the Merge\Weaver engine becomes available and the user presses retry the print operation will proceed as normal. This message can also occur in the following circumstances:

- If the server is offline and you are not using Proof Print
- On some occasions before the Print Wizard opens

REST Calls for Remote Services

The Server will now accept REST calls for all remote services and will make commands wait indefinitely until the required engines become available. The Server will log when it is waiting for an engine and when it becomes available. Note that there is no way to cancel any commands other than stopping the Server.

Print Content and Email Content in Workflow

In Workflow's Print Content and Email Content tasks, the option to Update Records from Metadata will only work for fields whose data type is set to String in the data model. Fields of other types will not be updated in the database and no error will be raised. This will be fixed in a later release.

VIPP Output

Some templates set up with landscape orientation are being produced as portrait in VIPP. It can also sometimes be the case that text and images can be slightly displaced. These are known issues and will be addressed in a later release of Connect.

Print Limitations when the Output Server is located on a different machine

The following limitation may occur when using the Print options from a Designer located on a different machine to the Output Server:

- The file path for the prompt and directory output modes is evaluated on both the client AND server side. When printing to a network share it must be available to BOTH the Designer and Server for the job to terminate successfully.
- The Windows printer must be installed on both the Server and Designer machines.
- When printing via the Server from a remote Designer, the output file remains on the Server machine. This is remedied by selecting "Output Local" in the Output Creation configuration.

OL Connect Release Notes 1.5

Overview

This document provides an overview of the new features and enhancements in OL Connect 1.5 and Workflow 8.5.

Installing OL Connect 1.5 and Workflow 8.5

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect and Workflow can be found in the online help in the installer.
- Note that the software comes with a 30 day trial license by default.

Updating from OL Connect 1.1

In order to upgrade from Connect Version 1.1 to Version 1.5 via the Update Manager it is necessary to install a new version of the Objectif Lune Update Client. The next time you run your current Update Client it will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem to run the update while the Client is open. It will automatically update itself.

Once you have done this, OL Connect 1.5 will become available for download.

From Connect Version 1.2.0 onwards, the newer version of the Update Client was included with the Connect installation.

Updating stand-alone Workflow Messenger installations

If Workflow Messenger were installed stand alone, with no other Workflow components installed, the Update Client cannot find the Messenger component and thus it will not automatically update the component to the Workflow 8.5 version of Messenger. To get around this, download and run the Workflow 8.5 installer manually.

Templates Used in Workflow

For improved performance we recommend re-saving Workflow templates set up in the previous versions to run with OL Connect 1.5\Workflow 8.5.

Reduced Memory Version

> This is **not** recommended for production.

It is now possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

Connect 1.5 Designer Enhancements and Fixes

General Designer improvements

- A **color selection eyedropper** has been added, to allow the selection of a color from elsewhere on screen. (SHARED-33561/33646/36293)
- **Improved responsiveness** within the Designer, particularly when dealing with large and complex documents. (SHARED-44309)
- A **configurable Auto Save functionality** has been added for both templates and DataMapping configurations. (SHARED-40942/42085)
- Improvements made to **image file selection** functionality. (SHARED-42231/42451/42503/42556//43778)

Simplified creation of templates based on existing PDFs

- Option added to allow the creation of a **new print document based on an existing PDF**. (SHARED-19220)

- Improved support for **adding PDF files as Section backgrounds**. Files can now be referenced from disk or imported into Template. (SHARED-42496)
- Added support for drag and dropping **Data Fields** directly onto the page as absolutely positioned textboxes. (SHARED-43311)

Print Layout improvements

- **Page Number** formatting options (start/stop page numbering for sections, set numbering notation) improved in Print Section Properties dialog. (SHARED-39048)
- Added **repeating background images** support for print documents. (SHARED-43201)
- Option added to allow the insertion of **absolute positioned tables** on a master page. (SHARED-21967)

Email enhancements

- **User-definable SMTP settings**. New defaults are added for Sendgrid and Mailgun (in addition to Mandrill). (SHARED-43897)
- The standard **New email** wizard has been replaced with the new **Basic Email template** wizard. The new wizard has improved HTML structure. (SHARED-43338)
- Sending a **test email** no longer requires data. (SHARED-41889)
- **Tighter compression** for PDF attachments that are based on a print section. (SHARED-38575)
- **Colour picker** support added to the Email template wizards. (SHARED-33561)
- Added support for **PNG barcode** images in email messages. (SHARED-43787)

Barcode enhancements

- **Improved Barcode creation** with improved dialogs, better data validation and better error messaging. (SHARED-39295/42879)
- Font controls added to the **Barcode Properties** dialogs. (SHARED-22722/43659)
- Barcode improvements made in **Preview** mode. Support added for resizing and dragging of absolute positioned barcode objects, as well as resizing of inline barcode objects. (SHARED-43641)
- Barcodes can now have **transparent backgrounds**. (SHARED-43659)

Scripting improvements

- New ***closest()* command** added to the Scripting API, to locate closest matching element above it in the Document Object Model (DOM) tree. (SHARED-41789)
- **Script editing improved**. Line numbering now available within the editor, support for code completion and syntax highlighting added, as well as support for various ECMA6 commands. (SHARED-42768/43696)
- Support added for **cloning Sections in a Control Script** to allow a document to have a dynamic number of Sections. (SHARED-43683)
- Improved **Scripts tool tip** warning and error messaging. (SHARED-42550/43758) Improvements include:
    - Better tailored error messages and warnings.
    - Icons added representing script type as well as showing the issue severity.
    - Duplicated problems now filtered out.
    - Several other minor improvements.
- Improved support for raw HTML within Designer scripting API commands. (SHARED-43075)

Capture On The Go (COTG) and Web form improvements

- Input fields residing in a Field Table or Dynamic table are submitted using an array notation (requires Workflow 8.5). This results in a **nested XML structure** (grouped fields) in the job data file, simplifying the Workflow process and extracting data in the DataMapper. (SHARED-45577)
- COTG/Web-Form: Option introduced to retrieve the **jobdata XML** file from within the Designer. An icon has been added to the toolbar that intercepts the Submit action in Live view and then submits the form to a local or remote Workflow engine. A dialog also allows for saving the data file. (SHARED-44899)
- Deploying a COTG **Test form** no longer requires data. (SHARED-41889)
- **New scripting options** have been added to the COTG.js library to register custom functions for save and restore. (SHARED-40670)
- **Colour picker** support has been added to the **COTG Starter Template** wizards. (SHARED-33561)
- Improved speed/size of **COTG Camera** objects. Rather than embedding images in output PDFs, Connect now supports embedding data URLs in COTG templates. (SHARED-38575)

- Documents from the Library now **automatically deleted** upon successful submission. (COTG-367)

Connect 1.5 DataMapping Enhancements and Fixes

- **Multiple Conditions** step can now evaluate several conditions and branch out accordingly. (SHARED-14329/44435)
- **Performance improvements** made when extracting text from PDFs and spool files. (SHARED-43056)
- **Improved default formatting** when extracting Date, Float or Currency data fields. (SHARED-43415)
- An **extra field** is now appended to every Document record and to every Detail table inside that record. This allows other processes to add data on the fly. This provides enormous flexibility. For example, adding a JSON object (which could contain several additional fields) to the new field value extends the data structure almost infinitely. (SHARED-43518)
- The **JavaScript API** now displays detailed hints for every command, object and method available. (SHARED-44838)

Connect 1.5 Output Enhancements and Fixes

General

- Improved **content creation** processing speed for templates featuring PDF backgrounds. (SHARED-44350)

Email Output

- **Basic Email Action** wizard now made the default for new Email templates. (SHARED-43338)
- Support added for **user defined** SMTP/Email Service Provider (ESP) settings. (SHARED-43897)

Print Output

- New option added, allowing printing to **Windows Printer Driver**. (SHARED-35536)
- Improvements made to **external sort** option in Job presets. Support added for using input/output file placeholders. (SHARED-40944)
- New **HCF** file added that supports “top down wrap around sequence marks”. (SHARED-42326)

- Use **PostScript Media** name values in the PostScript DSC comments, to improve subsequent searches. (SHARED-42826)
- Option added to allow **storing of job resources** on PostScript printer's own storage medium. (SHARED-43467)
- **OMR marks improved**, with support added for Match Numbers (Match Code, MC). (SHARED-43589)
- A **Proof preview function** has been added to the Output Wizard, to display onscreen how the current print job would appear when printed. (SHARED-43885)
- **Imposition improvement**. Can now set specific starting position via new Offset option. (SHARED-44022)
- Minor glitches in **Booklet** and **Imposition** output addressed. (SHARED-44340/44430)
- IPDS output: Added **support for levels and metadata**. Future versions of the Print Manager will be able to search and restart based upon these values. (SHARED-44173)

Web Output

- Extra customization added to **custom OMR settings**. (SHARED-36267)

Connect 1.5 General Enhancements and Fixes

Installer improvements

- Improvements made to installation robustness. The installer now copes better when encountered **permissions issues** during installation. (SHARED-43732/43737)
- The **Update Client** has been updated to 1.1.9 and has been included in both the Connect 1.5 and Workflow 8.5 installations. (SHARED-47065)

Print Manager improvements

- **Removing all finished jobs** for a specific printer with a single step. (SHARED-42984)
- **Retry counters** added to interface. (SHARED-44772)
- Minor dialog **updates and improvements** throughout the Print Manager. (SHARED-30040/41975/42189/42804/43890/44113/44214)

Connect 8.5 Workflow Enhancements and Fixes

- Support for PHP-like arrays for **COTG** or web-based form submissions. (SHARED-41706)

- New Workflow system variable (%r) added to allow a process to determine which if it is currently running in **service** or **debug** mode. (SHARED-43411)
- ***Create Output*** and the Connect ***All In One*** tasks can now be added as Output tasks without waiting for the operation's result. (SHARED-43413)
- The ***Folder Capture*** task can now monitor multiple folders. (SHARED-43417)
- The ***HTTP Server Input*** task can now be set to monitor multiple actions to receive files from different URLs. (SHARED-43419)
- The ***Create Web*** and ***Create Email Content*** tasks have been enhanced with a list of the available template sections made available to simplify selection. (SHARED-43421)
- A new **Data Repository** has been created to allow for storing data that can then be subsequently reused, modified or augmented, by different processes.
  A new **Data Repository Management Tool** (DRMT) has also been added to the Workflow, to provide simple repository management tasks. (SHARED-43423/43438/43488/43521)
- Support added for Regular Expressions in the ***Folder Capture*** task FileName masks. (SHARED-43436)
- The **Debug information** panel is now automatically made visible when debugging a process. (SHARED-43763)
- Additional **encryption options** (RC4 and AES-256 in addition to AES-128) added for password protecting PDF files. (SHARED-44208)

Known Issues

Connecting to OL Learn

The link to the OL Learn site from the Welcome screen is broken in OL Connect 1.5.
To get to the OL Learn site, use this link: http://learn.objectiflune.com/en-ca/pres-connect

Memory Usage in Clustered Environments

In high speed clustered environments, OL Connect will fill the database faster than the clean-up service can clear it (the ratio is approximately 3:1). Therefore if the system needs to run for some time under a considerable load, a larger database should be used to prevent production from being interrupted.

We also recommend only the Master Server performing the database clean-up or the local Server Extension if the database is not installed with the Master to improve performance. For ALL Designers, Server and Server Extensions except one, set the clean-up of the database entities to run every 100 months and disable the clean-up for any Designer which

also has a Server or Server Extension on that machine. Note that if the Designer is a stand-alone then it is not connected to the central DB for clean-up purposes.

Installation Paths with Multi-Byte Characters

When installing the Chinese (Traditional or Simplified) or Japanese versions of Connect, if the user specifies an alternative installation path containing multi-byte/wide-char characters it can break some of the links to the Connect-related shortcuts in the Start Menu and cause an error to appear at the end of the installer. The workaround for the moment is to use the default installation path. The problem will be addressed in a later release.

Switching Languages

Changing the language using the **Window>Preferences>Language Setting** menu option does not currently change all of the strings in the application to the selected language. This is a known issue and will be fixed in a later release.

In the meantime we offer the following workaround for anyone who needs to change the language:

1. Go to the .ini files for the Designer and Server Config:
    - C:\Program Files\Objectif Lune\OL Connect\Connect Designer\Designer.ini
    - C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.ini
2. Change the language parameter to the required one under Duser.language=en | es | de | fr | it | ja | pt | tw | zh

Only one of the above language tags should be selected. Once saved, Connect will appear in the selected language at next start-up.

GoDaddy Certificates

When installing Connect offline, dialogs allow installing the GoDaddy certificates. Most users should use the default settings and click **Next**. In some cases, however, this may not work correctly. For this reason those users should activate **Place all certificates in the following store** and then select the **Trusted Root Certification Authorities** as the target certificate store.

MySQL Compatibility

After installing Connect 1.5 a downgrade to a Connect version earlier than Connect 1.3 or to a MySQL version earlier than 5.6.25 is not seamlessly possible. This is because the database

model used in Connect 1.3 and later (MySQL 5.6) is different to that used in earlier versions. If you need to switch to an older version of Connect / MySQL, it is first necessary to remove the Connect MySQL Database folder from "%ProgramData%\Connect\MySQL\data" before installing the older version.

PostScript Print Presets

The print presets for PostScript were changed from Version 1.1 onwards meaning that some presets created in Version 1.0 or 1.0.1 may no longer work.

Any PostScript print preset from Version 1.0 that contains the following will not work in Version 1.5: *.all[0].*

Any preset containing this code will need to be recreated in Version 1.5.

Available Printer Models

Note that only the single Printer Model (Generic PDF) will appear on the **Advanced** page of the **Print Wizard** by default.

To add additional printer models click on the settings button next to the Model selection entry box.

External Resources in Connect

There are certain limitations on how external resources can be used in Connect. For example if you want to link a file (e.g., CSS, image, JavaScript etc.) from a location on the network but you do not want to have a copy of the file saved with the template you need to do the following:

1. The resource must be located where it can be accessed by all Servers/Clients run as users. Failure to do this will cause the image to appear as a Red X in the output for all documents which were merged by engines which could not access the file. The job will terminate normally and the error will be logged.
2. The file must be referenced via a UNC path e.g.,

   file:////w2k8r2envan/z%20images/Picture/Supported/JPG/AB004763.jpg

   - UNC paths are required because the services will be unable to access mapped network drives (Windows security feature).
   - The engine processing the job will look on the local file system for the direct file path leading to the “resource not found” issue mentioned above.

Important Note: The Designer itself and Proof Print do not use processes that run as services and they may find local files with non-UNC paths which can lead to the false impression that the resources are correct.

Using Capture After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, Capture can no longer be used within Workflow 7. The plugins are now registered uniquely to Workflow 8 and the messenger for Workflow 7 is taken offline. It is only possible to use Capture from OL Connect Workflow 8 thereafter.

Capturing Spool Files After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, the PlanetPress Suite 7 option to capture spool files from printer queues will no longer function. The solution is to use OL Connect Workflow 8 to capture spool files from printer queues.

Colour Model in Stylesheets

The colour model of colours defined in a stylesheet can sometimes change after editing the stylesheet. This is a known issue and will be addressed in a subsequent release.

Online Help Links Point to Introductory Page

Context sensitivity for the online help is not yet enabled in Connect. All links and F1 calls point to the introductory page, where you can Search on keywords to bring up Help pages relating to the topic.

Context sensitivity will be introduced in a subsequent release of Connect.

Image Preview in Designer

If in the Windows Internet settings (**Connection Settings > LAN configuration**) a proxy is enabled, but "Bypass proxy settings for local addresses" is not checked, the image preview service, conversion service and live preview tab in the Designer will not work and exhibit the following issues:

- Images will be shows as 0 size boxes (no red 'X' is displayed)
- Live preview does not progress, and when re-activated reports "browsers is busy"

To fix the issue you must check the "Bypass proxy settings for local addresses" option.

Merge\Weaver Engines when Printing

The print operation in the Designer will automatically detect whether the Merge\Weaver engines are available and display a message for the user to retry or cancel if not. Once the Merge\Weaver engine becomes available and the user presses retry the print operation will proceed as normal. This message can also occur in the following circumstances:

- If the server is offline and you are not using Proof Print
- On some occasions before the Print Wizard opens

REST Calls for Remote Services

The Server will now accept REST calls for all remote services and will make commands wait indefinitely until the required engines become available. The Server will log when it is waiting for an engine and when it becomes available. Note that there is no way to cancel any commands other than stopping the Server.

Print Content and Email Content in Workflow

In Workflow's Print Content and Email Content tasks, the option to Update Records from Metadata will only work for fields whose data type is set to String in the data model. Fields of other types will not be updated in the database and no error will be raised. This will be fixed in a later release.

VIPP Output

Some templates set up with landscape orientation are being produced as portrait in VIPP. It can also sometimes be the case that text and images can be slightly displaced. These are known issues and will be addressed in a later release of Connect.

Print Limitations when the Output Server is located on a different machine

The following limitation may occur when using the Print options from a Designer located on a different machine to the Output Server:

- The file path for the prompt and directory output modes is evaluated on both the client AND server side. When printing to a network share it must be available to BOTH the Designer and Server for the job to terminate successfully.
- The Windows printer must be installed on both the Server and Designer machines.
- When printing via the Server from a remote Designer, the output file remains on the Server machine. This is remedied by selecting "Output Local" in the Output Creation configuration.

OL Connect Release Notes 1.4

Overview

This document provides an overview of the new features and enhancements in OL Connect 1.4.n and OL Connect Workflow 8.4.

Installing OL Connect 1.4.n and Workflow 8.4

- OL Connect is released as a 64 Bit version only (with the exception of the Workflow, Fax, Search and Imaging modules).
- Full details on installing and licensing OL Connect can be found in the online help in the installer.
- Note that the software comes with a 30 day trial license by default.

Updating from OL Connect 1.1

In order to upgrade from Connect Version 1.1 to Version 1.4.n via the Update Manager it is necessary to install a new version of the Objectif Lune Update Client. The next time you run your current Update Client it will show that there is an update available for itself. Simply click on the download button in the dialog to install the new version of the Update Client. Note that it is no problem to run the update while the Client is open. It will automatically update itself.

Once you have done this, OL Connect 1.4.n will become available for download.

From Connect Version 1.2.0 onwards, the newer version of the Update Client was included with the Connect installation.

Templates Used in Workflow

For improved performance we recommend resaving Workflow templates set up in the previous versions to run with OL Connect 1.4.n\Workflow 8.4.

Reduced Memory Version

> This is **not** recommended for production.

It is now possible to install OL Connect on a machine with a minimum of 2 GB of RAM. The OL Connect Designer will automatically detect whether it has been installed on a machine with less than 4 GB of RAM and default to only using one internal Weaver and one internal merge engine on that system. The Server will also run using internal engines.

Connect 1.4.2 Enhancements and Fixes

Designer

- A blank page is no longer added to beginning of templates that use scripting to add pages from PDF files. This problem only appeared when saving to a new file from within Preview mode, or when generating output from Preview mode. (SHARED-44564)
- Image elements referencing a PDF image would multiply when switching back and forth between Live and Preview modes, in both email and web contexts. This has now been fixed. (SHARED-44066)

Email and Web Output

- Elements whose style was set to `display: none` would be removed from HTML output, rather than just not being displayed. This error has been fixed. (SHARED-44151)

  Elements hidden via the Conditional Script wizard are removed from the output.

Print Manager

- IPDS - Printing a range of pages with Trace On would lead to errors. This has now been fixed. (SHARED-44291)

Connect 1.4.1 New Features and Enhancements

New Languages Added

The Connect user interface is now supported in Spanish, Italian, Portuguese and Chinese (Simplified) as well as English, French, German, Japanese and Chinese (Traditional). The default language remains English. Further languages will be introduced in later releases.

The language can be selected during the installation of Connect or via the Language Setting options in the Preferences dialog (note that Connect needs to be restarted in order to apply the selected language).

At present only the Connect user interface has been translated. Error messages and warnings will be translated for a later release.

Welcome Screen Extended

- The Printer Definition Configs and HCF files available on the OL Connect website are now grouped by manufacturer, to simplify selection.
- Connect 1.4.1 also introduces **Responsive Email Templates**.

Virtualisation

- Connect is now supported on the Microsoft Hyper-V and Hyper-V/Azure environments as well as the VMWare Workstation, Server, Player and ESX infrastructure environments.

Modifying Connect Installations

- Connect 1.4.1 introduces the ability to Modify Connect installations, once Connect 1.4.1 has been installed.

Connect 1.4.1 Designer Enhancements and Fixes

Capture OnTheGo

- **Remote CSS** and **Javascript** resources can now be pre-fetched and shared between documents to avoid having to embed them in every template. (SHARED-39095)
- **Signature widgets** can now be marked as required. (SHARED-39833)
- **Annotations** can now be added to static images (not the picture widget). (SHARED-40369)

Email Context

- Email context sections can be **enabled or disabled based upon data value**. (SHARED-33656)
- Email **port number** can now be specified as part of the host name. (SHARED-38008)
- **New template wizard** for Slate templates by Litmus. (SHARED-36843)

Print Context

- Ability added to **mirror margins on back pages of Duplex jobs**, via Facing Pages selection added in Sheet Configuration dialog. (SHARED-40505)
- Can now suppress Master Page on duplex back pages, if there are no contents.

Scripting

- Result can now be written as a JSON string into an attribute or text (instead of the field value). Useful for web contexts where a front-end script can easily read the value.
- User-defined formatting masks can now be used when outputting dates and numerical values.
- Conditional script set to "Show if Condition is true" will hide the object by default.

- Page range can be specified via scripting when setting PDF as background. (SHARED-36998)
- Matched elements are highlighted when hovering over scripts. (SHARED-38293)

General

- Numerous improvements to Designer GUI and user-experience. These include:
  - A new "**Styles**" pane that displays the styles applied to the selected object.
  - **Image Selection** Dialog now inserts from resources, files or URL.
  - **Line number** option now available from any source view.
  - View menu now has entries to **switch between views** within the Workspace.
- Set any **PDF as background** for a Section. (SHARED-39880)
- Specify a **background image** and control its size, position and repeat mode. (SHARED-14522)
- Option to **generate JSON string from data model fields** to pass data record information to client side script. (SHARED-39337)
- **CSS Class name completion** suggests CSS classes based upon the current section. (SHARED-36870)
- **CSS Style inspector** allows full control over styles. (SHARED-22929)

Connect 1.4.1 DataMapping Enhancements and Fixes

- **All fields can now be renamed** through the Data Model view. (SHARED-40116)
- SQL Server data mapping can now use **Windows Authentication** credentials. (SHARED-38999)
- Support added for **MS SQL Server** as the backend database. (SHARED-39959)

Connect 1.4.1 Output Enhancements and Fixes

Print Output

- Images, barcodes, OMRs and text can now all be added to the page at time of output generation.
- Ability to **add Metadata** to any output type (previously only PDF and AFP), for use within output Presets.
- **Static strings** can now be added to Metadata in job Presets.
- The **Output Path** in the Output Preset can now be set dynamically.

- New Color setting for the **Add Text** option. (SHARED-40830)
- **Monochrome** and **Dithering** support added to PCL output. (SHARED-39937)
- **Dithering** and **B&W threshold** values now supported in the Print Wizard. (SHARED-39520)
- New option to control **Compact Font Format** settings within PostScript. (SHARED-39902)
- Support for Simple Network Management Protocol **(SNMP) queries** added to Print Manager. (SHARED-39213)

Email Output

- **SMTP port** can now be customized when sending email.

Connect 8.4.1 Workflow Enhancements and Fixes

- Major performance improvements when updating data records. (SHARED-38897)
- Stand-alone **Update Data Record** task allows data records to be updated in the database without having to create content. (SHARED-38867)
- **Update Records** operations performed in batches, allowing for unlimited number of records. (SHARED-38948)
- **All-In-One task** can now return the output file to the Workflow, to allow customising output destination(s). (SHARED-39434)
- **Job information** (such as Owner and Job Name) now sent to the printer in PostScript and Windows Driver output.

Known Issues

Memory Usage in Clustered Environments

In high speed clustered environments, OL Connect will fill the database faster than the clean-up service can clear it (the ratio is approximately 3:1). Therefore if the system needs to run for some time under a considerable load, a larger database should be used to prevent production from being interrupted.

We also recommend only the Master Server performing the database clean-up or the local Server Extension if the database is not installed with the Master to improve performance. For ALL Designers, Server and Server Extensions except one, set the clean-up of the database entities to run every 100 months and disable the clean-up for any Designer which also has a Server or Server Extension on that machine. Note that if the Designer is a stand-alone then it is not connected to the central DB for clean-up purposes.

Installation Paths with Multi-Byte Characters

When installing the Chinese (Traditional or Simplified) or Japanese versions of Connect, if the user specifies an alternative installation path containing multi-byte/wide-char characters it can break some of the links to the Connect-related shortcuts in the Start Menu and cause an error to appear at the end of the installer. The workaround for the moment is to use the default installation path. The problem will be addressed in a later release.

Switching Languages

Changing the language using the **Window>Preferences>Language Setting** menu option does not currently change all of the strings in the application to the selected language. This is a known issue and will be fixed in a later release.

In the meantime we offer the following workaround for anyone who needs to change the language:

1. Go to the .ini files for the Designer and Server Config:
    - C:\Program Files\Objectif Lune\OL Connect\Connect Designer\Designer.ini
    - C:\Program Files\Objectif Lune\OL Connect\Connect Server Configuration\ServerConfig.ini
2. Change the language parameter to the required one under Duser.language=en | fr | de | ja | zh

Only one of the above language tags should be selected. Once saved, Connect will appear in the selected language at next start-up.

GoDaddy Certificates

When installing Connect offline, dialogs allow installing the GoDaddy certificates. Most users should use the default settings and click **Next**. In some cases, however, this may not work correctly. For this reason those users should activate **Place all certificates in the following store** and then select the **Trusted Root Certification Authorities** as the target certificate store.

MySQL Compatibility

After installing Connect 1.4.n a downgrade to a Connect version older than Connect 1.3 or to a MySQL version earlier than 5.6.25 is not seamlessly possible. This is because the database model used in Connect 1.3 and 1.4.n (MySQL 5.6) is different to earlier versions. If you need to switch to an older version of Connect / MySQL, it is necessary to remove the Connect

MySQL Database folder from "%ProgramData%\Connect\MySQL\data" before installing the older version.

PostScript Print Presets

The print presets for PostScript were changed from Version 1.1 onwards meaning that some presets created in Version 1.0 or 1.0.1 may no longer work.

Any PostScript print preset from Version 1.0 that contains the following will not work in Version 1.4: *.all[0].*

Any preset containing this code will need to be recreated in Version 1.4.

Available Printer Models

Note that only the single Printer Model (Generic PDF) will appear on the **Advanced** page of the **Print Wizard** by default.

To add additional printer models click on the settings button next to the Model selection entry box.

External Resources in Connect

There are certain limitations on how external resources can be used in Connect. For example if you want to link a file (e.g., CSS, image, JavaScript etc.) from a location on the network but you do not want to have a copy of the file saved with the template you need to do the following:

1. The resource must be located where it can be accessed by all Servers/Clients run as users. Failure to do this will cause the image to appear as a Red X in the output for all documents which were merged by engines which could not access the file. The job will terminate normally and the error will be logged.
2. The file must be referenced via a UNC path e.g.,

   file:////w2k8r2envan/z%20images/Picture/Supported/JPG/AB004763.jpg

   - UNC paths are required because the services will be unable to access mapped network drives (Windows security feature).
   - The engine processing the job will look on the local file system for the direct file path leading to the “resource not found” issue mentioned above.

Important Note: The Designer itself and Proof Print do not use processes that run as services and they may find local files with non-UNC paths which can lead to the false impression that the resources are correct.

Using Capture After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, Capture can no longer be used within Workflow 7. The plugins are now registered uniquely to Workflow 8 and the messenger for Workflow 7 is taken offline. It is only possible to use Capture from OL Connect Workflow 8 thereafter.

Capturing Spool Files After Installing Workflow 8

If OL Connect Workflow 8 is installed alongside PlanetPress Suite Workflow 7, the PlanetPress Suite 7 option to capture spool files from printer queues will no longer function. The solution is to use OL Connect Workflow 8 to capture spool files from printer queues.

Colour Model in Stylesheets

The colour model of colours defined in a stylesheet can sometimes change after editing the stylesheet. This is a known issue and will be addressed in a subsequent release.

Online Help Links Point to Introductory Page

Context sensitivity for the online help is not yet enabled in Connect. All links and F1 calls point to the introductory page, where you can Search on keywords to bring up Help pages relating to the topic.

Context sensitivity will be introduced in a subsequent release of Connect.

Image Preview in Designer

If in the Windows Internet settings (**Connection Settings > LAN configuration**) a proxy is enabled, but "Bypass proxy settings for local addresses" is not checked, the image preview service, conversion service and live preview tab in the Designer will not work and exhibit the following issues:

- Images will be shows as 0 size boxes (no red 'X' is displayed)
- Live preview does not progress, and when re-activated reports "browsers is busy"

To fix the issue you must check the "Bypass proxy settings for local addresses" option.

Merge\Weaver Engines when Printing

The print operation in the Designer will automatically detect whether the Merge\Weaver engines are available and display a message for the user to retry or cancel if not. Once the Merge\Weaver engine becomes available and the user presses retry the print operation will proceed as normal. This message can also occur in the following circumstances:

- If the server is offline and you are not using Proof Print
- On some occasions before the Print Wizard opens

REST Calls for Remote Services

The Server will now accept REST calls for all remote services and will make commands wait indefinitely until the required engines become available. The Server will log when it is waiting for an engine and when it becomes available. Note that there is no way to cancel any commands other than stopping the Server.

Print Content and Email Content in OL Connect Workflow

In OL Connect Workflow's Print Content and Email Content tasks, the option to Update Records from Metadata will only work for fields whose data type is set to String in the data model. Fields of other types will not be updated in the database and no error will be raised. This will be fixed in a later release.

VIPP Output

Some templates set up with landscape orientation are being produced as portrait in VIPP. It can also sometimes be the case that text and images can be slightly displaced. These are known issues and will be addressed in a later release of Connect.

Print Limitations when the Output Server is located on a different machine

The following limitation may occur when using the Print options from a Designer located on a different machine to the Output Server:

- The file path for the prompt and directory output modes is evaluated on both the client AND server side. When printing to a network share it must be available to BOTH the Designer and Server for the job to terminate successfully.
- The Windows printer must be installed on both the Server and Designer machines.
- When printing via the Server from a remote Designer, the output file remains on the Server machine. This is remedied by selecting "Output Local" in the Output Creation configuration.

Support

Additional resources

The following resources are a great addition to this online Help and will help you learn more about the software and find solutions if you run into issues.

- Walkthroughs: Step-by-step walkthrough guides introduce you to the functionality of OL Connect at your own pace.
- Workflow Help: OL Connect Workflow enables you to build customized automated processes that fit your environment.
- REST API: The OL Connect REST API service provides programmatic access to the Connect Server in areas such as output creation, data management and file store operations.
- OL Connect Resource Center: Read our latest and greatest blog posts, follow hands-on tutorials and download examples and resources.
- Forum: Visit the user forum to obtain the information that *you* need to understand and use the software.
- Community: In the Upland Community, you can find information on product releases, submit and track support cases, sign up for notifications, post and reply to forum topics, and learn about training and webinars.
- If you run into an issue, take a look at the "Log files" on page 21.

Report an issue

The Upland Support team is dedicated to helping our customers succeed by providing timely resolutions to your issues and questions.

You can contact Support by by phone or via the Community.

Legal Notices and Acknowledgments

Warning: OL Connect is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, via any means, in part or in whole, may be prosecuted to the full extent of the law.

The license agreements for the associated open source third party components can be found in the following installation folder: *C:\Program Files\Objectif Lune\OL Connect\Legal Notices*

This application uses the following third party components:

- **Adobe PDF Library** which is either a registered trademark or trademark of Adobe Systems Incorporated in the United States and\or other countries.
- **Adobe XMP Core** Copyright © 1999 - 2010, Adobe Systems Incorporated. All rights reserved.
- **ASM librares** which are distributed under a BSD 3-clause License
- **Bouncy Castle** which is distributed under an MIT license (MIT License)
- **Dart Sass** which is available for use and modification under the MIT license (MIT License).
- **DD Plist** which is distributed under an MIT license (MIT License)
- **Eclipse Gemini Blueprint** which is distributed under the terms of the Apache Software License Version 2.0. This product includes sub-components with separate copyright notices and license terms.
- **Eclipse Nebula** This application also uses the following Eclipse Nebula components which are distributed under the terms of the **Eclipse Public License (EPL) v 2.0** :
    - Radiogroup Widgets
- **Eclipse Persistence Services Project (EclipseLink)**, Copyright © 2007, Eclipse Foundation, Inc. and its licensors. All rights reserved. This is distributed under the terms of the Eclipse Public License Version 1.0 and Eclipse Distribution License Version 1.0.
- **Fugue Icons** by Yusuke Kamiyamane which are distributed under the terms of the Creative Commons Attribution 3.0 License.
- **Gecko** which is distributed under the terms of the Mozilla Public License (MPL) Version 2.0. Information on obtaining Gecko can be found on the following page: https://wiki.mozilla.org/Gecko:Getting_Started
  NOTE: This library has been modified for Connect. To obtain copies of the modified library please contact your local Objectif Lune Support team.

- **Glassfish** This application also uses the following Glassfish components which are distributed under the terms of the **Eclipse Public License (EPL) v 2.0** :
    - HK2 API module
    - HK2 Implementation Utilities
    - Javax Inject
    - Jersey
    - Jersey Bean Validation
    - Jersey Container Jetty HTTP
    - Jersey Container Servlet
    - Jersey Container Servlet Core
    - OSGi Resource Locator
    - ServiceLocator Default Implementation

  Information on how to download the Glassfish project sources can be obtained from here: https://mvnrepository.com/search?q=org.glassfish
- **Glassfish Java Mail** which is licensed under the terms of the Common Development and Distribution License (CDDL) Version 1.0. Information on how to download the Glassfish source can be obtained from here: https://wikis.oracle.com/display/GlassFish/Java+EE+7+Maven+Coordinates
- **Google Core Protocol Buffers** library which is distributed under a BSD 3-clause License
- **Hamcrest Matchers** Copyright © 2000-2006, www.hamcrest.org. All rights reserved.
- **HyperSQL**, Copyright © 2001-2010, The HSQL Development Group. All rights reserved.
- **IcoMoon**. Connect uses unmodified icons from IcoMoon (https://icomoon.io/#icons-icomoon) which have been made available under the Creative Commons By 4.0 license (https://creativecommons.org/licenses/by/4.0).
- **ICU4J 4.4.2** Copyright © 1995-2013 International Business Machines Corporation and others. All rights reserved.
- **Jacob Java Com Bridge** which is licensed under the terms of the GNU Lesser General Public License (LGPL) Version 2. The source code for this can be obtained from the following location: http://sourceforge.net/projects/jacob-project/files/jacob-project/

- **Jakarta Annotation API** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/common-annotations-api
- **Jakarta Enterprise Beans** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/enterprise-beans
- **Jakarta Expression Language** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/eclipse-ee4j/el-ri
- **Jakarta JSON Processing** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/jsonp-api
- **Jakarta Interceptors** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/eclipse-ee4j/interceptor-api
- **Jakarta Mail** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/eclipse-ee4j/mail
- **Jakarta Messaging** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/eclipse-ee4j/messaging
- **Jakarta Resource API** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/connectors
- **Jakarta RESTful Web Services** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/rest
- **Jakarta Servlet** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://projects.eclipse.org/projects/ee4j.servlet
- **Jakarta Transaction API** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/transactions

- **Jakarta WebSocket** which is distributed under the terms of the Eclipse Public License v 2.0 (EPL2). The source code for this can be obtained from the following location: https://github.com/jakartaee/websocket
- **Java Advanced Imaging Image I/O Tools** which is distributed under a BSD 2-clause License (BSD-2-Clause license)
- **Java XmlHttpRequest** which is licensed under the terms of the GNU Lesser General Public License Version (LGPL) 2.1. The source code for this can be obtained from the following location: https://github.com/objectifluneCA/java-XmlHttpRequest
- **Jaxb OSGI** which is licensed under the terms of the Common Development and Distribution License (CDDL) Version 1.1. The source code can be found at the following location: https://bitbucket.org/uplandsoftware/vendor/src/master/java/github.com/javaee/jaxb-v2/
- **JBCrypt** library which is distributed under the ISC License (ISC)
- **Jersey** which is distributed under the terms of the Common Development and Distribution License (CDDL) Version 1.1. Information on how to obtain the source code can be found at the following location: http://repo1.maven.org/maven2/org/glassfish/jersey/jersey-bom
- **Jersey Multipart** which is distributed under the terms of the Common Development and Distribution License (CDDL) Version 1.1. Information on how to obtain the source code can be found at the following location: http://repo1.maven.org/maven2/org/glassfish/jersey/jersey-bom
- **JNA Version 3.5.1** which is distributed under the terms of the GNU Lesser General Public License Version (LGPL) 2.1. The source code for this can be obtained from the following location: https://github.com/twall/jna/releases
- **jQuery library** which is distributed under an MIT license (https://jquery.org/license/).
- **jQuery validation library** which is distributed under an MIT license (JQuery Validation licence).
- **Jsoup** which is made available for use and modification under the MIT license (MIT License).
- **Little-CMS** which is made available for use and modification under the MIT license (MIT License).

- **Logback** which is distributed under the terms of the Eclipse Public License (EPL) Version 1.0. The source code for Logback can be obtained from the following location: https://logback.qos.ch/download.html
- **MariaDB** which is distributed under the terms of the GNU General Public License Version 2. Information on how to obtain the source code can be found at the following location: https://mariadb.org/get-involved/getting-started-for-developers/get-code-build-test/
- **MariaDB Java Client** which is distributed under the terms of the GNU Lesser General Public License Version (LGPL) 2.1. The source code for this can be obtained from the following location: https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client
- **Minimatch Java** which is distributed under an MIT license (MIT License).
- **Objectweb ASM**, Copyright © 2000-2011 INRIA, France Telecom. All rights reserved.
- **Oracle JDBC Driver** which is licensed by the Oracle Free Use Terms and Conditions (FUTC) licence.
- **org.eclipse.equinox.weaving.springweaver** which is distributed under the terms of the Eclipse Public License (EPL) Version 1.0. The source code can be obtained from the following location: https://github.com/osgi-forks/dynaresume
- **PDF Renderer** which is licensed under the terms of the Lesser General Public License (LGPL) Version 2.1. The source code can be obtained from the following location: https://mvnrepository.com/artifact/pdf-renderer/pdf-renderer
- **Polyfills** which is licensed under the Unlicense license. The source code can be obtained from the following location: https://github.com/inexorabletash/polyfill.
- **RapidJSON** which is distributed under an MIT License.
- **Relique CSV Driver** which is licensed under the terms of the Lesser General Public License (LGPL) Version 2.1. The source code can be obtained from the following location: https://sourceforge.net/p/csvjdbc/code/ci/csvjdbc-1.0.31/tree/
- **Rhino 1.7R4 and 1.7.7.1** which are licensed under the terms of the Mozilla Public License (MPL) Version 2.0. The source code for these can be obtained from the following location: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/Download_Rhino
- **SASS Embedded libraries** which are made available for use and modification under the MIT license (MIT License).

- **Saxon** which is distributed under the terms of the Mozilla Public License (MPL) Version 2.0. The source code for this can be obtained from the following location: http://-sourceforge.net/projects/saxon/files/Saxon-HE/9.6/
- **Servlet API** developed by Sun as part of the Glassfish project and licensed under the terms of the Common Development and Distribution License (CDDL) Version 1.0. Information on how to download the Glassfish source (as part of Java EE platform) can be obtained from here: https://wi-kis.oracle.com/display/GlassFish/Java+EE+7+Maven+Coordinates
- **Simple Logging Facade for Java (SLF4J)** Copyright © 2004-2017 QOS.ch. All rights reserved.
- **SpotBugs Annotations** which is licensed under the terms of the GNU Lesser General Public License (LGPL) Version 2.1. The source code for this can be obtained from the following location: https://github.com/spotbugs/spotbugs
- **Spring Framework** which is distributed under the terms of the Apache Software License Version 2.0. This product includes sub-components with separate copyright notices and license terms.
- **StAX Utilities** Copyright © 2007, StAX Utilities Project. All rights reserved.
- **Stax2 API** Copyright 2010-2018 FasterXML.com.
- **Tern** which is distributed under the terms of the Eclipse Public License (EPL) Version 1.0. The source code for tern can be obtained from the following location: https://-github.com/angelozerr/tern.java
- **Web Services Description Language for Java** which is distributed under the terms of the Common Public License v 1.0. The source code for this can be obtained from the following location: http://wsdl4j.cvs.sourceforge.net/viewvc/wsdl4j/
- **XStream Core** library which is distributed under a BSD 3-clause License
- **XULRunner** which is distributed under the terms of the Mozilla Public License Version 2.0. The source code for this can be obtained from the following location: http://ft-p.mozilla.org/pub/mozilla.org/xulrunner/releases/latest/source/
- **zziplib** which is licensed under the terms of the Mozilla Public License (MPL) Version 1.1. The source code for this can be obtained from the following location: http://-sourceforge.net/projects/zziplib/files/zziplib13/

Portions of certain libraries included in this application which are distributed under the terms of the **Mozilla Public License** have been modified. To obtain copies of the modified libraries please contact your local Objectif Lune Support team.

Apache Software License Components

This application also uses the following components which are distributed under the terms of the **Apache Software License Version 2.0**:

- Apache ActiveMQ
- Apache Aries
- Apache Batik
- Apache Commons Beanutils
- Apache Commons CLI
- Apache Commons Codec
- Apache Commons Collections
- Apache Commons Compress
- Apache Commons Digester
- Apache Commons Imaging
- Apache Commons IO
- Apache Commons JCS Core
- Apache Commons Lang
- Apache Commons Logging
- Apache Commons Math
- Apache Commons Pool
- Apache Commons Text
- Apache Commons Validator
- Apache Commons JXPath
- Apache Felix and dependencies
- Apache Geronimo
- Apache HttpClient
- Apache HttpClient Mime
- Apache HttpClient Windows features
- Apache HttpComponents Core HTTP/1.1
- Apache HttpComponents Core HTTP/2
- Apache HttpCore

- Apache Log4j API
- Apache Log4j to SLF4J
- Apache POI
- Apache ServiceMix
- Apache Tika libraries
- Apache Xerces2 Java Parser
- Apache XML Graphics
- Apache XML Beans
- ASN1bean
- Barcode4j
- Gemini Blueprint Core
- Gemini Blueprint Extender
- Gemini Blueprint Extensions
- Gemini Blueprint IO
- Google Collections
- Google GSON
- Google Guava
- Hibernate Validator
- HikariCP
- Jackcess
- Jackson JSON processor
- Jackson Module Jakarta Xmlbind Annotations
- Jakarta Contexts and Dependency Injection
- Jakarta Dependency Injection
- Jakarta Validation API
- Javet
- JCIP Annotations
- JDBI
- JetBrains Java Annotations

- Jetty
- Jetty alpn client
- Jetty webapp
- Jetty Websocket
- Jetty Websocket API
- JSON Path
- JSON Sanitizer
- JSON Small and Fast Parser
- Kotlin Common Standard Library
- Kotlin Standard Library for JVM
- Liquibase
- LMAX Disruptor
- Minidev
- Nimbus JOSE+JWT
- Objenesis
- OkHttp
- OkIO
- OpenCSV
- OpenTelemetry API Incubator
- OpenTelemetry for Java
- OpenTelemetry Instrumentation for Java
- OPS4J Pax Web
- org.eclipse.persistence.logging.slf4j
- org.json.simple
- OSGI
- PAC4J
- Sisyphsu DateParser
- Sisyphsu Retree
- Snakeyaml

- SNMP4J
- SparseBitSet
- Spring Dynamic Modules
- Swagger
- Tika
- UCanAccess
- Woodstox
- xalan
- XML Resolver

Eclipse Technology:

This Software includes unmodified Eclipse re-distributables, which are available at www.eclipse.org.
The Eclipse re-distributables are distributed under the terms of the Eclipse Public License version 1.0 (https://www.eclipse.org/legal/epl-v10.html) or the Eclipse Distribution License version 1.0 (https://www.eclipse.org/org/documents/edl-v10.php).

The following disclaimer applies for all of the Eclipse Distribution Licensed libraries:
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Eclipse Adoptium OpenJDK

This application uses unmodified re-distributables from the Eclipse Adoptium OpenJDK project. The Licenses used in that project are:

- Build scripts and other code to produce the binaries, the website and other build infrastructure are licensed under Apache License, Version 2.0.
- OpenJDK code itself is licensed under GPL v2 with Classpath Exception (GPLv2+CE).
- Eclipse OpenJ9 is licensed under Several licenses.

Further Components:

- Portions of this software are copyrighted by **The FreeType Project** (www.freetype.org). All rights reserved.
- This product includes software developed by **JSON.org** (https://www.json.org), which is distributed under the JSON License (JSON License)
- This product includes software developed by the Indiana University Extreme! Lab. For further information please visit http://www.extreme.indiana.edu/

End User License Agreement (EULA)

Click here to download the EULA as a PDF

Copyright Information

Copyright © 1994-2025 Upland Objectif Lune. All Rights Reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any other language or computer language in whole or in part, in any form or by any means, whether it be electronic, mechanical, magnetic, optical, manual or otherwise, without prior written consent of Upland Objectif Lune.

Upland Objectif Lune disclaims all warranties as to this software, whether expressed or implied, including without limitation any implied warranties of merchantability, fitness for a particular purpose, functionality, data integrity or protection.

OL Connect is a registered trademark of Upland Objectif Lune.