Contents

1. Copyright Notice ......................................................................................................................................................... 6

2. Introduction ..................................................................................................................................................................... 8
  2.1 Why Integrate? .......................................................................................................................................................... 8
  2.2 External System sending commands To Automation Engine ...................................................................................... 9
  2.3 Automation Engine sending commands To an External System ............................................................................... 9
  2.4 Some general advice .................................................................................................................................................... 9
  2.5 Need any help? ......................................................................................................................................................... 10
  2.6 What about JDF? ...................................................................................................................................................... 10

3. Integration Concepts ..................................................................................................................................................... 11
  3.1 Access Points Triggering a Workflow on Automation Engine .................................................................................. 11
  3.2 Automation Engine tasks Interacting directly with the External System .................................................................. 12
  3.3 Automation Engine Mapper tasks Transforming Data ............................................................................................... 13
    3.3.1 Transforming data from an Access Point .................................................................................................................. 14
    3.3.2 Transforming data from a workflow .......................................................................................................................... 15
  3.4 Automation Engine sending data Back to the External System .................................................................................. 15

4. Access Points ................................................................................................................................................................. 17
  4.1 Folder Access Point .................................................................................................................................................... 18
    4.1.1 Concept ................................................................................................................................................................. 18
    4.1.2 What about Hot Folders? ........................................................................................................................................ 19
    4.1.3 Creating or modifying a Folder Access Point ........................................................................................................ 19
  4.2 FTP Access Point ....................................................................................................................................................... 21
    4.2.1 Concept ................................................................................................................................................................. 21
    4.2.2 Creating or modifying an FTP Access Point ........................................................................................................... 21
  4.3 Mail Access Point ....................................................................................................................................................... 24
    4.3.1 Concept ................................................................................................................................................................. 24
    4.3.2 Create or modifying a Mail Access Point ................................................................................................................ 24
  4.4 Database Access Point .............................................................................................................................................. 26
    4.4.1 Concept ................................................................................................................................................................. 26
    4.4.2 Create or modifying a Database Access Point ...................................................................................................... 27
  4.5 Web Service Access Point ......................................................................................................................................... 30
    4.5.1 Concept ................................................................................................................................................................. 30
    4.5.2 Checking the Automation Engine Web Service ................................................................................................... 30
    4.5.3 Creating a Web Service Access Point .................................................................................................................... 31
    4.5.4 HTTP POST Requests ............................................................................................................................................ 32
    4.5.5 HTTP GET Requests ............................................................................................................................................ 34
    4.5.6 Error Handling ..................................................................................................................................................... 35

5. Data Transforming Tasks ............................................................................................................................................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Create XML File task</td>
<td>37</td>
</tr>
<tr>
<td>5.2</td>
<td>Split XML Task</td>
<td>39</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Concept</td>
<td>39</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Example and Options</td>
<td>39</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Creating XPath Expressions</td>
<td>41</td>
</tr>
<tr>
<td>5.3</td>
<td>Join XML Files Task</td>
<td>43</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Concept</td>
<td>43</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Example and Options</td>
<td>43</td>
</tr>
<tr>
<td>5.4</td>
<td>Map Data task</td>
<td>46</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Concept</td>
<td>46</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Main Tools in the Task Panel</td>
<td>47</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Main Workflow in Using the Map Data Task</td>
<td>48</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Setting Up (Example) Input Files</td>
<td>49</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Adding Parameters</td>
<td>51</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Specifying the Output File</td>
<td>54</td>
</tr>
<tr>
<td>6.0</td>
<td>Tasks Interacting with other Systems</td>
<td>57</td>
</tr>
<tr>
<td>6.1</td>
<td>Interact with Database task</td>
<td>57</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Concept</td>
<td>57</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Set Up and Options</td>
<td>57</td>
</tr>
<tr>
<td>6.2</td>
<td>Interact with Web Service task</td>
<td>59</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Concept</td>
<td>59</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Set Up and Options</td>
<td>60</td>
</tr>
<tr>
<td>7.0</td>
<td>Tasks often used in Integration Workflows</td>
<td>63</td>
</tr>
<tr>
<td>7.1</td>
<td>Create Job task</td>
<td>63</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Concept, Workflow and FAQs</td>
<td>63</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Job</td>
<td>65</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Customer</td>
<td>66</td>
</tr>
<tr>
<td>7.1.4</td>
<td>Inks</td>
<td>66</td>
</tr>
<tr>
<td>7.1.5</td>
<td>Parameters</td>
<td>68</td>
</tr>
<tr>
<td>7.1.6</td>
<td>Bar Codes</td>
<td>68</td>
</tr>
<tr>
<td>7.1.7</td>
<td>Template</td>
<td>69</td>
</tr>
<tr>
<td>7.1.8</td>
<td>Advanced</td>
<td>70</td>
</tr>
<tr>
<td>7.1.9</td>
<td>Sample Input XML</td>
<td>70</td>
</tr>
<tr>
<td>7.2</td>
<td>Add To Products task</td>
<td>71</td>
</tr>
<tr>
<td>7.3</td>
<td>Link Product To Job task</td>
<td>71</td>
</tr>
<tr>
<td>8.0</td>
<td>Examples</td>
<td>72</td>
</tr>
<tr>
<td>8.1</td>
<td>Creating a Job using XML</td>
<td>72</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Step 1 - Analyze your Input XML file</td>
<td>72</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Step 2 - Create your SmartNames</td>
<td>73</td>
</tr>
<tr>
<td>8.1.3</td>
<td>Step 3 - Insert the SmartNames in the Create Job ticket</td>
<td>76</td>
</tr>
<tr>
<td>8.1.4</td>
<td>Step 4 - Test your Workflow</td>
<td>80</td>
</tr>
<tr>
<td>8.1.5</td>
<td>Step 5 - Add automation by starting this workflow from an Access Point</td>
<td>80</td>
</tr>
</tbody>
</table>
8.2 Creating Multiple Jobs from One XML

8.2.1 Step 1 - Analyze the XML describing Multiple Jobs

8.2.2 Step 2 - Create the Workflow Ticket

8.2.3 Step 3 - Splitting the XML but also using a Filter on Status

8.2.4 Step 4 - Importance of the Data Splitter

8.2.5 Step 5 - Create the Xpath SmartNames

8.2.6 Step 6 - Workflow Result

8.3 Create Multiple Jobs from a CSV file

8.3.1 Step 1 - Analyse the CSV file

8.3.2 Step 2 - Create the Workflow Ticket

8.3.3 Step 3 - Mapping the CSV to XML

8.3.4 Step 4 - Test your CSV to XML Mapping

8.3.5 Step 5 - Set up the Split XML step

8.3.6 Step 6 - Test your workflow again

8.3.7 Step 7 - Check the Data Splitter

8.3.8 Step 8 - Defining the SmartNames for the Create Job ticket

8.3.9 Step 9 - Workflow Result

8.4 Creating Jobs using a Database Access Point

8.4.1 Step 1 - Analyse the External Database

8.4.2 Step 2 - Configure the Link to the External Database

8.4.3 Step 3 - Set up the Database Access Point

8.4.4 Step 4 - Test your Database Access Point

8.4.5 Step 5 - Create your SmartNames

8.4.6 Step 6 - Create the Workflow with the Create Job task

8.4.7 Step 7 - Test your workflow

8.4.8 Step 8 - Add the workflow to the Access Point and test again

8.5 Launching a workflow with parameters from an XML

8.5.1 Step 1 - Analyse the incoming XML

8.5.2 Step 2 - Create the Workflow

8.5.3 Step 3 - Load Workflow Parameters from the XML

8.5.4 Step 4 - Select the Design file that is Referenced in the XML

8.5.5 Step 5 - Pick up the Workflow Parameters in your Workflow

8.5.6 Step 6 - Data structure of this Workflow

8.5.7 Step 7 - Test the Workflow

8.5.8 Step 8 - Create the Folder Access Point and Test Again

8.5.9 Step 9 - Possible Workflow Extensions

8.6 Getting info from a Web Service into a Workflow

8.6.1 Step 1 - Analyse the Input XML

8.6.2 Step 2 - Configure the Web Service

8.6.3 Step 3 - Create the workflow

8.6.4 Step 4 - Load Workflow Parameters from the XML

8.6.5 Step 5 - Define the 2 Interact with Web Service steps

8.6.6 Step 6 - Settings to Collect and Sort the 2 Responses
1. Copyright Notice

© Copyright 2015 Esko Software BVBA, Gent, Belgium

All rights reserved. This material, information and instructions for use contained herein are the property of Esko Software BVBA. The material, information and instructions are provided on an AS IS basis without warranty of any kind. There are no warranties granted or extended by this document. Furthermore Esko Software BVBA does not warrant, guarantee or make any representations regarding the use, or the results of the use of the software or the information contained herein. Esko Software BVBA shall not be liable for any direct, indirect, consequential or incidental damages arising out of the use or inability to use the software or the information contained herein.

The information contained herein is subject to change without notice. Revisions may be issued from time to time to advise of such changes and/or additions.

No part of this document may be reproduced, stored in a data base or retrieval system, or published, in any form or in any way, electronically, mechanically, by print, photoprint, microfilm or any other means without prior written permission from Esko Software BVBA.

This document supersedes all previous dated versions.

PANTONE®, PantoneLIVE and other Pantone trademarks are the property of Pantone LLC. All other trademarks or registered trademarks are the property of their respective owners. Pantone is a wholly owned subsidiary of X-Rite, Incorporated. © Pantone LLC, 2014. All rights reserved.

This software is based in part on the work of the Independent JPEG Group.

Portions of this software are copyright © 1996-2002 The FreeType Project (www.freetype.org). All rights reserved.

Portions of this software are copyright 2006 Feeling Software, copyright 2005-2006 Autodesk Media Entertainment.

Portions of this software are copyright ©1998-2003 Daniel Veillard. All rights reserved.

Portions of this software are copyright ©1999-2006 The Botan Project. All rights reserved.

Part of the software embedded in this product is gSOAP software. Portions created by gSOAP are Copyright ©2001-2004 Robert A. van Engelen, Genivia inc. All rights reserved.

Portions of this software are copyright ©1998-2008 The OpenSSL Project and ©1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Adobe Creative Suite, Illustrator, InDesign, PDF, Photoshop, PostScript, XMP and the Powered by XMP logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Microsoft and the Microsoft logo are registered trademarks of Microsoft Corporation in the United States and other countries.

SolidWorks is a registered trademark of SolidWorks Corporation.

Portions of this software are owned by Spatial Corp. 1986 2003. All Rights Reserved.

JDF and the JDF logo are trademarks of the CIP4 Organisation. Copyright 2001 The International Cooperation for the Integration of Processes in Prepress, Press and Postpress (CIP4). All rights reserved.
The Esko software contains the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems in the U.S. and other countries.

Part of this software uses technology by BestTM Color Technology (EFI). EFI and Bestcolor are registered trademarks of Electronics For Imaging GmbH in the U.S. Patent and Trademark Office.


All other product names are trademarks or registered trademarks of their respective owners.

Correspondence regarding this publication should be forwarded to:

Esko Software BVBA
Kortrijsesteenweg 1095
B – 9051 Gent

info.eur@esko.com
2. Introduction

This documentation is a guide for those who want to connect their Automation Engine server with other systems. Typically these systems are

- Business systems like MIS (Management Information System) or ERP (Enterprise Resource Planning). These systems usually have components like order entry, price calculation, customer database, production planning, stock etc.. These systems can be purchased by your company from an MIS vendor and then adapted to your specific business process or your company could have created her own unique system.
- Asset Management systems. They focus on managing and sharing your prepress data and are usually web-enabled.

A connection between your Automation Engine server and (one of) these systems is of course digital and can be one-directional, bi-directional or, in case more than 2 systems are connected, even triangular.

**Note:** Throughout this document, for the sake of simplification, we will name such systems ‘the external system’.

**Note:** Many features that we describe in this chapter require the optional module named Automation Engine Connect.

**Note:** A separate document is available on integrating Automation Engine with Esko WebCenter.

2.1 Why Integrate?

**Faster.**
No time is wasted by waiting, usually for a person.

**Better.**
Error reduction: no errors re-typing or misinterpreting. A small error can have huge costs. A wrong 'status' interpretation can have major production (cost) consequences.

**More consistent, more process control.**
Having a consistent way of working will benefit all. It is easier to train new people. It will be required when your company is trying to receive quality certifications (ISO etc..).

**Operators in both departments can focus on more added value tasks.**
The freed up time can now be used to increase the quality of what they do or to think their actions over. Hiring new (young) people will be easier when they see you have modern ways of working.

Now let’s have a look at what information typically is exchanged in each direction.
2.2 External System sending commands To Automation Engine

**Create an Automation Engine Job and set Job parameters.**
An Automation Engine Job represents a job order that you get from the business system. So it is logical not to create those Automation Engine Jobs manually but to get your instructions from that system in an automated digital way. All the parameters should then be picked up in a highly automated workflow.

**Create an Automation Engine Product and set Product parameters.**
New design data that arrive can be automatically added to the Automation Engine Products database, including the detailed product specifications like barcode and inks. Often, these design data arrive before there is a job order to produce them. The workflow can link this newly created product to a Job.

**Start an Automation Engine workflow with parameters defined by the external system.**
Not all parameters need to be stored first in a Job or Product: Automation Engine can also retrieve parameters from an external system during a workflow and use these parameters in subsequent steps of the workflow.

2.3 Automation Engine sending commands To an External System

**Setting Job or Product status in that external system.**
The business system is very interested in what is happening with the job in the prepress department. Being automatically informed on the exact status can have major benefits: Your sales/service people will have up to date info for your customer. Your production will be more efficient when its planning is based on reality. Automation Engine can send such feedback automatically. And based on that new status, the external system may also decide to start an own automatic workflow.

**Initiating actions on that external system.**
Automation Engine could even do more than that. It could really talk to that external application, ask questions and come back with a set of information.

2.4 Some general advice

**Have a project manager**
Appoint someone who owns the mission to do this. This integration can affect many people so someone needs to be in charge. It is actually possible that it is the first time that these 2 different departments (sales-service and prepress) are part of a same project. Each department will defend its way of working. Each department will not know all complexities of ‘the other side’.
2. Change Management

When you start eliminating some human intervention, it is possible that new questions will come to the surface. Any changes in the way you work in such a busy production environment will need to be defended. The integration is indeed technical but it is actually not 'just an IT project': the reason you do this will be of interest to higher management. Often higher management is where such ideas to integrate start. Making some changes will be necessary to get that higher efficiency that they are looking for.

2. Step by Step

Start with a step that is 'obvious' and build from there. Before 'going live', carefully test and dare to consider all sides of the changes that this integration will cause. Using both the new and the old workflow can be confusing, so be prepared to switch over fast.

2.5 Need any help?

If you want help with your integration project, Esko offers 2 levels of service.

- Standard training by an Esko Academy trainer.
  
  Esko trainers will explain the Esko part of the tools.

  Tip: After learning about the concepts and features in this documentation, we advise to also check the chapter Examples on page 72.

- Help from an Esko Solution Architect.
  
  Esko Solution Architects are experienced in such system integrations and have a broader knowledge of 'workflow', beyond the world of Esko products. Contact your Esko sales rep if you want to know more.

2.6 What about JDF?

Note: Job Definition Format is a standard language based on XML. It was created by the CIP4 group to help standardize system integration in the printing industry. Read more on [http://www.cip4.org](http://www.cip4.org)

Integrations that use JDF/JMF require assistance from an Esko Solution Architect. That is why this documentation does not cover JDF/JMF.

Note: With some specific MIS, Esko optionally offers a 'standard' integration package based on a White Paper. Some of these can be seen on [http://www.esko.com/movies](http://www.esko.com/movies). Contact your sales rep if you want to know more.
3. Integration Concepts

Let's distinguish the main concepts in integrating with Automation Engine. In many integrations, several of these concepts are combined. There is no pre-defined order in which these concepts should be used in your particular case.

The tools that we here introduce are documented in detail in the next chapters.

3.1 Access Points Triggering a Workflow on Automation Engine

Access Points trigger a workflow after they detected a signal from ‘outside’. They use a ‘polling’ mechanism: they check at a regular time interval if they have to trigger that workflow. There are 5 types of Access Points. You can create as many as you need.

A typical example is a classic hot folder (a Folder Access Point), where the incoming data is then processed and the result is output on another folder. For example: design PDFs arrive in a hot folder, from there they will be automatically trapped and then written to another folder.

However, Access points are not only used to start work on that incoming data. They can also receive parameters for the workflow that they will start. These parameters define what needs to be done in the workflow with some other (set of) file(s). For example: a Folder Access Point receives an XML from the order entry system. The XML describes a list of re-run print Jobs. And for each Job it describes which Products need to be linked into that Job and what quantity needs to be printed.
Or, instead of receiving those parameters, Access Points can go and search for them. For example: a Database Access Point looks up the desired trapping distance in an external system. This results in a SmartName the trapping task then uses.

Other types of Access Points are FTP, Mail and Web Service. Find their detailed documentation in the chapter Access Points on page 17.

It’s all about SmartNames!
In most cases, these parameters that come in from the external system are turned into SmartNames. And then these SmartNames will be picked up in the workflow tasks. These SmartNames can be stored in Job or Product parameters, when they describe job or product characteristics that are relevant for the lifetime of the job or product. When the parameters are relevant only for the workflow started by the access point, they can be used as workflow parameters.

3.2 Automation Engine tasks Interacting directly with the External System

Not only Access Points can contact an external system. There are also 2 Automation Engine tasks that are dedicated to interact with external systems. Contrary to Access Points, their interaction can be bi-directional. Besides retrieving data from external systems, these tasks can also be used to provide feedback to external systems. This will be described in more detail later. However, these 2 tasks are typically used because the workflow needs this interaction before it continues. The result of these tasks will be used in the next steps of the workflow. As usual, this can be creating Jobs or Products, or get values for SmartNames to be used in that workflow.

• The Interact with Database task
Automation Engine can use this task to **read** a value from an external database. For example: an operator starts a workflow to Step & Repeat and output. The workflow asks the database of the production planning system on which press this job will be printed. This is needed to know the layout size and also any dot gain curve. This press choice can be decided very late, so it might not have been part of the Job parameters that were created earlier.

**Note:** Reading from an external database is also possible by using a simple SmartName query or by using a Database Access Point.

Automation Engine can also use this task to **write** directly into the database of the external system. For example to inform it on the new status of a Job or Product. When that system can not read XML messages with such information, then you can consider to have Automation Engine write this information directly into the database of that system.

**Caution:** This is a very direct way to have applications connect. This must be well designed and carefully tested.

**Important:** Esko takes no liability for queries that write or change fields in that external database.

• The **Interact with Web Service** task

The concept is very similar to the one above, but here the task uses HTTP calls to a web service. Also here, both **read** (‘GET’) and **write** (‘POST’) commands are possible. When the external system is running a web service, you will be able to access it via HTTP (Hypertext Transfer Protocol). However, you also need to learn how to communicate to that system. Some applications will indeed publish their API (Application Programming Interface). For example Automation Engine can use this task to ask a question to Google Maps. This is only possible because Google Maps did publish its API (and runs a web service). This example is illustrated in the chapter **Examples** on page 72.

Typically, both these tasks execute such interactions as part of a larger workflow in Automation Engine. Find their detailed documentation in the chapter **Tasks Interacting with other Systems** on page 57.

**Note:** The **Integrate with WebCenter** task is described in **Advanced WebCenter Integration**, a chapter of the document on integrating Automation Engine with WebCenter.

### 3.3 Automation Engine Mapper tasks Transforming Data

Automation Engine offers many ways to use XML to control the workflow. Automation Engine also offers many features to create XML files. But sometimes you need to create an XML with a different content or formatting.

This **Create XML File** task is typically used to create XML files that are to be read by external systems. In such XML files, you sometimes want more or less information. And sometimes you need to reformat data in a way that the external system prefers.

Then there are the specific data formatting tasks like **Split XML**, **Join XML** or **Map Data** task. These are powerful tools that offer many more ways to control XML files. Sometimes the data (information)
that comes from an external system first needs to be transformed before it can be used in Automation Engine. Sometimes, these transformations are classic file conversions like CSV to XML. The Map Data task was created to do this. Sometimes the XML will first need to be split in several XMLs (Split XML task). Sometimes it will be useful to join several data files into one XML (Join XML task).

Data transforming tasks can be used at any step in a workflow. They are used

- at the start of a workflow, to help transform what an Access Point delivered.
- at any other point in a workflow, also often at the end, to translate information into a format that the external system prefers.

Find a detailed description of these tasks in the chapter Data Transforming Tasks on page 37).

3.3.1 Transforming data from an Access Point

Here are some examples where such data transforming tasks are used right at the beginning of a workflow that is started by an Access Point.

For example:

- The external system provides a list of jobs to be processed under the form of a large XML that it writes on the Folder Access Point. The Split XML file task then creates 1 XML file per job entry. These separate XML files are then used for the Create Job task.

- The external system sends an CSV file on a Folder Access Point. The Map Data task first transforms it into an XML. Later in the workflow, SmartNames will get their values by reading from that XML file (Xpath queries).

- Some Access Points do not even result in data files yet. In case of a Database Access Point or a Web Service Access Point, the accessed data can arrive as a several XML files. This is where you can use the Join XML task to create a single and maybe even simpler XML file. Again, that XML file is then typically the base to provide SmartNames their workflow specific value.
3.3.2 Transforming data from a workflow

There are many moments where it can be necessary to transform Automation Engine information. Typically when sending information to external systems. Some of these external systems have their own mapping tools to read these incoming XMLs anyway. But sometimes you will be asked to have Automation Engine transform them first.

Some examples:

- The MIS can not read an XML file but can read CSV. This is a classic use case for the Map data task.

- The XMP (metadata) of a PDF needs to be sent to the external system as an XML. The LinkEdge task does that. But maybe the MIS wants to have the ink coverage information formatted in a different way. And maybe it is also necessary to convert the millimeters into inches.

- When using the Interact with Web Service task to ask the web service of Google Maps the distance to a customer, the answer needs to end up in an XML or a SmartName.

- A Step & Repeat task requires a specific XML as input file, with a different structure than the one from the external system.

- The Create Job task requires information that is present in 3 XMLs, from 3 different sources.

- The XML with feedback that you want to send to the external system needs to contain both a Job status and some meta data from the PDF.

3.4 Automation Engine sending data Back to the External System

We briefly described this topic in the introduction (Automation Engine sending commands To an External System on page 9). Here, we add some detail and mention some tools.

A workflow in Automation Engine always generates output data, and often even many kinds. A lot of that data is not just for other prepress components, but for external systems. We distinguish 2 main types of data:

- **Sending Prepress data (back)**
  - Data for output devices like proofers and RIPS (CTP, DFE, Press)
  - Data for people, often customer service or sales people or end-customers (PDFs attached in e-mail or sent via FTP)
  - Data for external systems, some of which web-enabled, often to view, approve or exchange 'assets'. In many cases it is not the original production files that are sent but smaller versions (less Mb) or more secure versions like password protected PDFs (JPEGs, RGB-PDFs, viewer streaming data, 3D files...)

- **Sending Communication data (back)**. Here, the goal is to inform. Some communication will be about a status, for example 'plates made'. Some will be about describing the (new) prepress data themselves, for example the ink coverage of a production PDF. This communication data can also be files or it could be 'live':
• Communication files: XML, XMP-XML, CSV, TXT
  • written on a network share, typically on a hot folder of the external system
  • sent via e-mail or FTP

• Communicating directly with the external system: (see also Automation Engine tasks Interacting directly with the External System on page 12.)
  • directly to its database, writing or updating fields
  • directly to the web service of the API of that external system

Transforming data before sending
Sometimes the external system does not support one of the above file formats or interaction protocols. In those cases, the Map Data task can help to convert. Find an introduction to this concept in Automation Engine Mapper tasks Transforming Data on page 13.

Tip: You do not always need to send a lot of Mb to send a lot of information. For example: an XML describing de bar code parameters can be more useful than a high-res PDF. You could zoom-in on the PDF but the XML will tell you even the non-visible parameters of that bar code (type, bar width reduction, distortion...)
4. Access Points

An Access Point launches a workflow after it detected a signal from 'outside'. Automation Engine offers 5 types of Access Points. The Folder Access Point is the most classic example and is also often referred to as a 'hot folder'.

Tip: We advise you to first read the introduction about the concept of Access Points in Access Points Triggering a Workflow on Automation Engine on page 11.

The Access Points Tool and View

In your Pilot, the Access Points Tool is where you create, modify and have an overview of your Access Points.

The top part of this view shows a list of all your Access Points.

The bottom part is a task monitor that shows the tasks that were executed by the selected Access Point.

Working with Access Points (the basics)

We here list the main functionality that is available in tool bar and also via the File menu. The next chapters will add detail per type of Access Point.

- 📄 to create a new Access Point.
- 📂 to duplicate an Access Point. The selected Access Point will open with empty fields for Name and Description.
- ✗ to delete a selected Access Point. You will be asked to confirm.
- 📊 to 'scan now'. To decide manually when the selected Access Point should scan for new files. This can be useful when you do not want to wait for the time interval (see further).
Note: This function is disabled for the Web Service Access Point.

To **modify** an existing Access Point, select it from the list and double click it to open and modify. To **deactivate** an Access Point, right-click it and choose Deactivate Access Point.

**Differences in how they work**

This table illustrates how some Access Points work in a slightly different way:

<table>
<thead>
<tr>
<th>Access Point Type</th>
<th>Checks for data at regular time interval ('Polling')</th>
<th>Uploads that data</th>
<th>Starts a workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder</td>
<td>V</td>
<td>V</td>
<td>typically</td>
</tr>
<tr>
<td>FTP</td>
<td>V</td>
<td>V</td>
<td>typically</td>
</tr>
<tr>
<td>Mail</td>
<td>V</td>
<td>V</td>
<td>typically</td>
</tr>
<tr>
<td>Database</td>
<td>V</td>
<td>- (**)</td>
<td>typically</td>
</tr>
<tr>
<td>Web Service</td>
<td>- (*)</td>
<td>- (***)</td>
<td>always</td>
</tr>
</tbody>
</table>

(*) The external system decides when to contact the Access Point
(**) **goes to and gets info from** an external database.
(***) **waits** until it **receives** an HTTP command from the external system.

**Access Point SmartNames**

There is a specific category of SmartNames for Access Points. They enable smart re-using of many attributes of an Access Point. All Access Point SmartNames are also automatically stored as workflow parameters.

For more information on SmartNames, see the dedicated chapter **SmartNames**.

### 4.1 Folder Access Point

#### 4.1.1 Concept

**What is a Folder Access Point?**

A **Folder Access Point** monitors a folder located somewhere in an Automation Engine **Container**. Every time one or more files are uploaded into that folder, the **Folder Access Point** will move the files to another folder where they will be processed. **Folder Access Points** are typically created to launch a workflow using those incoming files as input files for that workflow.

However, you can also choose not launch a workflow. In that case the Access Point only serves to upload files onto a specific folder in an Automation Engine **Container**.

**Tip:** The chapter **Examples** on page 72 contains an example case of a **Folder Access Point**.
Note: **Folder Access Points** can also be used to process incoming JDF files. For more info on using JDF, read more in *What about JDF?* on page 10.

### 4.1.2 What about Hot Folders?

Before version 14 of Automation Engine, the user could create 'Hot Folders'. When you upgrade to version 14, they will **not** be automatically migrated to **Folder Access Points**. They will stay Hot Folders and will still function as they did before.

If you did create such Hot Folders in a version older than version 14, you will also keep your **Hot Folders View** in the Pilot of your version 14. If you haven’t created Hot Folders in Automation Engine before upgrading to version 14, you will not see the Hot Folders view in the Pilot.

The following options were previously available in Hot Folders but are not available in **Folder Access Points**. This is because there are now other and better ways to do these actions:

- the annotation for the workflow task (you now write a ‘description’)
- grouped launching (you now can still create the effect by using a ZIP, but the most typical use cases are covered when you use the dedicated ‘gang run’ related features)
- delete input files when the workflow is finished (this can be done with the **Delete File** task at the end of your workflow)

### 4.1.3 Creating or modifying a Folder Access Point

Follow these steps to create or modify **Folder Access Points**:

1. In the **Pilot** click on **Access Points** view. Alternatively, choose **Tools > Access Points**.
2. In the dedicated **Access Points** view, click on **Folder**. Alternatively, choose **File > New**.
3. In the resulting window, select **Folder** and click **OK**. The **New Folder Access Point** window will open. Here is one with some typical settings already filled in:

![New Folder Access Point](image)

4. Enter a **Name** for the **Access Point**.
5. If you want, enter a **Description**. This will help identifying your **Access Point** without having to read all its settings.
6. In the tab **Folder Settings**, define which folder will be the actual Access Point that will be monitored by Automation Engine. Use the browse button to indicate it.  
This folder needs to be **inside** a **Container**. This folder may not be inside another **Folder Access Point**.

7. In the tab **Monitoring**, define the time interval.
   - **Check Every.** Enter the time interval with which Automation Engine should check if any new files came in. Think of the time interval as the maximum time it takes for the new files to be picked up.
   - **Schedule.** If you want this Access Point to be inactive during a part of the day, then activate this option and indicate the start and end time of that period.

   **Note:** Such a temporary inactive state will not be shown in the general Access Points View. In that View, the column 'Active' shows its general status.

8. In the tab **Automatic Actions**, define what needs to happen.
   - **Move Files to**. This is the folder where the files will be processed. If you want, use **SmartNames**. These files have to be moved to avoid that the **Access Point** would process the same files again and again.
   - **Rename Files to**. If you want you can choose to rename these files too. To do this, enter the new file name.

   **Tip:** Typically, SmartNames are used here to help indicate in their name when they were processed or where they came from.

   - **Action.** Choose:
     - **None.** Choose this option when you only want to use this **Access Point** to move the files.
     - **Workflow.** Choose this option when you want a workflow to start. Select the workflow ticket from the offered list. The new accessed files will be used as input files for this workflow.

   **Attention:** Make sure your workflow **can** start with the type of files that will arrive.

   **Options:**
   - **Operator:** choose the operator that will be shown as operator for the workflow task.

     **Tip:** If you do not want to assign one specific person, you could create an operator named "Access Points" or "Automatic" and assign that one here. Also remember that the Access Points Task Monitor shows you the tasks launched by the selected Access Point.

     **Note:** If you leave this option blank, a system user "UPLC" will be used.

   - **Priority:** choose the priority for this workflow: Low, Normal or High. The default priority is 'Normal'.

   **Options:**
   - **Operator:** choose the operator that will be shown as operator for the workflow task.

     **Tip:** If you do not want to assign one specific person, you could create an operator named "Access Points" or "Automatic" and assign that one here. Also remember that the Access Points Task Monitor shows you the tasks launched by the selected Access Point.

     **Note:** If you leave this option blank, a system user "UPLC" will be used.

   - **Priority:** choose the priority for this workflow: Low, Normal or High. The default priority is 'Normal'.
• **Process JDF.** Choose this option when this Access Point will be used to process incoming JDF files. For more info on JDF, read this section: *What about JDF?* on page 10.

### 4.2 FTP Access Point

#### 4.2.1 Concept

An FTP Access Point monitors a folder on an FTP site. At a regular time interval, it will check if new files have arrived in that folder and it will move them to a designated folder in a Container of Automation Engine.

**Note:** There is no FTP server embedded in Automation Engine, so this functionality relies on a FTP server that has been setup separately from Automation Engine.

This also works for new files in any subfolders of the scanned FTP site. Those files will be moved to an identical structure of subfolders that will be automatically created.

The FTP Access Point can also start a workflow on those files.

After uploading them, the Access Point deletes these files from the FTP site. However, the folders (and subfolders) are not deleted. This way, the folder structure of FTP site stays intact.

**Note:** The Pilot also offers a manual way to upload files from an FTP site. When you right-click on a folder or Job Folder, the function *Upload from FTP site* is offered. This FTP Access Point is an automated version of that feature.

#### 4.2.2 Creating or modifying an FTP Access Point

Follow these steps to create or modify FTP Access Points:

1. In the Automation Engine Pilot go to the Access Points View. Alternatively, choose Tools > Access Points.
2. In the dedicated Access Points View, click on . Alternatively, choose File > New.
3. In the resulting window, select FTP and click OK.

The New FTP Access Point window will open. Here is one with some typical settings already filled in:
4. Enter a **Name** for the **Access Point**.

5. If you want, enter a **Description**.
   
   This will help identifying your **Access Point** without having to read all its settings.

6. In the tab **FTP Settings**, define where and how you wish to connect.
   
   - **Server**: Enter the name of the FTP server.
   
   - **File Transfer**: Choose either **Binary** or **ASCII**.
     
     **Tip**: **Binary** will always work, but if you have an ASCII file, the transfer speed will be higher when you choose **ASCII**.
   
   - **Connection Mode**:
     
     Select **Passive** if there is a firewall between the FTP server and Automation Engine that is blocking the FTP server from setting up connections to Automation Engine.
     
     **Tip**: By default, the FTP transfer uses Active Connection Mode (where the FTP server opens the data connection). When **Passive** is selected, the system will use Passive Connection Mode instead (where the client initiates the connection).
   
     - **Secure FTP** if you are connecting to a secure FTP server (over FTPS, not SSL or SSH). If you are experiencing problems using secure FTP, try connecting over regular FTP and ask your IT administrator to check the security settings of your FTP server.
   
   - **User Name** and **Password**: Enter a valid user name and password for this FTP server.
• **Folder:** This is the folder on the FTP server that will be monitored. As mentioned in *Concept* on page 21, also its sub folders will be monitored.

• **Customer:** If you want to link a specific customer to this Access Point, use *Select* to pick it from the list. Then you can use it as a SmartName.

7. In the tab **Monitoring**, define the time interval.

• **Check Every.** Enter the time interval with which Automation Engine should check if any new files came in. Think of the time interval as the *maximum* time it takes for the new files to be picked up.

• **Schedule.** If you want this Access Point to be inactive during a part of the day, then activate this option and indicate the start and end time of that period.

  **Note:** Such a temporary inactive state will not be shown in the general Access Points View. In that View, the column 'Active' shows its *general* status.

8. In the tab **Automatic Actions**, define what needs to happen.

• **Move Files to.**
  
  • **Folder.** Indicate where you want the files to be moved to. Browse to a folder in a *Container*. If you want, use *SmartNames*.
  
  • **Customer's Upload Folder.** Alternatively, you can choose to move these files to the *Customer's Upload Folder*. This is a folder you can define in the definition of the Customer.
  
  • **Rename Files to.** If you want you can choose to rename these files too. To do this, enter the new file name.

  **Tip:** Typically, SmartNames are used here to help indicate in their name *when* they were processed or *where* they came from.

• **Action.** Choose:
  
  • **None.** Choose this option when you *only* want to use this *Access Point* to move the files.
  
  • **Workflow.** Choose this option when you also want a workflow to start. Select the workflow ticket from the offered list. The new accessed files will be used as input files for this workflow.

  **Attention:** Make sure your workflow *can* start with the type of files that will arrive.

**Options:**

• **Operator:** choose the operator that will be shown as operator for the workflow task.

  **Tip:** if you do not want to assign one specific person, you could create an operator named "Access Points" or "Automatic" and assign that one here. Also remember that the Access Points Task Monitor shows you the tasks launched by the selected Access Point.
• **Priority**: choose the priority for this workflow: Low, Normal or High. The default priority is 'Normal'.

• **Process JDF**: Choose this option when this Access Point will be used to process incoming JDF files. For more info on JDF, read this section: *What about JDF?* on page 10.

### 4.3 Mail Access Point

#### 4.3.1 Concept

A **Mail Access Point** scans an e-mail account and extracts the e-mail attachments of new incoming e-mail. These files are uploaded to a folder in a **Container**. A **Mail Access Point** can also start a workflow on those files.

Here are 2 typical examples of how **Mail Access Point** SmartNames can help you do even more:

- You can agree with the e-mail sender to start the subject line with the Job number. This helps you direct these files to the right folder.
- The address of the e-mail sender can be used to later send him an e-mail back about the status of the files he sent.

#### 4.3.2 Create or modifying a Mail Access Point

Follow these steps to create or modify **Mail Access Points**:

1. In the **Pilot** click on **Access Points** view. Alternatively, choose **Tools > Access Points**.
2. Click on ![Create Access Point](image.png) to create a new **Access Point**. Alternatively, choose **File > New**.
3. In the resulting window, select **Mail** and click **OK**. The **New Mail Access Point** window will open up. Here is one with some typical settings already filled in:
4. Enter a Name for the Access Point.
5. If you want, enter a Description.
   This will help identifying your Access Point without having to read all its settings.
6. Define the Mail Settings:
   - **Incoming Mail Server**: Enter the name of the mail server.
   - **Account Type**: Depending on the mailbox configuration, choose either POP or IMAP.
   - **Secure Mail**: Select this if the mailbox is a secure type.
   - **User Name** and **Password**: Enter the user name and password of the mail account that will receive the e-mails that you want to access.
   - **Customer**: If you want to link a specific customer to this Access Point, use Select to pick it from the list. You can then use it as a SmartName.
7. In the tab Monitoring, define the time interval.
   - **Check Every**: Enter the time interval with which Automation Engine should check if any new files came in. Think of the time interval as the maximum time it takes for the new files to be picked up.
   - **Schedule**: If you want this Access Point to be inactive during a part of the day, then activate this option and indicate the start and end time of that period.
   **Note**: Such a temporary inactive state will not be shown in the general Access Points View. In that View, the column 'Active' shows its general status.
8. In the tab Automatic Actions, define what needs to happen.
• Move Files to.
  • **Folder.** Indicate where you want the files to be moved to. Browse to a folder in a **Container.** If you want, use **SmartNames.**
  • **Customer's Upload Folder.** Alternatively, you can choose to move these files to the **Customer's Upload Folder.** This is a folder you can define in the definition of the Customer.
  • **Rename Files to.** If you want you can choose to rename these files too. To do this, enter the new file name.

  **Tip:** Typically, SmartNames are used here to help indicate in their name *when* they were processed or *where* they came from.

• **Action.** Choose:
  • **None.** Choose this option when you only want to use this **Access Point** to move the files.
  • **Workflow.** Choose this option when you want a workflow to start. Select the workflow ticket from the offered list. The new accessed files will be used as input files for this workflow.

  **Attention:** Make sure your workflow **can** start with the type of files that will arrive.

**Options:**
  • **Operator:** choose the operator that will be shown as operator for the workflow task.

  **Tip:** if you do not want to assign one specific person, you could create an operator named "Access Points" or "Automatic" and assign that one here. Also remember that the Access Points Task Monitor shows you the tasks launched by the selected Access Point.

  • **Priority:** choose the priority for this workflow: Low, Normal or High. The default priority is 'Normal'.

  • **Process JDF.** Choose this option when this Access Point will be used to process incoming JDF files. For more info on JDF, read this section: *What about JDF?* on page 10.

---

### 4.4 Database Access Point

#### 4.4.1 Concept

A **Database Access Point** reads, at a regular time interval, information from an external database. That information is written into one or more XML files. The **Database Access Point** can also start a workflow.
A typical use case is checking an external system for a new status. That new status will be a reason for Automation Engine to start an action. For example: in that external database, when the status of a Job or Product is found on "preflight required", Automation Engine starts a preflight workflow. Or the output tasks will be started for all jobs that are found in a status "Jobs to expose".

**Tip:** The chapter *Examples* on page 72 contains an example case of a **Database Access Point**.

To prevent that these statuses keep triggering these actions over and over again, the **Database Access Point** offers the choice to update or delete those (status) fields. For example: the status "Proof required" would be updated to "Proof started on Esko".

These use cases are often identical to those where an external system sends an XML file to Automation Engine. Here are some typical reasons to use a **Database Access Point** instead of such an XML transfer:

- The external system can not export the wanted data to XML. Or it would require an expensive intervention or an upgrade of the software version.
- Exporting an XML would be a manual action by a user of that external system. And all manual actions can be forgotten.

### 4.4.2 Create or modifying a Database Access Point

Follow these steps to create or modify a **Database Access Point**:

1. In the **Pilot** click on **Access Points** view. Alternatively, choose **Tools > Access Points**.
2. Click on ![new](image) to create a new **Access Point**. Alternatively, choose **File > New**.
3. In the resulting window, select **Database** and click **OK**.
   - The **New Database Access Point** window will open up. Here is one with some typical settings already filled in:
4. Enter a **Name** for the Access Point.

5. If you want, enter a **Description**.
   This will help identifying your **Access Point** without having to read all its settings.

6. Define the **Database Settings**:
   - **External Database**: Select the database from the list. The list shows the **External Databases** that you configured via **Tools > Configure**.
   - Enter the desired **SQL Select Query**: The query will search for records or it will search specific values in one or more columns. An XML file will be generated for each record that matches the query.

Here is an example of such an XML file. The query was looking for more info on Jobs with the status 'New'. This XML shows the resulting details of the Job 1213. See how the 'Select' returned all the columns of the row of this Job 1213: name, operation etc...

```xml
<?xml version="1.0" encoding="UTF-8?>
<design select="SELECT * FROM design WHERE status = 'NEW'"
producer="DBPoller-MIS-1" created="2013-04-24T11:26:14.210+02:00">
  <id>1213</id>
  <name>MyDesign</name>
  <operation>CreateJobCard</operation>
  <width>12</width>
  <height>500</height>
  <depth>50</depth>
  <status>NEW</status>
</design>
Note: As soon as you select a **Database Access Point**, the query will be checked if it is valid. If it is not valid, an error message will be shown.

- **After Processing**: Now choose what should happen after the results from the query were processed.
- **Delete the records**: This will delete the records that the query selected in the external database. It will make them empty.
- **Update the records**: This will update the records that the query selected in the external database with a new value. Click on **Add** to update the desired parameters. Fill in the **Column Name** and the **New Value** in that database column and click OK.

Note: If you want to update a record that contains a date and/or time, use the format `yyyyMMddHHmmss` for the new value of the column or use **now** to insert the date and time of the moment that the record is updated.

7. In the tab **Monitoring**, define the time interval.
   - **Check Every**: Enter the time interval with which Automation Engine should execute this database query.
   - **Schedule**: If you want this Access Point to be inactive during a part of the day, then activate this option and indicate the start and end time of that period.

Note: Such a temporary inactive state will not be shown in the general Access Points View. In that View, the column ‘Active’ shows its general status.

8. In the tab **Automatic Actions**, define what needs to happen.
   - **Move Files to**: Indicate where you want the resulting XML file(s) to be written. Browse to a folder in a **Container**. If you want, use **SmartNames**.
   - **Rename Files to**: If you want you can choose to rename these files too. To do this, enter the new file name.

Tip: Typically, SmartNames are used here to help indicate in their name when they were processed or where they came from.

- **Action**: Choose:
  - **None**: Choose this option when you want no further action.
  - **Workflow**: Choose this option when you also want a workflow to start. Select the workflow ticket from the list.

  **Attention**: Make sure your workflow finds its input files. These can be the XML files from the query but that is not required.

**Options**:
- **Operator**: choose the operator that will be shown as operator for the workflow task.
Tip: if you do not want to assign one specific person, you could create an operator named "Access Points" or "Automatic" and assign that one here. Also remember that the Access Points Task Monitor shows you the tasks launched by the selected Access Point.

- **Priority**: choose the priority for this workflow: Low, Normal or High. The default priority is 'Normal'.

### 4.5 Web Service Access Point

#### 4.5.1 Concept

When a system has a web service, other software applications can send commands or queries to that system using HTTP requests to the URL of the web service.

**Note**: Some users prefer this way of direct communication rather than exchanging (XML) files on hot folders.

Automation Engine also has a web server. A **Web Service Access Point** is a way for external systems within the intranet to start workflows on Automation Engine by sending a command to the web service of Automation Engine.

**Tip**: In **this section on differences between Access Points**, you can read how a **Web Service Access Point** works somewhat different than the other types of Access Points. One important difference is that the **Web Service Access Point** does not use the concept of time intervals. It is the external system that fully decides when to send a command to Automation Engine.

The HTTP requests can be both HTTP POST requests and HTTP GET requests. If the HTTP POST request also 'posts' data files, then they will be used as input files for the workflow that will be started.

**Note**: When using a HTTP GET request, parameters of the web service call are encoded in the URL using HTTP query strings. When using HTTP POST requests, the parameters are encoded in the content part of the HTTP request.

#### 4.5.2 Checking the Automation Engine Web Service

The web service of Automation Engine will be running automatically on the Automation Engine server.

**Note**: You actually also use this web service when you access the Automation Engine **Client Apps** in a browser.

You can check the status of the Automation Engine Web Service by entering this in a browser: `http://<name of your server>:<port>/ws/`. The port number is the one as defined in the **Configure** tool > **Automation Engine Web Service**. By default this port number is 4415.
For example: if your Automation Engine server is named AESERVER1, then the URL to enter in your browser is http://aeserver1:4415/ws/

The web page that then appears offers more guidelines on formatting more detailed HTTP commands.

### 4.5.3 Creating a Web Service Access Point

**Note:** Setting up a Web Service Access Point is simple. The complexity lies in the content of the HTTP requests that are sent to it (see next chapters).

Follow these steps to create or modify a Web Service Access Point:

1. In the Pilot click on Access Points view. Alternatively, choose Tools > Access Points.
2. Click on ![New Access Point](image) to create a new Access Point. Alternatively, choose File > New.
3. In the resulting window, select Web Service and click OK.
   The New Web Service Access Point window will open up. Here is one with some typical settings already filled in:

   ![New Web Service Access Point](image)

4. Enter a Name for the Access Point.
This name will automatically be used to complete the URL for this Access Point.

5. If you want, enter a **Description**. This will help identifying your **Access Point** without having to read all its settings.

6. See the resulting URL. This is where the external system needs to send its command to launch the workflow defined below.

7. Select the **Workflow** that this Access Point will start.

   **Options**:
   - **Operator**: choose the operator that will be shown as operator for the workflow task.
     
     **Tip**: If you do not want to assign one specific person, you could create an operator named "Access Points" or "Automatic" and assign that one here. Also remember that the Access Points Task Monitor shows you the tasks launched by the selected Access Point.
   - **Priority**: choose the priority for this workflow: Low, Normal or High. The default priority is 'Normal'.

4.5.4 HTTP POST Requests

**Launching a workflow using an HTTP POST request**

The HTTP POST requests must meet the following requirements:

- The value of the **Content-Type** header of the HTTP POST request should be, in most cases, **multipart/form-data**.
- HTTP request parts can be added to the HTTP POST requests to attach data files and/or to add parameters (also see further).
- For every HTTP request part that is added to an HTTP POST request, a **Content-Disposition** header should be present.
  - For each data file, the (UTF-8 encoded) file name of the data file should be specified in the corresponding **Content-Disposition** header.
  - For each parameter, the name of the parameter should be specified in the corresponding **Content-Disposition** header.
- The encoding of the HTTP POST requests should be **base64** or no encoding (for example **binary**, **8bit**, ...).

For example:

```plaintext
POST /ws/myFavoriteWorkflow HTTP/1.1
Host: aeserver01:4415
Content-Type: multipart/form-data;
boundary=-----------------------------146677a63ef
Content-Length: <length of request body in octets (8-bit bytes)>
...
-----------------------------146677a63ef
Content-Disposition: form-data; name="wfparam1"
2mm
-----------------------------146677a63ef
Content-Disposition: form-data; name="wfparam2"
```
Attaching data files to be used as input files for the workflow that is launched

You can attach data files to be used as input files for the workflow that is launched by adding them as HTTP request parts to the HTTP POST request. For each data file, the (UTF-8 encoded) file name of the data file should be specified in the corresponding `Content-Disposition` header.

For example:

```
POST /ws/myFavoriteWorkflow HTTP/1.1
Host: aeserver01:4415
Content-Type: multipart/form-data; boundary=----------------------146677a63ef
Content-Length: <length of request body in octets (8-bit bytes)>

----------------------146677a63ef
Content-Disposition: form-data; name="inputfile1";
filename="inputfile1.pdf"
Content-Transfer-Encoding: binary

<binary contents of inputfile1.pdf>
----------------------146677a63ef
Content-Disposition: form-data; name="inputfile2";
filename="inputfile2.pdf"
Content-Transfer-Encoding: binary

<binary contents of inputfile2.pdf>
```

When the workflow is launched, the data files are handled in one of the following ways:

- When the workflow is launched in a specific Job, the data files will be moved to the Job Folder.
- When the workflow is not launched in a specific Job and the workflow is successfully finished, the data files will be deleted.
- When the workflow is not launched in a specific Job and the workflow did not finish successfully, the data files will be kept in this shared folder:

```
\<name of your server>\AutomationEngineTmpFolder
```

When no data files are attached to the HTTP POST request, the **Web Service Access Point** will generate an XML file containing some information about the HTTP GET request. This XML file will then be used as input file for the workflow to be launched and will be handled in the same way as attached data files would be handled.

### Launching a workflow in a specific Job

You can specify the Job in which the workflow should be launched by adding parameters as HTTP request parts to the HTTP POST request. The name of each parameter should be specified in the corresponding `Content-Disposition` header.
The Job in which the workflow should be launched can be specified using the **Job ID**, the **Job Name**, the **Order ID** or using both the **Order ID** and the **Sub Order ID**. The parameters you need to add as HTTP request parts to the HTTP POST request are, respectively, **jobid**, **jobname**, **orderid** or **orderid** and **suborderid**. The values for these parameters should be UTF-8 encoded.

For example:

```plaintext
POST /ws/myFavoriteWorkflow HTTP/1.1
Host: aeserver01:4415
Content-Type: multipart/form-data;
boundary=----------------------146677a63ef
Content-Length: <length of request body in octets (8-bit bytes)>
...
----------------------146677a63ef
Content-Disposition: form-data; name="orderid"
2014-100254
...
```

### Specifying the values of the workflow parameters of the workflow that is launched

You can also set the values of the workflow parameters of the workflow by adding parameters as HTTP request parts to the HTTP POST request. These parameters should have the same names as the names of the workflow parameter(s) which value you want to set. The values for these parameters should be UTF-8 encoded. For each parameter, the name of the parameter should also be specified in the corresponding **Content-Disposition** header.

For example:

```plaintext
POST /ws/myFavoriteWorkflow HTTP/1.1
Host: aeserver01:4415
Content-Type: multipart/form-data;
boundary=----------------------146677a63ef
Content-Length: <length of request body in octets (8-bit bytes)>
...
----------------------146677a63ef
Content-Disposition: form-data; name="wfparam1"
2mm
...
```

### 4.5.5 HTTP GET Requests

#### Launching a workflow using an HTTP GET request

When using HTTP GET requests to launch workflows on a **Web Service Access Point**, no data files can be attached.

Every time an HTTP GET request is received on the URL of the **Web Service Access Point**, it will generate an XML file containing some information about the HTTP GET request. This XML file can then be used as input file for the workflow to be launched.

- When the workflow is launched in a specific Job, this XML file will be moved to the Job Folder.
- When the workflow is not launched in a specific Job and the workflow is successfully finished, this XML file will be deleted.
• When the workflow is not launched in a specific Job and the workflow did not finish successfully, this XML file will be kept in this shared folder:

```
\<name of your server>\AutomationEngineTmpFolder
```

**Launching a workflow in a specific Job**

You can specify the Job in which the workflow should be launched by adding parameters to the query string of the HTTP URL. The Job in which the workflow should be launched can be specified using the **Job ID**, the **Job Name**, the **Order ID** or using both the **Order ID** and the **Sub Order ID**. The parameters you need to add to the query string of the HTTP URL are, respectively, **jobid**, **jobname**, **orderid** or **orderid** and **suborderid**. The values for these parameters should be UTF-8 encoded.

For example:

```
http://aeserver01:4415/ws/myFavoriteWorkflow?jobid=ee6b08ca-eecf-4cb7-b2e1-254c20d82546
http://aeserver01:4415/ws/myFavoriteWorkflow?
orderid=2014-100255&suborderid=001
```

**Specifying the values of the workflow parameters of the workflow that is launched**

You can also set values for workflow parameters by adding parameters to the query string of the HTTP URL. These parameters should have the same names as the names of the workflow parameter(s) which value you want to set. The values for these parameters should be UTF-8 encoded.

For example:

```
http://aeserver01:4415/ws/myFavoriteWorkflow?wfparam1=2mm
http://aeserver01:4415/ws/myFavoriteWorkflow?wfparam1=2mm&wfparam2=4mm
http://aeserver01:4415/ws/myFavoriteWorkflow?jobname=Libelle-2014-week12&wfparam1=2mm
```

**4.5.6 Error Handling**

When using this **Web Service Access Point**, the corresponding HTTP responses should always be checked because they can contain error messages.

In case the workflow was successfully launched, the HTTP response will contain the task ID of the workflow task that was launched.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
  <response>taskid=dd145a15-49bb-4625-a0da-6d3ae6e63ae3</response>
</message>
```

In case the workflow could not be launched successfully and/or an error occurred, the HTTP response will contain one or more error messages.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
```
<error>
  <code>-110</code>
  <msg>Problem loading ticket: swft/Web Service Access Point Test WF</msg>
</error>
</message>
5. Data Transforming Tasks

We introduced the concept of these tasks in the chapter Automation Engine Mapper tasks Transforming Data on page 13.

**Attention:** These types of tasks require good knowledge of SmartNames.

**Attention:** The XPath Builder makes it a lot easier to extract parts from XML files. This tool is documented in a separate chapter on SmartNames.

5.1 Create XML File task

The Create XML File task is typically used to export value from SmartNames to an external system.

**Important:** This task does not use its task input file to start building the XML from. The content of the XML that you will create is completely defined inside the task window.

**Defining the content of the XML**

The Create XML File task offers a blank canvas where you need to define all the content of the XML that you want to create. In below example you see a combination of self written XML code (blue) and some SmartNames (green). Click on + [ ] insert SmartName to pick a SmartName from the list. Alternatively, right-click and choose a SmartName from the offered categories.
Note: The XML file has to start with an XML declaration. This line indicates the file encoding to the reading application. To add such a line, you can simply right-click and choose **Insert XML Declaration** to insert this industry default one:

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

Note: Any links to externally defined XML schema (for example a reference to a DTD or XSD) may influence performance. Preferably remove such links from the XML file.

- **Replace reserved characters in SmartName values with XML entities**: It is possible that the SmartName returns characters that are not allowed in XML. With this option, you choose to map these special characters mapped into these ‘XML entities’. Below list shows these mappings:

<table>
<thead>
<tr>
<th>Reserved characters in result value</th>
<th>XML Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>`</td>
<td>'</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
</tbody>
</table>

- **Task fails when output XML is not a valid XML file**: Enable this option to have Automation Engine check the validity of the XML. If this option is on, the task will fail if the XML is not valid. The validation checks the XML formatting and the XML tag endings.

Note: This task does not check XML schemas such as XSD, DTD, etc.
5.2 Split XML Task

5.2.1 Concept

Sometimes you want to split XML files to make them more easy to use Automation Engine workflows. These XML files typically come from an external system. The Split XML task needs 1 XML file as input and will write 1 or more XML files as output. Some typical examples:

- Every morning your MIS system sends you a list with new Jobs and their details. To use that XML for the Create Job task, you need to split it into 1 XML per Job.
- Your MIS system sends an XML per customer order, but your workflows are set up per production item of an order. In this case, the Split XML task helps you to generate an XML file per production item. This makes it easier to automate your workflow. This example is elaborated below.

**Tip:** You can also use this task to split an XML that was created by Automation Engine itself. For example to make that XML more manageable for an external system.

**Supported input files**
The input of this task is XML. Files with a different extension than .xml will not be recognised or processed.

**Note:** Exception: JDF files are also allowed as input. The resulting output will be XML files. These XML files will no longer be valid as JDF file.

**How do you decide what to split?**
How you split the task’s input file is defined by XPath expressions. The built-in XPath Builder will help you to do write those expressions. See further.

5.2.2 Example and Options

Example: The following code is grouping 2 production items in 1 XML file:

```xml
<?xml version="1.0"?>
<job>
  <commoncomponent>Z</commoncomponent>
  <commoncomponent content="Y" />
  <productionitem>
    <subelement content="A" />
    <subelement>1</subelement>
  </productionitem>
  <productionitem>
    <subelement content="B" />
    <subelement>2</subelement>
  </productionitem>
</job>
```

To split each production item into a separate XML file, set this expression in the XPath canvas:
This will generate these 2 files:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<job>
  <commoncomponent>Z</commoncomponent>
  <commoncomponent content="Y" />
  <productionitem>
    <subelement content="A" />
    <subelement>1</subelement>
  </productionitem>
</job>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<job>
  <commoncomponent>Z</commoncomponent>
  <commoncomponent content="Y" />
  <productionitem>
    <subelement content="B" />
    <subelement>2</subelement>
  </productionitem>
</job>
```

**Note:** The SmartName [File Number] is available for this task only. It automatically generates a unique file name.

---

**Include the parent XML structure**

This option will preserve all XML content that the XPath expression does not select.

**Note:** In case of a conditional expression this will also include siblings that do not match the condition.

If you only want to keep the parts that you selected with the XPath expression, then deselect this option. Using the previous example, this will result in following XML files:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```
When an input file contains a 'namespace declaration' in the parent XML structure ('xmlns'), then that namespace declaration will be added as an attribute of the top level element. For example:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<productionitem>
  <subelement content="A" />
  <subelement>1</subelement>
</productionitem>

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<productionitem>
  <subelement content="B" />
  <subelement>2</subelement>
</productionitem>
```

Then the split result with namespace declaration on the top level element will be:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<esko:productionitem xmlns:esko="http://www.esko.com/myowns/">
  <esko:subelement content="A" />
  <esko:subelement>1</esko:subelement>
</esko:productionitem>

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<esko:productionitem xmlns:esko="http://www.esko.com/myowns/">
  <esko:subelement content="B" />
  <esko:subelement>2</esko:subelement>
</esko:productionitem>
```

Encoding

If the input XML file contains a header that specifies an encoding (for example encoding="ISO-8859-3"), the task will write the output files in that encoding. If no header or encoding is specified, the resulting files will have UTF-8 encoding by default.

XML Validation

The task will generate well formed XML, but there is no support for schema or DTD validation. References to DTD's for validation will be removed.

5.2.3 Creating XPath Expressions

Usually you will not define the XPath expression by typing it in the task window yourself. Use the Edit button to open the XPath Builder. This tool will help you create your own correct XPath expression.
by clicking the variable elements. By selecting parts in the window with **Example XML Content**, you will see the resulting XPath expression automatically appear in the top part of this tool.

**Important:** When the **Example XML content** window is empty, this means that you forgot to select an XML input file before opening the **Split XML task**.

![Example XPath Expression](image)

Selecting the option **Select siblings** allows you to automatically select the right level.

In above example, the user selected the level 'productionitem'. The selection is shown in yellow. The siblings are also automatically selected and are shown in grey. The correct **XPath expression** that is needed to split that level of elements will automatically appear in the top part of this tool.

You can easily build complex expressions with this tool, such as:

- selecting elements conditionally.
- selecting an element from a specific 'namespace'.

Find more information on the **XPath Builder** in *The XPath Builder* (chapter SmartNames).
5.3 Join XML Files Task

5.3.1 Concept

The **Join XML** task joins the content of its input XML files. In case you do not want the full content of all these input files, there are options to determine what parts you want or from what root level down that you want their content.

Some typical examples:

- The information you need is spread over several source XML files, maybe even coming from different systems. You need (parts of) them combined in 1 XML file.
- You want to change the content of an XML. After splitting it you can re-join specific parts.

**Note:** The **Map Data** task can also be used to **join** XML files, but it offers much more features and options than this **Join XML** task.

5.3.2 Example and Options

The **Join XML task** joins multiple input XML files into 1 output XML file. The default option of this task is to take the full content of the input files and join everything together into 1 XML file.

![Edit Ticket of Join XML Files](image)

**Name of the output XML file**
The default name of the output XML file is set to [File]-joined. [File] is here the name of the first input file. You can of course change this setting.

Xpath expression that extracts content from the input XML files

Tip: In the previous topic about the Split XML task, we introduced XPath expressions and also introduced the XPath Builder tool.

This canvas is where you can define the content of the output XML. If you do not want the full content of all your input files, use XPath expressions to define what you want to extract from them. Per input file, all nodes that comply with the given XPath expression are imported into the output XML file.

Click on +Ω to insert SmartNames to help build your XPath expression.

Click on Edit to open the XPath Builder. For more info on XPath Builder please see The XPath Builder (chapter ‘SmartNames’).

If this canvas is left empty, and if you didn't specify a Root Element, then all the nodes under the root node are considered (XPath == /*/*).

Root Element in your output file

Each input XML file has its own root element. You can define here what you want as root element in the output XML file. See an example where this root element was set to 'ink':

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ink>
  <name>BananaYellow</name>
  <type>Designer</type>
  <book>FruitColors</book>
</ink>
```

If this field is left empty, then the tag name of the root of the first input file will be used.

Example

You have these 2 XML files:

- One XML file lists ink attributes of a (first) ink of a PDF:

  ```xml
  <?xml version="1.0" encoding="UTF-8"?>
  <job>
    <name>banana.pdf</name>
    <ink>
      <name>803</name>
      <type>pantone</type>
      <book>pms1000c</book>
      <angle>7</angle>
      <ruling>92</ruling>
      <dotshape>round</dotshape>
    </ink>
  </job>
  ```

- Another XML file contains information on the ink coverage of that file:
You now want to join these 2 files into 1 XML file with all the ink attributes together.

- This is the result if you use **no options:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
- <job>
  - <name>banana.pdf</name>
  - <ink>
    - <name>803</name>
    - <type>pantone</type>
    - <book>pms1000c</book>
    - <angle>7</angle>
    - <ruling>92</ruling>
    - <dotshape>round</dotshape>
  </ink>
  <name>banana.pdf</name>
  <customer>FruitCo</customer>
  - <inkcov>
    - <pct>27.12</pct>
    - <mm2>6512</mm2>
  </inkcov>
</job>
```

- This is the result when you set the option **Root Element** to **PrintItem:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
- <PrintItem>
  - <name>banana.pdf</name>
  - <ink>
    - <name>803</name>
    - <type>pantone</type>
    - <book>pms1000c</book>
    - <angle>7</angle>
    - <ruling>92</ruling>
    - <dotshape>round</dotshape>
  </ink>
  <name>banana.pdf</name>
  <customer>FruitCo</customer>
  - <inkcov>
    - <pct>27.12</pct>
    - <mm2>6512</mm2>
  </inkcov>
</PrintItem>
```

- This is the result when you also used the **Edit** button to add the **XPath Expression /Job/ink**
Because the Xpath extracted only the content of <ink>, the pdf file name and the customer name are not kept in this joined file.

**Note:** To further change the structure, order or content, you can use the Map Data task.

### 5.4 Map Data task

#### 5.4.1 Concept

The **Map Data** task is a powerful tool that can be used to

- convert CSV files into XML and vice versa.
- transform the content of a CSV or XML file: extract, join, rename or re-format parts.

**Note:** A CSV or comma separated file is a structured text file. In such files commas or some other character are used as field separators. CSV can be generated by exporting an XLS file in Microsoft Excel.

Some example use cases:

- After interacting with an external system, you need to simplify or re-write the XML files that you received.
- You need to combine information from different sources into 1 file, but also reformat it (this is more than what the Join XML task can do). For example getting job specification from an incoming XML file and getting ink information from the metadata of an uploaded PDF (XMP). You want to combine those into one XML that can be used for the Job Creation task. To do this, the XMP ink information should also be transformed to the XML format that is needed for Job Creation task.
- You need to transform an XML that you received before it can be used by the Create CAD Sheet task.
- You receive an XML file with all possible sheet layouts. Now you need to create a list of these layouts using a custom text format (for example plain text with line breaks, CSV or XML).

**Tip:** The chapter Examples on page 72 contains many examples that use this task.
5.4.2 Main Tools in the Task Panel

The Map Data task panel consists of three main parts:

1. Option to choose the Output Format.
2. Canvas to specify the content of the output file of this task.
3. List of Parameters that you want to read from the task’s input files.

Choose the Output Format

- **XML**: Click on Options to define these output format options:
  
  - **Replace reserved characters in parameter values with XML entities**: It is possible that the parameters from the input files return characters that are not allowed in XML. With this option, you choose to map these special characters to ‘XML entities’. Below list shows these mappings:

    | Reserved characters in result value | XML Entities |
    |-------------------------------------|--------------|
    | "                                  | &quot;       |
    | '                                  | &apos;       |
    | &                                  | &amp;         |
    | >                                  | &gt;          |
    | <                                  | &lt;          |

  - **Task fails when output XML is not a valid XML file**: Enable this option to have Automation Engine check the validity of the XML. If this option is on, the task will fail if the XML is not valid. The validation checks the XML formatting and the XML tag endings.
Note: This task does not check XML schemas such as XSD, DTD, etc.

- CSV: Click on Options to define this Output Format option:
  - Text Encoding: Choose what type of text encoding that the system that will read this CSV file prefers. The default is UTF-8.

Canvas to Specify the Content of the Output file

The canvas on the left there is where you can
- insert XML or CSV data from another file. This is the typical way to insert your example output file, after which you can start replacing its absolute values with SmartNames or Parameters.
- insert Parameters from your input files (from the list on the right side of the panel)
- insert SmartNames
- insert Xpath expressions (the standard language to extract specific parts of an XML)
- ...

See more in Specifying the Output File on page 54.

Tip: Right-click in the canvas to get a list of content to insert.

A List of Parameters from your Input Files

In this field, you build up a list of parameters from your input files. You can then insert these parameters easily in the canvas on the left, similar to how you insert SmartNames. These parameters can come from one or more input files.

At the moment you set up the Map Data task, it is good practice to use example input files that contain the same parameters as the files that will be the real input files of the Map Data task once it is started in a workflow. It is also a good idea to start from a sample output file (see further).

See more in Setting Up (Example) Input Files on page 49 and Adding Parameters on page 51.

5.4.3 Main Workflow in Using the Map Data Task

Before we go in more detail explaining all the possibilities of this powerful task, it is good practice to stick to following main workflow:

- Step 1: Get an example of the output file you need to create. Analyse its structure.
- Step 2: Get (an) example input file(s). Analyse its structure.
- Step 3: Create the parameters from the input file(s).
- Step 4: Specify the output file. Use your example output file and the parameters that you made from the input file(s).
• Step 5: Test the resulting ticket in a workflow, using real input files.

5.4.4 Setting Up (Example) Input Files

**Attention:** An example input file is a file that you use while you set up the Map Data ticket. Be aware that this example file is often not your real input file once the task is running!

**Important:** The Map Data task allows to create parameters *without* the help of example input files but this will be much more complex. We advise to create the parameters based on example input files because it will help you with correct spelling, see the larger context and give you immediate feedback.

2 ways to select an Input File

- In the Pilot, select one or more input files before you open the Map Data ticket. These files will show up as input files.
- Open the Map Data ticket and click on Set Up Input Files....

Setting up an Input File is explained in more detail in the next sections of this document.

Working with Multiple Task Input files

By default a Map Data task will start as soon as it gets one input file. The task will not wait for a list of input files to be complete. But you can easily solve that.

To make the Map Data task work on multiple input files, you need to make a workflow where you have a Data Collector task before your Map Data task.

The Data Collector task will wait for outputs of previous tasks and hand them over as one group to the next task.

How to Control the Order of Multiple Input Files

If your Map Data task has more than one input file, it is very important to control the order of those input files. What you want to avoid is

- parameters that do not find their value because they look in the wrong input file.
- parameters that find a value but one that comes from the wrong input file.

Some examples that illustrate the importance of parameters looking in the correct input file:

- Your Map Data ticket is part of a workflow that creates a new Job. Does the parameter [date] or [time] come from the input file from the MIS or from the metadata-XML from the PDF that the customer sent you? The difference can result in a conflict about your due date of that Job!
- Your parameter [print technique] selects the field in row 2, column 5 of a CSV file. If this parameter searches this field in the wrong input file, then your search can return "Rio de Janeiro" instead of "Digital Flexo"!
To control the order of your input files, we advise to first use the **Data Collector** task, followed by a **Sort** task and only then followed by the **Map Data** task.

The **Data Collector** task will wait for outputs of previous tasks and hand them over as one group to the next task.

The **Sort** task enables you to sort that group. You can base the order on file name, time or SmartNames.

### Setting up (Example) Input Files - Details

1. Click on the **Set up Input Files**...
   
   A panel opens that shows your current list of Input Files, their name and order in the list, their type (XML or CSV) and if they link to an example file or not.

2. To add or modify one, click on **Add** or double click an existing Input file definition.
   
   The **Set Up** panel opens.

3. Choose the **File Type** of the desired Input File: **XML File** or **CSV File**.

4. In case you chose **CSV File**, you can change the following CSV options:
   
   - **Text Encoding**: Indicate the **Text Encoding** of the CSV file. The default encoding is Unicode UTF-8.
   
   - **Field Separator**: Indicate which character is used to separate the fields in the CSV. Check your CSV in a simple text editor to detect it.
• **Quote Character**: Indicate if the value in the CSV contains quotes that you do not want to keep. For example: In the CSV the value is written as "25000" but you want to extract only the 25000.

• **Ignore line breaks**: Check this option if the CSV only contains one record (column).

• **Ignore first line**: Check this option if the first line of the CSV contains the names of the columns.

**Note**: These CSV options have a great impact on the way the CSV input files are handled. A selected value field (using the field number and the line number of the field) can be the wrong one if the CSV options are not set correctly.

5. If you want, select an example file to be able to easily select data to be used as the value for parameters.

**Removing an Input file**

In the **Set Up Input files** panel, the **Remove** button will remove the selected input file.

When there are no parameters created yet that use that input file, the input file item will be removed. You will not be asked to confirm.

But when the input file is in use, you will see a warning mentioning this use. You will be asked to confirm before deleting the input file item together with all the parameters or ‘repeating content’ that require that input file.

**Note**: A similar warning and effect happens when you change the **File Type** of the Input file (from XML to CSV or vice versa).

### 5.4.5 Adding Parameters

There are basically 3 ways to create new parameters:

• creating them one by one, using the **Add** button.

• loading **multiple** parameters from a file, using the button.

• duplicating an existing parameter, using the button.

**Adding Parameters**

Once you specified the output format and set up the (example) input files, you can start adding parameters that represent the values that you wish to insert from the task’s input files into the output file.

To add parameters, proceed as follows:

1. **Click Add** underneath the **Parameters** list.
   The **Add Parameter** dialog prompts.
2. **Enter a Name** for the new parameter.
3. Select the Input File that contains the value for the parameter.

4. Specify where the value for the parameter is located in the selected Input File:
   - In case you selected an XML Input File, specify the XPath Expression of the value in the input file to be used as the value for the parameter. If you selected an example XML file for the XML input file, you can easily create the XPath Expression using the Edit button. This will open the XPath Builder. For more information on XPath Builder, see The XPath Builder (chapter 'SmartNames').
   - In case you selected a CSV Input File, specify the Field number and the Line number of the field in the input CSV file to be used as the value for the parameter. If you selected an example CSV file for the CSV input file, you can select the desired field using the Select Field button.

5. Separator. If the XPath expression returns multiple elements, decide what character you want to be used as separator.

6. Formatting Options. If you suspect that the chosen value might have leading and/or trailing spaces, select Remove leading and trailing spaces.

7. If you want to save the created parameter as a SmartName, then click Save as SmartName.... In the prompted dialog, enter a comprehensive Name and Description and choose whether or not you want this SmartName to be Private.
8. Click **OK** to add the parameter to the **Parameters** list.

The created Parameter is now in the **Parameters** list. You can now insert the created parameter in the output XML or CSV (see *Specifying the Output File* on page 54).

**Note:**

- To edit an existing parameter, select it in the **Parameter** list and double-click it.
- To delete an existing parameter, select the parameter and click **Remove**. You will get a warning when you try to delete a parameter that is in use (in the canvas on the left).
- To delete all unused parameters, click on the action button underneath the **Parameters** list and choose **Delete Unused Parameters**. You will be asked to confirm.
- To add Parameters from an example input file, see the next section.

---

**Loading Multiple Parameters from a File**

Instead of creating parameters one by one, you can choose to load multiple parameters from a file. Those that you would not need are easy to remove from the list anyway. To do so, proceed as follows:

1. Underneath the **Parameters** list, click on the **Action** button.
2. Choose **Load Parameters from File** in the prompted menu.
   
   The **Load Parameters from File** dialog will prompt.
3. Select the desired **Input File** from which you want to load the parameters.
4. If the selected **Input File** is an XML file, proceed as follows:
   
   a) Specify the **XPath Expression** to load the XML parameters from the **Input File**. You can click **Edit** to open the Xpath Builder to help you build the desired **XPath Expression**.
   
   b) Click **OK**.

   All parameters from the selected XML Element will be loaded, including child elements and all attributes

5. If the selected **Input File** is a CSV file, proceed as follows:

   a) If the **First line contains parameter names**, select the similarly named option and the values from the first line will also become parameters.
   
   b) Click **OK**

   CSV Parameters will now be added for every field of every record in the **Input File**.
Note: About names of parameters:

- When one or more parameters that are loaded from the example input file have the same name (because the XML element name, the XML attribute name or the CSV field name in the example XML/CSV file is the same), their names will be made unique by auto-numbering them.
- When one or more parameters with the same name already exist, a warning dialog will pop up asking you if you want to replace them or not. If you choose to replace them, this will overwrite their current contents.

5.4.6 Specifying the Output File

The canvas on the left of a Map Data ticket is where you define what your resulting XML or CSV output file will contain.

There are many types of smart content that you can add to this canvas. And you can even type in some content yourself. Some of these types of content are listed via the button. You will also see a complete list of possibilities when you right-click inside the canvas.

Note: Most of these are sensitive to the position of your cursor at the moment you right-clicked. They will not be offered when they are not valid at that specific insertion point.

Caution: There is no ‘Undo’ function while editing in this canvas.

Tip: Find a good example in the chapter Examples on page 72. It shows how 2 XMLs from the application Google Maps are mapped into one smaller output XML. That output XML so contains only specific content to be picked up as workflow parameters.

Here are all the methods and data types that you can enter:

Cut-Copy-Paste
These standard functions work on all types of data you have entered here.

Clear Contents
This will clear the whole canvas.

Insert SmartName
Your output XML can also contain SmartNames. This way you can add information to the XML that was maybe not present in the input files but important extra data for the system that will read the output file.

Note: You can also use the dedicated button.

Paste from File...
The complete content of the selected file will be added here. This will be possible with all content that is in a format that can be output to XML or CSV.

Important: Use this function to paste your example output file.
**Insert XML Declaration**
It is a good practice to have an 'XML declaration' at the start of your XML file. This line indicates the file encoding to the reading application. This feature will here insert the industry default one:

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

**Insert XML Element**
Type in the name of the element that you want to insert. An opening and closing statement of that element will be inserted. For example:

```xml
<Distance></Distance>
```

Often, you will continue by inserting a Parameter in between those two.

**Insert XML Attribute**
Type in the name and value of the attribute that you want to insert. For example:

```xml
units="miles"
```

For example: see how this attribute was inserted in the above element:

```xml
<Distance units="miles"></Distance>
```

**Insert Parameter**
Choose the parameter from the list. Alternatively, select the parameter in the parameter list on the right side and

- Click the to insert that parameter at the current cursor position in the canvas.
- Drag & drop the to the right position in the canvas.

For example: the parameter [GoogleMapsDist] was inserted in above example:

```xml
<Distance units="miles">[GoogleMapsDist]</Distance>
```

**Insert New Parameter**
This is a shortcut that combines the adding of a new parameter with its immediate placement in the canvas. After you inserted this new one, it will also appear in the parameter list on the right.

**Insert Repeating Content (XML statement 'For Each')**
For example: for each job that is described in the input file, you want to pick up all its ink specifications (name, type, angle, ruling). This content will be repeated in the output file for each found job.

After making sure your insertion pointer is at the correct place in the canvas, right-click or use the dedicated button. A dialog appears and in the canvas a visual "repeating block" appears in pink.

Now start typing or add XML elements in the repeating block.

Mind that you insert the parameter at a valid insertion point.

**Note:** When you add a parameter while your insertion point is inside a pink block of repeating content, the dialog will mention that that parameter will only serve to be inserted in repeating content.

Choose to insert a new parameter and click OK. The parameter is inserted and also listed in pink in the parameter table.
Note: When the insertion pointer is out of the repeating block, parameters that were used inside the repeating block cannot be inserted outside the repeating block. Their insert buttons are then disabled (shown in grey).

It is also possible to insert Repeating Content inside another Repeating Content block. This is called **Nested Repeating Content**. See an example:
6. Tasks Interacting with other Systems

6.1 Interact with Database task

6.1.1 Concept

Tip: The tasks that interact directly with external systems were introduced in Automation Engine tasks Interacting directly with the External System on page 12. You there also read how this task compares to the Database Access Point.

The Interact with Database task allows you to communicate directly with an external database. It allows both getting information from that database (read) and sending information to that external database (write).

As mentioned in the chapter Integration Concepts on page 11, a typical use for this task is to feed information back to external systems by using this task in a workflow. For example, you can set up this task to update a record in a table of your MIS when your workflow has reached a certain stage. For example setting the status record to 'Proof Ready'.

Warning: This task may not be used to interact with Automation Engine databases. This is not supported by Esko. This can damage your server configuration.

Warning: See the important remark in the task panel. As mentioned earlier in this document, using SQL statements to change records of an external database can potentially harm that database. If you are using this technique then this means that you have a database specialist who takes all liabilities for these SQL queries.

Tip: The chapter Examples on page 72 contains 2 examples that contact an external database.

6.1.2 Set Up and Options

File Name
When the SQL query is a 'select' statement (see further), the task will dump the found records in an XML file. The default ticket offers the SmartName [Table Name] as name for that output XML file. In below screenshot, this name would then be 'ProductionDetails'.

Note: The SmartName [Table Name] is only offered in a context where it makes sense.

Caution: Not all SQL queries will resolve a name of a database table. Do not use the SmartName [Table Name] when it is possible that no table name is found.
External Database
Select a database from the list of configured databases. Click Configure... to access the configure tool directly. Read about configuring **External Databases** in the chapter *External Databases* in the Reference Guide.

**Important:** Before linking Automation Engine with an external database, please check your license agreements with the database manufacturer.

**Statement**
Choose the type of SQL statement that you want to use: **SELECT** or **OTHER**.

- **SELECT**
  
  When using the Select statement, the output of the task will be an XML file.
  
  See an example of such a resulting XML. Notice that the SQL statement is also mentioned inside.

```xml
<xml version="1.0" encoding="UTF-8" >
  <one created="2019-05-23T10:41:32.100+02:00" producer="sqlexec" select="select * from ProductionDetails where Factory="FRANCE" stateid="1">
    <name>Factory</name="FRANCE"></name>
    <name>Machine</name="5082"></name>
    <name>ItemCode</name="18-0474"></name>
    <name>Run</name="March11"></name>
    <name>QC</name="Passed"></name>
    <name>Comments</name/>
  </one>
</xml>
```

**Advanced Options...:** You can avoid creating an XML file that is too big by limiting the number of records with the option **Maximum number of records per file**. The task will then generate as many files as needed.
• Other

In this mode, you can execute any other SQL statement such as INSERT, UPDATE etc. However, the task will in this case not generate an output file.

**Note:** When the **Statement** type is set to **Other**, you cannot execute a **Select** statement.

In this example the status of the Job is changed in the orders database:

![SQL Statement Example](image)

**Execute SQL Statement**

Decide if you want the SQL statement to be executed **Once** or **Once per input file**.

For example: You have multiple input files for this task: banana.pdf, lemon.pdf and mango.pdf. When your SQL statement uses the SmartName [File] (= the task’s input file), then the option **Once per input file** will make the task run **for each input file**. With the option **Once**, the task would only run for the banana.pdf (the first input file).

**Tip:** Starting a task once and having it run on multiple input files is much lighter for the software than starting the task multiple times with each time only one input file.

### 6.2 Interact with Web Service task

#### 6.2.1 Concept

**Tip:** The tasks that interact directly with external systems were introduced in the chapter *Integration Concepts* on page 11.

The **Interact with Web Service** task uses HTTP requests to communicate with a web service of an external system. Both **read** (‘GET’) and **write** (‘POST’) commands are possible. The result of this task is typically an XML file.

When an external system is running a web service, you will be able to access it via HTTP (HyperText Transfer Protocol). However, to do this, you also need to learn what you can communicate to that system, what language you need to speak to it. Some applications will indeed publish their API (Application Programming Interface). And it is that API documentation that will tell you what you can enter in the **Interact with Web Service** ticket.

**Tip:** The chapter *Examples* on page 72 contains an example where a web service is contacted.
6.2.2 Set Up and Options

Output folder and File Name
Define here the path and name of your output file. The type of output file is defined by the API. We advise to choose XML.

When the Method is GET, this file will contain what the external system returns you. When the Method is any other type, the output file will only contain a status of the interaction (for example 'OK').
Service
Choose the web service from the list. Click Configure... to access the Configure tool where you define such external Web Services. For more information see the chapter Configure in the Reference Guide.

Method
There are various types of HTTP methods that you can send to a web service. Choose the method from the list:

• **GET.** This most frequently used method will retrieve data from that external system. Consider the URL as the question and the output file as your answer.

For example: ask Google Maps the distance from your company to your customer.

• **POST.** This method creates a new object on the external system.

  • This method is commonly used to post actual data, to upload a file to the external system. Therefore, click on Options... and choose to Attach input files. All input files are added to a single HTTP request and SmartNames are resolved in the context of the first input file. In case each input file must be sent separately, then add a Data Splitter task before the Interact with Web Service task in the workflow. In that case SmartNames will be resolved for each file.

  **Note:** The parameters are sent as "application/x-www-form-urlencoded" data. When XML files are attached, the data is sent as "multipart/form-data".

  • The POST method also enables to send a SOAP request. The exact text of the SOAP request needs to be put in an XML file and you need to add a single parameter "SOAPAction" with the needed value.

  **Note:** the Create XML task can be used to embed SmartNames inside the SOAP request.

  **Note:** Find an example on Esko Knowledge Base.

  • **PUT.** This method requests the external system to store data or information under the given (existing) URL.

  • **DELETE.** This method requests the external system to delete the information corresponding to the given URL.

  • **Select SmartName...** You can even use a SmartName to define the method.

  **Note:** PUT and DELETE are only used for 'RESTful' web services. 'REST' is the main architecture style of HTTP. An alternative is 'SOAP'.

Path
This path will be documented in the API of the external system. APIs often present several paths that you can address.

In the above example of a ticket to access Google Maps, the information will be gotten from the 'directions' part of the API. The path ends with XML which here means that we choose to get an XML file back (and not a JSON file for example).

Headers
The API documentation will instruct you if you need to use HTTP headers. To do this, add their key and value in the Headers list.
Parameters
The API documentation will instruct you what parameters you can use. To do this, add their key and value in the Parameters list. This is where the actual 'question' will become more clear.

In the above example, you can see how the ticket asks Google Maps a distance between the origin and destination address. Other parameters could be more optional, like in this case the option to only calculate distances for routes that avoid toll roads.

Tip: Just Google "Google maps API" or click here to learn how Google Maps is a nice example of how an external system publishes its API.

Tip: The chapter Examples on page 72 contains an example that uses the Google Maps API.
7. Tasks often used in Integration Workflows

7.1 Create Job task

7.1.1 Concept, Workflow and FAQs

The Create Job task creates an Automation Engine Job based on an XML input file containing the job parameters. Typically this input XML file comes from your business system. It is inside this system that the job-orders are decided and where many of its prepress production parameters are entered first. Typically, a Folder Access Point receives this XML file and then starts a workflow containing this Create Job task.

Tip: Read more about the concept of integrating with external systems in Introduction on page 8.

Tip: The chapter Examples on page 72 contains many examples where this task is used.

It is possible that the XML file that feeds this Create Job task does not come directly from the business system. Maybe you first have to combine it with another input file that contains other job parameters. Such other parameters could for example arrive as metadata inside a PDF. The data formatting tasks Split XML, Join XML and Map Data offer all tools to combine these parameters into 1 XML file.

Should the XML be in a specific format?
No, but it should be a valid XML of course. You will set up SmartNames or XPath expressions that map the vocabulary of that business system to the terminology required in Automation Engine. For example your business system uses the term ‘Order Number’ to describe the name of a job. You can then make a SmartName that maps ‘Order Number’ to ‘Job Name’.

Can one XML file create multiple Jobs?
If your XML contains the description of multiple jobs, then you first need to split that XML into multiple XML files that each describe one job. To do that you will set up a workflow with the Split XML task in front of this Create Job task. Also use the Data Splitter task to make sure that the Create Job task is started for each incoming XML file.

Can the input file be a CSV file?
If you want to create a Job from a CSV file, then use the Map Data task to convert it to an XML file.
What if my business system cannot export an XML file?
When your business system is not capable to export a job description in XML format, you can consider the alternative where Automation Engine asks your business system if there is a new job to be created. Read about the various possible techniques in the chapter **Access Points** on page 17.

Can the task find its parameters outside the task's input file?
Yes. The input file of this task does not have to be the XML file that delivers all the parameters. It can even be a PDF. The parameters could also be looked for in other files than the task's input file. Some examples of such special setups:

- A **Create Job** ticket with a PDF as input file and where the SmartNames do SQL queries to an external database.
- A **Create Job** ticket with SmartNames that find their values in **workflow parameters**.
- A **Create Job** ticket with (SmartNames that are) XPath expressions to some centrally stored XML that the business system writes there every night at 3 AM.

The **Create Job** task can get the values for its parameters from different sources. It can for example

- find the main job order specifications from the business system's XML file (name, ID, customer, due date)
- AND get the some ink information from the metadata of a PDF
- AND get a custom parameter 'Press' from a workflow parameter `[wfp.PressName]`.

What if the Job already exists? Can I use this task to **update** Jobs?
Yes you can also use this task to update existing Jobs. If Automation Engine detects that the Job already exists, it will update the Job with any new or changed values.

This task will function as an **Update Job** task

- when it sees that the [Job Name] already exists
- OR when it sees that the **combination of** [Job Order ID] and [Job Sub Order ID] already exists.

At the end of this chapter, you can find a sample XML that can be used to create jobs from. Remember that the format of the XML is free. This sample is not a template, it is just one example.

We will now document the settings in the different tab sheets.
Important: To set up a Create Job ticket, you need advanced knowledge of SmartNames.

7.1.2 Job

For most of the fields in this task, you can choose to set a SmartName or a combination of SmartNames.

You can also directly write XPath expressions that will extract values from an XML file. The syntax for doing this is "XPath:" followed by a valid XPath expression for the input file.

Caution: If you are not an expert in building XPath expressions, we advise to not type them here but to create them via XPath SmartNames.

- **Order ID**. This field must have a value. You cannot leave it empty.
- **Sub Order ID**
- **Name**. In the screen shot example, we combined 2 SmartNames with some absolute text ( _ ).
- **Description**. In the screen shot example, we wrote an XPath expression.
- **Due Date**. The format needs to be compliant with ISO 8601 (YYYY-MM-DDThh:mm:ssTZD). For example: 2014-07-28T12:00:00+01:00
- **Project ID**.
Tip: Some users use this field to define the name of the project on Esko WebCenter that corresponds with this Job.

- **Category.** For this field, you can also choose a category from the list.
  
  Tip: The categories in the list are all those that were already used in a Job.

- **Custom Fields.** The custom fields that appear here are the ones that you defined in Configure > Jobs > Job Setup (maximum 6). Alternatively to using a SmartName or XPath, choose a value from the list.

- **Job Folder.** The system needs to know where to create the actual folder for this Job.

  Note: Beware that you do need to define a complete network path here, not just a folder name.

You have these choices:

- Use the SmartName [Smart Job Location] to use this setting as defined in Configure > Jobs > Smart Job Location.
- Click **Browse** to browse to the desired job location
- Click **Browse** to browse to a folder in a container, and then use SmartNames to define a sub folder.

  Note: If you want to make the path smarter by adding the name of the customer, then you best set that up as a Smart Job Location.

### 7.1.3 Customer

Using the same techniques as described for the tab 'Job', you here define the customer details for the Job you are creating.

- **Customer ID.** You can alternatively select the ID from the customer list panel.
- **Customer Name.**
- **Customer Description.**
- **Customer’s Job Reference.** This is the name that your customer uses for this Job.
- **Customer Service Representative.** You can alternatively select the CSR from the list.
- **E-mail.**

### 7.1.4 Inks

Setting up the inks for the new Job can only be done if the input file is an XML file and if the ink properties are stored in attributes of XML elements.

In the example below the XML input file contains an element "Farben" (German for "inks"). This is an array of "Farbe" elements, each describing an ink.
The **Create Job** task will create an ink in the new Job for each of these elements in the XML. The table in the dialog defines how to map XML attributes of each element to an ink property.

- Load Ink Information from an XML element. Enable this and then choose how you will load this specification: You have these 2 options:
  - **XML element Name**: Indicate the XML element that contains the ink specification. In above example the ink info is described in the element `Farben`.
  - **XPath expression**: Alternatively, use an XPath Expression to the XML element containing the ink information. For example: `/CreateJob/Farben`
Note:
- To open the XPath builder, select XPath expression and click on the Edit button in the text field.
- The XML that the ticket sees as input file will be loaded in the XPath builder. Alternatively, you can right-click in the canvas and choose Paste from File... to select the example XML input file.
- When you edit the XPath expression again, that same XML (example) input file will be automatically used.

For each job ink parameter, specify the corresponding XML element that is used in the XML file. To do this, select the parameter and click Edit.

- Ink name
- Ink type
- Ink book
- Angle
- Ruling
- Dot Shape
- Printing Method

### 7.1.5 Parameters

To add custom parameters to the job you are creating, click Add and enter its Name and Value. Both fields are SmartName enabled. Typically, the Name will be an absolute value and the Value is a SmartName that extracts the custom parameters from an XML file.

For example:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapping</td>
<td>[FromMIS.XML_TrappingYesNo]</td>
</tr>
<tr>
<td>Trap Distance</td>
<td>[FromMIS.XML_TrappingDistance]</td>
</tr>
</tbody>
</table>

### 7.1.6 Bar Codes

The Create Job task also allows to add parameters for one or more bar codes.

Important: The Create Job task does not check the consistency of these bar code parameters. This means that the creator of the input file is responsible to create bar code parameters that are compliant.
You can Add or Remove parameters for multiple bar codes.

For each bar code, you can set the following parameters by adding a SmartName or an XPath expression. Some also offer to select a value from a list.

- **Type**.
- **Sub Type**. A list will be offered if you selected a Type that offers Sub Types.
- **Code**. As mentioned above, the code is not checked for validation here.
- **Composite Code**. Only some types offer to add an extra composite code.
- **Output Resolution**
- **Bar Width Reduction**
- **Device Compensation**
- **Narrow Bar**. Only some types work with a specification of Narrow Bar and Ratio
- **Magnification**
- **Ratio**. Only some types work with a specification of Narrow Bar and Ratio

### 7.1.7 Template

You can use an existing job as a template job for the new job that you here create. If you decide so, you can select which attributes and parameters that you want to get from that template job.

**Note:** For more information about using template jobs and their options, see the dedicated chapter Jobs.
Select **Use an existing job as a template** if you want to create your job based on another job.

In **Choose Job, Select** the job from the list or use a **SmartName** to have the job selected. An example of a SmartName could be `[customer name]_template`

Then choose what to **Include** from the template job into the new job:

- **Files**
- **Folders**
- **Hot Folders.** This item will only appear if your server still has hot folders from before v14.
- **Tickets.** These are the (blue) Job tickets.
- **Milestones.** Times will be set relative to the creation date of the new job.
- **Parameters.** Custom parameters.
- **Inks.**
- **Categories**

**Attention:** If there any parameters that you pick up both from the input file and from a template job, then those from the input file will be kept.

### 7.1.8 Advanced

The **Create Job** task can automatically launch a workflow when the new job is created and even when the task updates the job.

To do this, select when you want have a workflow launched and select the workflow from the list. You can even choose both moments.

**Note:** If the **Create Job** task itself is part of a workflow, the workflow that is chosen in this task will start immediately, as a separate workflow. It will not wait for the workflow that contains the **Create Job** task to finish.

The advantage of this function is that the chosen workflow will be launched in the context of that created/updated job. This means that all the used SmartNames will get their value from that job.

**Attention:** In case you create a workflow that contains the **Create Job** task as a workflow step and then continues with other steps, then the tasks following that **Create Job** step will not run in the context of the new/updated job: they will run in the context where the (main) workflow initially started!

### 7.1.9 Sample Input XML

Click **this link** to find a ZIP with the samples used in the chapter **Examples** on page 72. There is a sample XML in the folder “Create a Job using XML” that you can use to test setting up SmartNames, Xpath expressions and so build and test an example **Create Job** task.

Download the ZIP file and unzip it to a local folder or directly into a **Automation Engine Container**.
7.2 Add To Products task

The Add To Products task allows to create Automation Engine Products. The task can get their properties out of file names, paths, SmartNames or values out of XML data.

This task is often used after workflow steps first made a smart selection of input files.

This task is documented in the chapter Products. Find a direct link here.

7.3 Link Product To Job task

The Link Product To Job task allows to link or multiple Products to a Job.

This task is documented in the chapter Products. Find a direct link here.
8. Examples

These examples will help you build and understand the workflows of typical integration use cases. At the beginning of each example, we also mention which tools and tasks are used. We also add a link where you can find the sample (XML) files used in that example.

Note: An example often requires you to understand the features explained in the previous examples.

8.1 Creating a Job using XML

Summary

We will create a Job based on the sample XML. Most of the work is creating the XPath Query SmartNames.

Tools used in this example:

- A sample XML describing 1 job order
- SmartNames including Xpath Builder
- The Create Job task

8.1.1 Step 1 - Analyze your Input XML file

Click this link to find a ZIP with the sample XML file used in this example.

Save the file in a logical place in an Automation Engine Container.

Open it (double-click) and have a look at its structure:
See how a lot of job info is inside the parent element 'CreateJob' and how the ink information is inside a different parent element 'InkSpecifications'.

**Note:** In this sample, even the full path for the JobFolder is filled in. This is usually not the case. A business system usually leaves it up to you to decide where to store your prepress data. Some MIS do want to control this as well. Advantage of already knowing exactly where your prepress files are is that they can then also show some of them to their MIS users (final PDF, JPEG,..). However, most MIS will not decide that path for you but will ask you to let them know where you decided to create that JobFolder. In that workflow, the MIS usually wants you to return them a XML containing that JobFolder path.

### 8.1.2 Step 2 - Create your SmartNames

SmartNames will read specific pieces of that XML and so populate the various properties of the Job you will create. To read parts of that XML, we will use SmartNames of the type *XPath Query*.

**Note:** You can also create SmartNames directly from within the Create Job task. However, because they are Xpath queries that involve a sample input file, we advise to create them using the main SmartNames tool. It will be somewhat easier to do it there.
1. Go to the SmartNames tool and indicate your sample input XML file
Because you will probably make a dozen new XPath SmartNames, it will be easier when you only have to indicate your sample input file once. Do this in the **Resolve all using:** section.

![SmartNames tool screenshot](image)

**Note:** You do not have to click on 'Show' at this moment.

2. Make a first XPath Query SmartName

   1. In the SmartNames tool, in the top bar, click on [ ] to create a new SmartName.. Choose Type **XPath Query**.

   ![XPath Builder screenshot](image)

   An example query about 'vsize' appears. It will be removed automatically later. See the part / `text()` at the end. This means that the query will not select the full element but only the value of that element.  

   At the bottom,  

   2. Click on 'Edit' to open the **XPath Builder**.
As a first SmartName, we will create the Xpath Query representing the Job's OrderID.

All you need to do here is click on the value that the SmartName will be looking for. It will be highlighted in yellow and the resulting XPath Expression will automatically appear in the top field.

3. Click on 'OK' to close the 'Edit Xpath Expression' dialog. The expression will have replaced the example one. The resulting value will also be shown bottom left in 'Result:' (in below screenshot: "002").

4. Now save your new SmartName. Give it a logical name and maybe add some tags and a description.
3. Now make the other SmartNames (quickly)

The next SmartNames will be made very similarly. Double click the one you just made and while keeping the SmartName dialog open, follow these steps:

- click **Edit**
- in the **Xpath Builder**, 
  - click on the value you want the SmartName to represent
  - click **OK** to close the Xpath Builder
- click **Save As...** and enter an appropriate name, tag and descriptions

Repeat this for all the SmartNames you wish to use in the **Create Job** task.

8.1.3 Step 3 - Insert the SmartNames in the Create Job ticket

Open a **Create Job** ticket and insert the created SmartNames in their corresponding fields.
Do the same for all fields in the other tabs.
Don’t forget to save your ticket.

**Picking up the Ink parameters from the XML**

As explained in *Inks* on page 66, the *Inks* tab is constructed differently.
In our sample XML, see how the ink information is described in a parent element *InkSpecifications*. And in there, *per ink* in an element named ‘*Ink*’. And there are 8 inks described.
You also see that the names of the ink parameters are not all identical to those used in the Job setup in Automation Engine. In many cases these words in the XML may not be in English. This terminology in the XML (in the business system) needs to be mapped to the official one in Automation Engine. This is explained in the documentation of the Create Job task.

See the correct settings for our example:

- The element name under which all the ink info is found is `InkSpecifications`.
- The parameter names in your XML (right) have been mapped to the standard names in Automation Engine (left):

```
<?xml version="1.0" encoding="UTF-8" ?>
<CreateJob>
  <Job>
    <InkSpecifications>
      <Ink>
        <ColorBook>Process</ColorBook>
        <ColorType>Normal</ColorType>
        <ColorName>Cyan</ColorName>
        <Angle>15</Angle>
        <Ruling>150</Ruling>
        <Dot>CS4</Dot>
        <PrintingProcess>Offset</PrintingProcess>
      </Ink>
      <Ink>
      <Ink>
      <Ink>
      <Ink>
      <Ink>
      <Ink>
      </InkSpecifications>
    + <Customer>
    + <Parameters>
  </CreateJob>
```
Note: These names are not case-sensitive.

Note: If one XML Element name is wrongly mapped, the Create Job task will not fail but only this wrong parameter will stay empty in the Job Inks setup.

- Alternatively, in our example, you can use this **Xpath Expression** to select that element describing all the inks:

This is the result in the created Job:
8.1.4 Step 4 - Test your Workflow

Test your workflow by launching it from within the Pilot.

In the Pilot, select the input XML file and launch the workflow. It should create the Job and the Job should contain all the parameters you had set up.

When the Job was created but one or more parameters were not found, the status of that 'Create Job' workflow step will be 'warning' (orange).

- The task’s details info will already tell you that "There were problems resolving some job fields".

- The task's log file will tell you what specific SmartName(s) has a problem. For example: "Could not resolve(2) job.customer : <NameOfTheSmartNameThatFails>/*

8.1.5 Step 5 - Add automation by starting this workflow from an Access Point

When an external system sends you a XML files describing Jobs to create, we advise to automate this workflow.

A typical way to do this is to create a Folder Access Point. The folder of that Folder Access Point is then the place where the XMLs will arrive. The workflow that the Folder Access Point will automatically start at a regular time interval is then the one you created in this example.

Note: Read more on setting up such automation in Folder Access Point on page 18.

Attention: It is also common for external systems to send the prepress department you an XML file containing the description of multiple job orders. And often those systems only send such a list once or twice per day. This use case is explained in a next example Creating Multiple Jobs from One XML on page 81.
### 8.2 Creating Multiple Jobs from One XML

**Summary**

**Note:** This example requires you to master the features explained in the previous example *Creating a Job using XML* on page 72.

You receive an XML from the business system. It is a list of job-orders and their production parameters. Some are already started in prepress, some are new, some are on hold. Your goal is to create Jobs of all those in the status ‘ReadyForPrepress’. The XML needs to be split into multiple XML files each describing one Job. These XML files are then input files for the *Create Job* task. The job-order info in the XML files is too elaborate. You will only pick up some selected parameters from those XML files.

Tools used in this example:
- **SmartNames** including *Xpath Builder*
- **Split XML** task
- **Data Splitter** workflow control
- **Create Job** task

#### 8.2.1 Step 1 - Analyze the XML describing Multiple Jobs

**Tip:** Click [this link](#) to find a ZIP with the sample XML file used in this example.

The XML in this example describes the 3 job orders. See how 2 of them have the status ‘ReadyForPrepress’.
"<?xml version="1.0" encoding="UTF-8"?>
  <Jobs>
    - <Job>
      <OrderID>1404-W86203</OrderID>
      <SubOrderID>003</SubOrderID>
      <Name>1404-W86203</Name>
      <Description>Promo strawberry 100ml South Europe</Description>
      <DueDate>2015-06-01T08:00:00.00Z</DueDate>
      <Status>ReadyForPrepress</Status>
      <ProjectID>Project193</ProjectID>
      <CustomerID>2712</CustomerID>
      <Category>flexo</Category>
      <CustomField1>Nipeter FB-2500</CustomField1>
      <CustomField2>White BOPP</CustomField2>
    </Job>
    - <Job>
      <OrderID>1404-W74536</OrderID>
      <SubOrderID>002</SubOrderID>
      <Name>1404-W74536</Name>
      <Description>New flavour Cocos and lemon</Description>
      <DueDate>2015-06-01T08:00:00.00Z</DueDate>
      <Status>OnHold</Status>
      <ProjectID>Project123</ProjectID>
      <CustomerID>2712</CustomerID>
      <Category>flexo</Category>
      <CustomField1>Gallus ECS 340</CustomField1>
      <CustomField2>OPP</CustomField2>
    </Job>
    - <Job>
      <OrderID>1404-W74599</OrderID>
      <SubOrderID>009</SubOrderID>
      <Name>1404-W74599</Name>
      <Description>New flavour Banana</Description>
      <DueDate>2015-06-01T08:00:00.00Z</DueDate>
      <Status>ReadyForPrepress</Status>
      <ProjectID>Project123</ProjectID>
      <CustomerID>2712</CustomerID>
      <Category>flexo</Category>
      <CustomField1>Gallus ECS 340</CustomField1>
      <CustomField2>OPP</CustomField2>
    </Job>
  </Jobs>

**Note:** In this example XML, we removed typical extra parameters about the customer, bar code and inks. The principle of those types was already explained in *Creating a Job using XML* on page 72.

**Attention:** If you send this XML to the **Create Job** task, it would only create the first Job it finds in the XML.

### 8.2.2 Step 2 - Create the Workflow Ticket

This is the workflow we need:
The Split XML task splits the XML into multiple XML files. If you don’t do this, the Create Job task would only create the first Job is finds in the XML.

The Data Splitter makes sure the next task sees the many input files. It prevents that the task is only launched on the first input file.

The Create Job task creates Jobs, as determined in the Data Splitter, for each incoming XML file.

**Note:** The Create Job has no output files. If you want to continue the workflow with its input files (the split XML files), then configure a specific output pin for that. Read more on these possibilities in *The Workflow Editor* and also *Building Workflows* in the User Guide.

### 8.2.3 Step 3 - Splitting the XML but also using a Filter on Status

In the Split XML task, we need this XPath Expression:

```
/Jobs/Job
[Status = 'ReadyForPrepress']
```

- The expression `/Jobs/Job` will result in splitting the XML on that level of elements: one per Job.
- The extra filter `[Status = 'ReadyForPrepress']` will limit that to jobs with that status.

This XPath expression can easily be built by using the Xpath Builder. Open it via the **Edit** button.

**Note:** When the filter does not find any matching ‘node’ (job), the task will error and mention this as the reason.

- **Output in:** Define where you want the split XMLs to be written.
- **File name:** Define the name of the split XML(s). The default setting will add a number.

### 8.2.4 Step 4 - Importance of the Data Splitter

When you do not use the Data Splitter, the Create Job task will receive all the XMLs generated by the Split task as 1 group on its input pin. In this case only the first incoming XML will be processed and the rest will be ignored. The Data Splitter step in this workflow will make sure the Create Job task receives the XMLs as separate files so each of them gets processed.

Because we (possibly) have multiple input XML files, we need these (default) settings:
We do not want the split XML files to be grouped. The Create Job task needs to be launched on each one of them.

Find more details on this task in Data Splitter.

8.2.5 Step 5 - Create the Xpath SmartNames

Identical as explained in the previous example Creating a Job using XML on page 72, you need to

- create SmartNames that you will use in the Create Job ticket. As explained in Step 2 - Create your SmartNames on page 73.

- insert those SmartNames in the Create Job ticket. As explained earlier in Step 3 - Insert the SmartNames in the Create Job ticket on page 76.

8.2.6 Step 6 - Workflow Result

Select the XML describing multiple Jobs and launch your workflow.

The Create Job task will be started per incoming XML. In this case 2 times, because only 2 had the status ‘ReadyForPrepress’.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Task Type</th>
<th>Progress</th>
<th>Phase</th>
<th>State</th>
<th>Launched</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Orders today_2.xml</td>
<td>Create Job</td>
<td>100%</td>
<td></td>
<td></td>
<td>7/11/14 2:49 PM</td>
</tr>
<tr>
<td>Job Orders today_1.xml</td>
<td>Create Job</td>
<td>100%</td>
<td></td>
<td></td>
<td>7/11/14 2:49 PM</td>
</tr>
<tr>
<td>Job Orders today.xml</td>
<td>Split XML File</td>
<td>100%</td>
<td></td>
<td></td>
<td>7/11/14 2:49 PM</td>
</tr>
<tr>
<td>Job Orders today.xml</td>
<td>Workflow</td>
<td>100%</td>
<td></td>
<td></td>
<td>7/11/14 2:49 PM</td>
</tr>
</tbody>
</table>

**Note:** Workflow controls like the Data Splitter do not appear in this task overview.

See the files the workflow created:
The 2 Jobs that had the status 'ReadyForPrepress' will be created:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Due Date</th>
<th>Order ID</th>
<th>Sub Order ID</th>
<th>Customer Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1404-W86203_003</td>
<td>Promo strawberry 100ml South Europe</td>
<td>Jun 1, 2015</td>
<td>1404-W86203</td>
<td>003</td>
<td>FruitCo</td>
</tr>
<tr>
<td>1404-W74599_009</td>
<td>New flavour Bananas</td>
<td>Jun 1, 2015</td>
<td>1404-W74599</td>
<td>009</td>
<td>FruitCo</td>
</tr>
</tbody>
</table>

8.3 Create Multiple Jobs from a CSV file

**Summary**

Every morning, you receive a CSV file from the planning department with a list of job orders for the prepress department. Each row in the CSV represents a job order. This data needs to be converted into separate XML files to send to the **Create Job** task.

**Note:** a CSV is a Comma Separated Value file. This is usually an exported file from a database or an XLS file.

Tools used in this example:
- **SmartNames** including **Xpath Builder**
- **Map Data** task
- **Split XML** task
- **Data splitter** workflow control
- **Create Job** task

8.3.1 Step 1 - Analyse the CSV file

**Tip:** Click [this link](#) to find a ZIP with the sample CSV file used in this example.

Our example CSV contains a description of 2 job orders.
Because the **Create Job** task does not work with CSV files, we first need to create an XML. And as you saw in the previous example *Creating Multiple Jobs from One XML* on page 81, we will need 1 XML per Job to create.

**Note:** At this point it is good to detect any data fields that will need reformatting. In this example workflow, the format of the due date will need to be mapped to the format that is required for that field in the Job Setup.

### 8.3.2 Step 2 - Create the Workflow Ticket

This is the workflow that we need to create Jobs from a CSV with descriptions of multiple job orders:

- The **Map Data** task defines what parts of the CSV will be mapped into the XML file.
- The **Split XML** task splits that XML into multiple XML files, like we did in the *previous example*.
- The **Data Splitter** makes sure that the next task sees the many input files. It prevents that the task is only launched on the first input file.
- The **Create Job** task creates Jobs, one for each incoming XML file.

### 8.3.3 Step 3 - Mapping the CSV to XML

The **Map Data** ticket will
- convert the CSV to XML
- define which parts of the CSV are needed in the XML
- define how the parameters should be named in the XML. You can keep their names if you want to.
- define what part in the XML will be **Repeating Content**. This is needed because you are not sure how many jobs there will be in that CSV / XML. The CSV that you receive tomorrow can have a different amount of rows. This is the `for each` principle as explained earlier in *Specifying the Output File* on page 54.

**Note:** The order in which we define the settings in this ticket is important.

1. **Define the CSV as input file**
   - Select your CSV file and open the workflow ticket that you made in the previous step.
   - Double click the **Map Data** step. This automatically sets your CSV as input file:
• Now set 2 specific options that define how our example CSV should be read. Double click the line 'Input File 1' to open its setup. Set the Field Separator to Semicolon. Because the first line of the CSV contains the field headers (not values we want to read), check the option Ignore first line. Remember that your specific CSV might be different, so do check all options.

• Click OK to close that dialog.

• Maybe now is a good time to already define the folder and name of your output XML. As usual, you can use SmartNames in Output in: and File Name:

• Click Continue.

• Make sure the Output Format: is on 'XML'. In the Options... make sure to activate the Replace reserved characters in parameter values with XML entities. In this example, there are no such characters.
2. Define the Repeating Content in the XML

**Note:** Make sure you have already read *the chapter documenting this advanced technology*.

This ticket allows many ways to use it. Following steps are specific for this CSV to XML workflow:

In the **Map Data** ticket, in the canvas where you define the output XML,

- Right-click and choose **Insert XML Declaration**. This is just good practice when creating XML.
- Right-click and choose **Insert Element**. Give it a logical names, for example `<Jobs>`
- Put your cursor in between the open and closing element, right-click and choose **Insert Repeating Content** (or click on ± Ξ).

**Remember:** You can not make any existing content 'Repeating' *later*; you need to create the 'Repeating Content' block first and then add content inside.

Every line (row) in your CSV represents a job order. So you do want **Every line** in your CSV to become the Repeating Content. Remember we already decided to ignore the first line of the CSV.

- Click OK. A pink field for **Repeating Content** is created. Also the field parameters were created automatically. They are colored pink because they are linked to the selected **Repeating Content** field.
Now, inside that **Repeating Content** field, continue specifying the content of the XML. Insert Elements and click the pink parameters in between the opening and closing element.

**Caution:** Make sure your XML elements do not contain spaces!

This is a basic and valid example:

```xml
<Job>
  <OrderID>[Order ID]</OrderID>
  <DueDate>[Due Date]</DueDate>
  <CustomerName>[Customer Name]</CustomerName>
  <CSR_email>[CSR email]</CSR_email>
</Job>
```

You can see we only use 5 of the parameters in this example. You can of course pick up a lot more.

**Note:** Remember that the XML file will just be an in-between format. You will later, in the **Create Job** ticket, still map the XML elements to the real fields in the Job Setup.

- Click **OK** to close the set up of your **Map Data** ticket.
- Save your Workflow ticket. No need to close it.

### 8.3.4 Step 4 - Test your CSV to XML Mapping

We advise to already test what we set up in our **Map Data** ticket. Because if that output does not look good, it is better to correct that now instead of using a wrong XML file as input for the next workflow steps.

With the CSV as input file, launch the workflow. Because we haven't defined the settings for the next workflow steps yet, the workflow will surely fail.

You now see that your CSV was mapped to an XML. Double click the output file to open it. It should look like this:
You see

- the main structure /Jobs/Job
- the description of the 2 Jobs,
- including the 5 parameters that we picked up from their definition in the CSV.

When your XML is not OK, then solve the problem first before defining the next workflow steps.

8.3.5 Step 5 - Set up the Split XML step

- Leave the workflow ticket and go back to your Pilot.
- Go to the folder where you wrote the output of the Map Data ticket and select the Orders_for_Prepress_CSV2XML.XML
- Open the default Split XML ticket. Your XML will be seen as input file.
- Click Edit to open the Xpath Builder.
- Click on a Job node
The resulting Expression will be shown above: /Jobs/Job.

- Click OK to close the dialog.

- Your ticket is ready. Save it with a logic name.
- Open your workflow and replace the default Split XML step with this specific one.
- Save your workflow.

### 8.3.6 Step 6 - Test your workflow again

- Select the CSV and launch your workflow. Because we haven't defined the settings for Create Job ticket yet, the workflow will surely fail.

What you now need to see is the 2 XMLs as output of the Split XML ticket.

- Double-click one of the split off XML files to open it. It should look like this:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Jobs>
  <Job>
    <OrderID>EXA_CSV_Job01</OrderID>
    <DueDate>30-09-14</DueDate>
    <CustomerName>FruitCo</CustomerName>
    <CustomerNr>2712</CustomerNr>
    <CSRemail>Franky.fruit@fruitco.com</CSRemail>
  </Job>
</Jobs>
```

### 8.3.7 Step 7 - Check the Data Splitter

The Data Splitter control in your workflow should have the (default) settings to 'not group'. It's purpose is identical as in previous example, where we explained the importance of the Data Splitter.
8.3.8 Step 8 - Defining the SmartNames for the Create Job ticket

Making the SmartNames and inserting them into your Create Job ticket is not different here than explained in earlier examples, like in Step 2 - Create your SmartNames on page 73 and Step 3 - Insert the SmartNames in the Create Job ticket on page 76. We will first briefly recapitulate that part and then we will pay extra attention to setting up the SmartName for the Due date.

Creating the necessary SmartNames

Follow these steps to create the SmartNames:

- In your Pilot, go to the SmartNames tool.
- On the right, in Resolve all using:, select A file: and browse to one of the split off XML files (for example Orders_for_Prepress_CSV2XML_1.xml).
- Create Xpath Query SmartNames for
  - OrderID
  - CustomerName
  - CustomerID
  - CSRemail

Creating the SmartName for the Due Date

- In your Pilot, go to the SmartNames tool.
- Click on to create a New SmartName.
- Choose the type: Xpath Query.
- Click on Edit. In the XPath Builder, select the value for the element DueDate.

XPath Expression:
/Jobs/Job/DueDate/text()

1 match found
Example XML Content:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Jobs>
  <Job>
    <OrderID>EXA_CSV_Job01</OrderID>
    <DueDate>30-09-14</DueDate>
    <CustomerName>FruCo</CustomerName>
    <CustomerNr>2712</CustomerNr>
    <CSReemail>Franky.fruco@frutco.com</CSReemail>
  </Job>
</Jobs>
```

- In the setup of an Automation Engine Job, the due date needs to be specified according the ISO 8601 standard. We therefore need to reformat that date from how it was defined in the CSV (‘30-09-14’). Now set the formatting options as follows (see screen shot below):
  - Click on Formatting Options....
  - Set the Data Format to Date and Time.
• Set the Format to ISO 8601.

• Choose your Time Zone (the one of your Automation Engine server). Below left in the dialog, you will see that the Result is still not OK.

• Click on Input Format Options....

• Set the Format to Custom.

• Set the Language.

• Choose Custom Format. Now write how your data was formatted in the CSV: dd-MM-yy.

  Caution: The month is here defined by an ‘M’ in capitals! A lowercase ‘m’ will be seen as minutes!

• Click OK. The correct result should now be visible in the Formatting Options dialog.

Now save your SmartName and give it a logic name. For example "From XML - CSV Due date for ISO".

The resulting Create Job ticket

• Open the workflow and double-click the Create Job ticket.
• Add your SmartNames. It could look like this:

<table>
<thead>
<tr>
<th>Job</th>
<th>Customer</th>
<th>Info</th>
<th>Parameters</th>
<th>Bar Codes</th>
<th>Template</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order ID:</td>
<td>[EXA XML OrderID]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub Order ID:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due Date:</td>
<td>[From XML - CSV Due date for ISO]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project ID:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printing Press:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substrate:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job Folder:</td>
<td>q/Automation Engine/Public/democ_container/[EXA XML OrderID]_Braze</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Now save your workflow ticket.

8.3.9 Step 9 - Workflow Result

In the Pilot, select your CSV file and launch your workflow. It will look like this:

- OK
- Orders_for_Prepres_CSR2XML.xml
- Orders_for_Prepres_CSR2XML_1.xml
- Orders_for_Prepres_CSR2XML_2.xml

It will have created these 2 Jobs:
8.4 Creating Jobs using a Database Access Point

Summary

We will use a Database Access Point to query an external database (one from a business system where the job orders originate). The query will, at a regular time interval, ask that database which new Jobs need to be created on Automation Engine. The result of this query are XML files with the description of those Jobs. Those XML files will be used to create the Jobs (as shown in previous examples).

Tools used in this example:

- An external database
- A Database Access Point
- SmartNames including Xpath Builder
- The Data Splitter workflow control
- The Create Job task

8.4.1 Step 1 - Analyse the External Database

Tip: Click this link to find a ZIP with the sample database used in this example.

This is the database we will query in this example:

- The database software type is a **MS SQL server**.
- It runs on a a server computer named **EAW11DL306**.
The name of the database that we will access is **MyExternalSystem** (there could be more served by this same software).

The name of the (opened) table is **MyOrders**.

The table contains the description of 6 job orders.

The **Status** column shows that 3 job orders are 'Ready For Prepress'. Those are the ones that we want to create on Automation Engine.

### 8.4.2 Step 2 - Configure the Link to the External Database

Set up the link with the external database. This is done in via **Tools > Configure > External Databases**. Read more on such configurations in **External Databases**.

For this example, this is the configuration we need:

![Configuration Interface](image)

### 8.4.3 Step 3 - Set up the Database Access Point

**Note:** Setting up **Database Access Points** is documented in **Create or modifying a Database Access Point** on page 27.

This is an example of the Access Point we need:
• In the **Pilot**, create a new **Access Point** and choose the type **Database**.

• Give it a logical **Name** and **Description**.

**Database Settings**

• Select the **External Database** you configured in the previous step.

• Enter the **SQL Select Query**:
  
  • `'SELECT' indicates which columns you want to get a value from.
  
  • `'FROM' indicates the database **table** where these columns are.
  
  • `'WHERE' indicates an extra filter: you do not want all the jobs from that table, but only those with the status 'Ready For Prepress'.

**Note:** SQL queries are case-sensitive!

**Note:** It is possible to select values from different tables, for example if you want to get more details about the customer or about production tools. This examples does not do that.

**Note:** Esko does not offer training or support in database queries. This is mentioned in more detail in *Concept of tasks interacting directly with external systems* and *Concept of the Interact with Database task*.

• **After Processing:** We here use **Update the records** to change the status field in the external database to 'InPrepress'. This is necessary to avoid that, after the next time interval, these jobs would be created/updated again (because they would still be in the status 'Ready For Prepress').
Tip: You can choose to not do this status-update here but rather wait until the workflow really has created the Job. In the following workflow, you can then use an extra Interact with Database ticket to update that status in the external database with for example a status 'Job Created and Started in Prepress'.

Monitoring
Define the time interval with which the query should be done.

Note: There is no need to set it to ‘1 minute’ to make testing easier. See further how we will use the scan now button instead.

Automatic Actions

The query will write an XML file for each job order that it finds in the specified status.

- **Move Files to:** Define where you want those XML files to be written. The Job they represent does not exist yet so we advise to write them in some central folder. See an example in the screen shot.

- **Rename Files to:** Their default name [File] actually consists of [table_name] [date-and-time]. You can overrule this or add more SmartNames. Also see later when we show the result.

- **Action:** We will add a workflow in later steps of this example. For the moment, leave this to None.

Click OK to confirm your settings

Your Database Access Point now exists.

Activate the Access Point

Right-click and choose Activate Access Point.
8.4.4 Step 4 - Test your Database Access Point

Select your Database Access Point and click the button to scan now. This will execute the SQL query immediately.

An Alert window will appear with the result. In our example, we see

- 3 XML files were created because 3 job orders had the status that we were asking for: 'Ready For Prepress'.
- Their name is the result of the SmartNames as explained in the previous step.

This is one of the resulting XML files:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<MyOrders created="2014-08-01T12:52:56.928+02:00" producer="DBPoller-MyExternalSystem"
  select="SELECT OrderNr, CustomerName, CustomerId, DueDate, ProofType, Process, Status FROM MyOrders WHERE Status = 'Ready For Prepress'">
  <OrderNr>1235</OrderNr>
  <CustomerName>Graceland</CustomerName>
  <CustomerId>6666</CustomerId>
  <DueDate>27-12-14</DueDate>
  <ProofType>FTP-PDF</ProofType>
  <Process>Screen</Process>
  <Status>Ready For Prepress</Status>
</MyOrders>
```

See how the XML file contains the query that was used.

These XML files will now be used as input files to create Jobs.

Check the new status in the database

Remember the Update the records setting in previous step. See how, for the 3 job orders that first had the status 'Ready For Prepress', the status has now changed to 'InPrepress':
Attention: When testing this Access Point, do remember that every time you run it that the status of those job orders will change to 'InPrepress'! This means that the next time the Access Point will do nothing simply because there are no more job orders in that database with the required status 'Ready For Prepress'! So, in between tests, you have to manually reset those statuses to 'Ready For Prepress'. And remember that the queries are case-sensitive!

8.4.5 Step 5 - Create your SmartNames

Identical to how we did this in step 2 of the first example, we will now create the SmartNames that will map the fields in the XML to the Job setup fields in the Create Job task.

1. In the Pilot, go to the SmartNames View.
2. On the right, in Resolve all using: check the button for A file and browse to one of the XML files you created in the previous step.
3. Click on Create a New SmartName.
4. In the dialog, indicate the type Xpath Query.
5. Click on Edit to open the Xpath Builder.
6. In the XML content, highlight the first value you want to make a SmartName for. For example OrderNr.

The resulting Xpath Expression will appear.
7. Click OK.
   You will see the result value 1235 appear bottom left in Result:
8. Click **Save As...** and give the new **SmartName** a logical name. For example **EXA_DB_Myorders_OrderNr**.

9. Keep the dialog open and click again on **Edit**. Repeat steps 6 to 8 for the next **SmartNames** that you want to make in this example, **except the SmartName for the Due Date**, that one needs an extra formatting. See next step:

10. Creating the **SmartName** for due date is here identical to how we did it in the previous example. You need to make sure the output format is according the ISO standard for this Job setup field. And you need to indicate how the input is constructed. In our example this is **dd-MM-yy**.

11. When finished saving all **SmartNames**, close the dialog (press **Cancel** or use **Save** for your last **SmartName**)

12. Back in the **SmartNames View**, close the dialog (press **Cancel** or use **Save** for your last **SmartName**)

You should now see this:

---

**8.4.6 Step 6 - Create the Workflow with the Create Job task**

To have the Jobs created automatically, create this workflow:

1. In the **Pilot**, go back to the **Files** or **Job** mode and click on ![Create new workflow](image) to create a new workflow.

2. Build a workflow that starts with a **Data Splitter** (keep its default settings `'don't group output'`), followed by a **Create Job** ticket.

3. Already save your workflow with a logical name. Remember we will later attach it to the **Database Access Point**.

4. Double click the **Create Job** ticket and add the **SmartNames** to their matching fields.

   It should look like this:
The **Job Folder** setting in the screen shot is just an example of many ways to define it.

5. Save your workflow.

### 8.4.7 Step 7 - Test your workflow

In your **Pilot**, go to that (central) folder where your 3 XML files were written. Select these 3 input files and launch your workflow.
See how the Data Splitter receives 3 XML files and hands them over one by one to the Create Job ticket that creates these 3 Jobs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Order ID</th>
<th>Customer Name</th>
<th>Customer ID</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1225</td>
<td>1235</td>
<td>Graceland</td>
<td>6666</td>
<td>Dec 27, 2014</td>
</tr>
<tr>
<td>1226</td>
<td>1236</td>
<td>FratCo</td>
<td>2712</td>
<td>Oct 11, 2014</td>
</tr>
<tr>
<td>1237</td>
<td>1237</td>
<td>FratCo</td>
<td>2712</td>
<td>Dec 2, 2014</td>
</tr>
</tbody>
</table>

Tip: Also check again if the due dates are formatted correctly.

8.4.8 Step 8 - Add the workflow to the Access Point and test again

Open your Database Access Point. In Action, choose Launch Workflow and select your workflow. Click OK to confirm.

According the set time interval, your Database Access Point will run and now also use its resulting XML files as input files for your workflow.

When your Database Access Point is selected, you see the workflow task appear:
8.5 Launching a workflow with parameters from an XML

Summary
A workflow is automatically launched when the business system writes an XML file on a Folder Access Point. The XML file describes what to do with what file: it contains a full path to a new design file and describes many parameters that will be used as Workflow Parameters in the workflow that will be launched.

Such a concept is used by business systems that trigger a workflow on Automation Engine based on a new status. For example: when the business system notices that the status of a job order changes to 'Design Arrived', it decides to already start a prepress workflow on Automation Engine (the one that is typical for new designs from that customer). It is also common that such setups are not limited to only one status leading to one workflow. Some customers elaborate on this process and have 2 or 3 such links between a status in the business system and an automatically launched workflow on Automation Engine.

Tools used in this example:

- **SmartNames** including the Xpath Builder
- **Workflow Parameters**
- **A Folder Access Point**
- **These tasks:** Preflight with PitStop, Create Report (ReportMaker), Zip, Send E-mail, Upload via FTP
- **These workflow controls:** Select Referenced File, Router, Data Collector.

Note: In this example we do not use Automation Engine’s Job concept.

Note: This example assumes you are already comfortable making advanced workflows.

8.5.1 Step 1 - Analyse the incoming XML

Tip: Click this link to find a ZIP with the sample XML file used in this example.

This is the type of XML file that we receive from the business system:
In the XML file, notice

- the job order number and some customer details
- an absolute full URL to the design file that just arrived. The design is named 'Shrek.pdf'.
- some decisions if the workflow needs preflighting or creation of a report card.

8.5.2 Step 2 - Create the Workflow

Now create the basic structure of our workflow. Add the steps as in the screen shot. We will add more detail later. It is possible that some of the connections can not be made yet.

Notice these main process steps:

- the workflow needs to run on the PDF, not on the incoming XML file that will land on the Folder Access Point (that we will create later in this example). That is why we will also use Select referenced File to select that 'Shrek.pdf' (see further).
- the workflow parameters will indicate whether the route to Preflight with PitStop and/or route to the Create Report (ReportMaker) needs to be taken.
- if the Router confirms that preflighting is needed, the step will do that and use a Profile defined in the XML.
- if the Router confirms that a design report card is needed, the Create Report (ReportMaker) will create that.
the Data Collector collects the Preflight Report and/or the Design Report card, after which they are ZIP'ed together.

the XML will indicate the next router if this ZIP needs to be sent to the customer via E-mail or via FTP.

Save your workflow with a logical name, for example EXA_XML_Workflow.

8.5.3 Step 3 - Load Workflow Parameters from the XML

The XML files that the external system sends will always have the same elements, but with mostly different values. This is a typical case to use Workflow Parameters.

Note: Read more about Workflow Parameters in Workflow Parameters.

1. Open the workflow you created and in the menu go to Advanced > Manage Workflow Parameters.
2. Check Use values from XML file and enter these SmartNames [URL of Folder of Input]/[File].xml (see also in below screen shot).
3. Click on Load Workflow Parameters from XML... (below right). Browse to your XML example file and click Open. Choose to Leave blank to select all elements below the root element. All the XML element names will be listed, with their value from in this specific example XML:

4. Click on X to close this dialog.
These parameters are now Workflow Parameters. Every time this workflow is launched, they will be "loaded" before even starting the first workflow step.
5. Save your workflow.

8.5.4 Step 4 - Select the Design file that is Referenced in the XML

The workflow needs to run on the design file, not on the XML file. The design file is referenced in the XML file.
1. In the Pilot, select the example input XML file and then open the workflow you created.
2. In the workflow, open the step Select Referenced File.
3. In Select by:, choose File Path.
4. In File Path, click on Edit to open the Xpath Builder.
5. In the Xpath Builder, the input XML file of your workflow will be shown as Example XML Content.

Select the path (the value of the Design element). On top, the resulting Xpath Expression will be shown.

6. Click OK to close the Xpath Builder.
   The Xpath Expression is now copied as File Path.
7. Check the option Provide a warning when no files were selected.

8. Click OK to confirm these settings.
9. Save your workflow.

8.5.5 Step 5 - Pick up the Workflow Parameters in your Workflow

We refer to the standard documentation to learn more on making workflows or on the tasks and workflow controls used in this example. We will here focus on what is specific in our example workflow.

Our workflow does not use the Job concept and we (therefore) use Workflow Parameters:

- in the 3 Routers
- in the Preflight with PitStop step
- in the Create Report (ReportMaker) step
- in the Send E-mail step
- in the Upload via FTP step.

In the 3 Routers

A value in the XML decides where the workflow will route to. Follow these steps for each Router (they are very similar in all 3 routers):

1. In the workflow, open the Router.
2. In Route based on, choose SmartName value from the list.
3. In SmartName, pick the appropriate one from the list, category 'Workflow Parameters' ([WFP.]
   and click Insert.

As can be seen in below screen shot, the 1st Router in the workflow needs the SmartName [wfp.PreflightYesNo]. The 2nd Router needs [wfp.ReportCardYesNo]. The 3rd Router needs [wfp.SendViaFTPorEMAIL].

4. Close the list of SmartNames by clicking on the X.
5. Now set the relation between the SmartName value and the route to follow. See this example for the 1st **Router** deciding if preflight should be done:

![Router for preflight decision](image1)

See this example for the 2nd **Router** deciding if a report card should be created:

![Router for report card decision](image2)

See this example for the 3rd **Router** deciding whether FTP or E-mail should be used:

![Router for FTP vs. E-mail decision](image3)

6. Click **OK** to confirm the Router settings and close the workflow step.

7. Save your workflow.
In the Preflight step

In our example, the XML file also decides which preflight profile needs to be used.

Note: See the full documentation of this workflow step in Preflight with PitStop.

1. In the workflow, open the step Preflight with PitStop.
2. Define the Output folder and name. Mind that this is not for the preflight report file.
3. In the Preflight tab, in Profile, insert the Workflow Parameter SmartName for the profile as part of the path.

4. In the Report tab, define the output folder and name for the preflight report that you will send to the customer.

5. Set any other options as wished.
6. Click OK to confirm the settings and close this workflow step.
7. Now drag the logical connections between the steps. The pin that represents the output of the preflight report should link to the Data Collector. That file will be sent to the customer, the input design file will not be sent.
8. Save your workflow.

In the Send E-mail step

In our example, the XML file decides if the resulting files will be e-mailed or FTP’d to the customer.

**Note:** See the full documentation of this ticket in *Send E-mail*.

1. In the workflow, open the step **Send via E-mail**.
2. In **To**, add the **Workflow Parameter SmartName** for the E-mail of the customer contact for this job order.
3. In the **Subject** line, add the **Workflow Parameter SmartName** for the job order.
4. In **Message**, add the **Workflow Parameter SmartName** for the name of the customer contact.
5. Fill in extra fields as wished

6. Click **OK** to confirm the settings and close this workflow step.
7. Save your workflow.

In the Upload via FTP step

In our example, the XML file decides if the resulting files will be e-mailed or FTP’d to the customer.
Note: See the full documentation of this ticket in Upload via FTP.

1. In the workflow, open the step Upload via FTP.
2. Enter the name of the Host.
3. In Folder, use the Workflow Parameter SmartNames to send the ZIP into a folder with a logical name. For example:

4. Set other settings as wished.
5. Click OK to confirm and close this dialog.
6. TIP: Optionally, you can send the customer contact an E-mail that his files are on the FTP server. Use the Notification tool of this workflow step to do this. Insert the Workflow Parameter SmartName for the E-mail and name, similar to how we did it in the settings of the 'Send E-mail' step.
7. Click OK to confirm the settings and close this workflow step.
8. Save your workflow.

8.5.6 Step 6 - Data structure of this Workflow

This is a workflow that does not use the Job concept, so you can not store all your data in sub folders of a Job Folder.

The data structure that this example workflow creates looks like this:

- The folder MIS-XML-WFL DATA contains all the data that this workflow creates. See the logical names of the (automatically) created sub folders. These folder names were set in the 'Output in' fields of the relevant workflow steps.
The folder MIS-XML-WFL HOT will be the folder used for the Folder Access Point.

8.5.7 Step 7 - Test the Workflow

1. In the Pilot, select your input XML file.
2. Then select the workflow you created and open it (double click).
3. Launch and Monitor: Press CTRL-ALT-Enter or Press ALT while clicking .
4. Drag the output file numbers so you can see these output files:

8.5.8 Step 8 - Create the Folder Access Point and Test Again

Note: If you are not familiar yet with Access Points, then first read Access Points on page 17.

1. In the Pilot, go to the Access Points view.
2. Create a new Access Point and choose the type Folder.
3. Give it a logical Name and Description.
4. In Folder Settings, indicate the folder where the external system will write the XML files.
5. In Monitoring, set your preferred time interval.
6. In Automatic Actions, indicate the folder where the XML files should be moved to once they are processed.
7. Set the Action to Launch Workflow and select the workflow of this example.
8. Click OK to confirm and close.
9. With the new Access Point selected, wait until the time interval passes or press 'Scan now' to see the resulting tasks appear:

The screen shot shows 2 extra tests where the Preflight ended with a warning.
10. Check the workflow in more detail (output files and folders, XMLs with different options).

8.5.9 Step 9 - Possible Workflow Extensions

Here are some ideas to make this workflow even more powerful:

Feedback new status to MIS

It would be logic to inform the (business) system that sent you the XML on the fact that the workflow has finished. Here are some tips on how to do that:

- Adding a Create XML step to the workflow and send an XML with the status on a folder that that system will scan.
- Adding a Interact with Database step to the workflow and have a database query write that status right back into that system's database.
Note: We can not use Milestones here because we are not working in a Job context.

Read more on such possibilities in Automation Engine sending data Back to the External System on page 15

Adding the Preflight Status already in the Email or Report File name

The customer might want to first check the designs where the Preflight showed an error or warning. To make this possible, you could extend the workflow to capture the Preflight status and use it as a SmartName. You could then insert that SmartName

- in the Email subject and or message
- in the FTP folder or name of the ZIP.

8.6 Getting info from a Web Service into a Workflow

Summary

Your company has printing plants in site A and site B. Automation Engine is in headquarters (‘site HQ’) where also the business system is. To enable a cheaper and faster delivery, your planning system asks Automation Engine which printing plant is closest to the customer, where the printed product should be shipped to (‘site C’).

Automation Engine asks the web service of GoogleMaps the distances and reports the result back to the planning system. The planning system sends its request to Automation Engine via an XML file. Automation Engine will also send a result back as an XML file.

Tools used in this example:

- SmartNames using Xpath Builder
- Workflow Parameters
- Interact with Web Service task
- Map Data task
- The Data Collector and Sort workflow control

8.6.1 Step 1 - Analyse the Input XML

Tip: Click this link to find a ZIP with the sample XML file used in this example.

In our example, this is the XML file that we get from the planning system (RouteCalcRequest.xml):
For job-order 234567, the planning system asks which printing site is closest to the delivery site in Memphis: A (Nashville) or B (New Orleans).

Note: This example contains address as they are written in the USA.

8.6.2 Step 2 - Configure the Web Service

As documented in Interact with Web Service task on page 59, the GoogleMaps API needs to be added in Configure > External Web Services.
8.6.3 Step 3 - Create the workflow

Create a workflow with these steps. We will define some details later.

1. In Pilot, select the input XML file and create a workflow with these steps (the 2 'GoogleMaps' steps are Interact with Web Service steps.)

   ![Workflow diagram]

   See how the workflow will use a Map Data step to join and partially redefine the outputs from the 2 interactions with GoogleMaps. Alternatively, you could make this workflow by using a Map Data step after each interaction with GoogleMaps step, and then later end it with a Join XML step.

2. Edit the names of the 2 Interact with Web Service steps to those as in the screen shot.

3. Save your workflow with a logical name. For example "EXA - WEB - RouteCalc".

8.6.4 Step 4 - Load Workflow Parameters from the XML

This example does not use the Job concept so we will create Workflow Parameters and pick those up throughout the workflow.

1. Open the workflow you created and in the menu go to Advanced > Manage Workflow Parameters.

2. Check Use values from XML file and enter theis combination of SmartNames: [URL of Folder of Input]/[File].xml (see also in below screen shot).

3. Click on Load Workflow Parameters from XML... (below right). Browse to your XML example file and click Open. Choose to Leave blank to select all elements below the root element.
All the XML element names will be listed, with their value from in this specific example XML:

![Workflow Parameters dialog](image)

4. Click on X to close this dialog. These parameters are now **Workflow Parameters**. Every time this workflow is launched, they will be "loaded" before even starting the first workflow step.

5. Save your workflow.

### 8.6.5 Step 5 - Define the 2 Interact with Web Service steps

The settings for these 2 steps are almost identical. We will explain the ones for step 'A' and at the end show the differences with those for 'B'.

1. Double-click the step 'GoogleMaps A' to open it.
2. Set logical values for the **Output in** and **File Name** fields.
3. Choose the web Service you configured for Google Maps and choose the Method GET.

4. Enter the Path as defined in the API documentation of Google Maps. Because we are only interested in the distances, we here use Google Map’s API 'distancematrix' and not the API 'directions'.

5. In Parameters, use Workflow parameters to define the API's key origins and destinations. Unlike in Google Map's API 'directions', the keys 'origin' and 'destination' are both in plural here! The comma or space between the parts of the address needs to lead to a valid address specification for Google Maps.

6. The key units is optional. The default already is 'imperial' (miles, feet, inches ..)

7. Press OK to confirm and close this dialog.

8. Now double-click the other step 'GoogleMaps B' to open it.

9. Make sure the File name of the output XML file is different! And for the key origins, use the Workflow parameters that match the printing site 'B'.
10. Press OK to confirm and close this dialog.
11. Save your workflow.

### 8.6.6 Step 6 - Settings to Collect and Sort the 2 Responses

It is important in this workflow to keep control over the order of the 2 XML files that Google Maps will return us. You do not want to risk mixing up the distances for site A or B.

**Settings for the Data Collector step:**

Keep the default settings: **Group all collected data**

**Settings for the Sort step:**

Keep the default settings: **Sort by file name in ascending order**

Save your workflow.

### 8.6.7 Step 7 - First test to get the Google Maps result files

To define the settings for the **Map Data** step, we need the result XML files from the interactions with Google Maps.

1. In **Pilot**, select the input XML file RouteCalcRequest.xml and launch your workflow **EXA - WEB - RouteCalc**. It will probably fail, because we haven't defined the **Map Data** step yet.
2. You should get these 2 files:
Right-click and choose 'Open' to inspect them.

This is what Google Maps returns us when we ask the distance between site A and the delivery site:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<DistanceMatrixResponse>
  <status>OK</status>
  <origin_address>2804 Opryland Drive, Nashville, TN 37214, USA</origin_address>
  <destination_address>706 Union Avenue, Memphis, TN 38103, USA</destination_address>
  <row>
    <duration>3 hours 19 mins</duration>
    <distance>358003</distance>
    <text>222 mi</text>
  </row>
</DistanceMatrixResponse>
```

And this is what Google Maps returns us when asking the distance between site B and the delivery site:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<DistanceMatrixResponse>
  <status>OK</status>
  <origin_address>726 Saint Peter Street, New Orleans, LA 70116, USA</origin_address>
  <destination_address>706 Union Avenue, Memphis, TN 38103, USA</destination_address>
  <row>
    <duration>5 hours 38 mins</duration>
    <distance>634574</distance>
    <text>394 mi</text>
  </row>
</DistanceMatrixResponse>
```
Note: In Google Maps API, the default value unit for time is seconds and the default value unit for distance is meters.

8.6.8 Step 8 - Settings in the Map Data step

Next thing to do in this workflow is merge this information into one XML file.

1. In your workflow (edit model), open the Map Data step. You will see that it proposes the workflow’s input file (RouteCalcRequest.xml) as example input file. This is not what we want here!

2. Select the line with that default example Input File and double-click to open it. Browse to the first input file we really want as example: GoogleMaps_result_A.xml, which we wrote in a subfolder GoogleMapsFeedback. Click OK to confirm.

3. Click Add and browse to the 2nd file we got back from Google Maps and add it as 2nd example Input File. Click OK to confirm.

4. This also a good time to set a logical folder and file name for this output XML file that we will feed back to the planning system. Let’s assume that our output folder is a hot folder of that planning system. Use the Workflow Parameter [wfp.OrderNr] in the file name.

5. Click Continue to start defining the XML content that we want to send back.

6. We will first pick up some parameters from the example input files. On the right, in Parameters, click Add.

7. Click Edit to open the Xpath Builder and highlight the value of the <text> version of the element <distance> (222 mi).
8. Click OK to close. In the Add Parameter dialog, enter the name 'Distance_From_A' and click OK to close.

9. Do the same for the value of the <text> version of the element <duration> (3 hours 18 mins).

10. Then create the same 2 parameters but using the other example input file 'GoogleMaps_result_B.xml' ('394 mi' and '5 hours 38 mins'). You end up with these 4:

11. Now specify the output XML as shown in below screen shot. See how we use a mix of Workflow Parameters and Parameters from the example Input Files.
12. Click **OK** to close this workflow step.
13. Save your workflow.

8.6.9 Step 9 - Test the complete workflow

Our workflow is now complete (we assume that the place where the **Map Data** step writes the output XML is monitored by the planning system).

1. In **Pilot**, select the input XML file `RouteCalcRequest.xml` and launch your workflow 'EXA - WEB - RouteCalc'.
2. Check the flow and names of the output files. It should look like this:

![Workflow Diagram]

3. Right-click and open the final output file '2345657_RouteCalc_Result.xml'. It should look like this:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<RouteCalcResult>
  <OrderNr>234567</OrderNr>
  <CustomerID>6666</CustomerID>
  <CustomerName>Graceland</CustomerName>
  <Delivery_City>Memphis</Delivery_City>
  <PrintSiteA_City>Nashville</PrintSiteA_City>
  <PrintSiteA_Distance>222 mi</PrintSiteA_Distance>
  <PrintSiteA_Duration>3 hours 18 mins</PrintSiteA_Duration>
  <PrintSite5_City>New Orleans</PrintSite5_City>
  <PrintSite5_Distance>394 mi</PrintSite5_Distance>
  <PrintSite5_Duration>5 hours 38 mins</PrintSite5_Duration>
</RouteCalcResult>
```

The planning system will conclude that, for this job order, the printing site in Nashville is closest.
8.7 Updating a Product Status in an External System via Interact with Database

**Summary**

Our workflow includes changing the status of a Product Part to Waiting for Approval. Because this information is important for the service and sales staff, we will also inform the business system of this new status.

**Note:** In this example, we assume you are familiar with the Products tool. Learn more in the dedicated chapter Products.

Tools used in this example:

- Products tool
- Manage Product Status task
- Interact with Database task, to an external MS SQL Server
- Mark File and Select Marked File task
- ReportMaker task
- Send E-mail task

### 8.7.1 Step 1 - Analysis and Setup of the External Database

**Tip:** Click [this link](#) to find a ZIP with the sample database file used in this example.

**Structure of the External Database Table**

This is the database we will interact with:
• The database software type is a MS SQL server.

• It runs on a server computer named EAW11DL306.

• The name of the database that we will access is MyExternalSystem.

• The name of the (opened) table is ProductionItems.

• The table contains the description of 4 production items.

• The item 'yolo juice' shows its 'ItemStatus' on 'Design arrived'. That is the database record that we will change in this example.

Configuring the Link to this External Database

Setting up the link to this External Database is identical to how we did it for a previous example in Step 2 - Configure the Link to the External Database on page 96.

8.7.2 Step 2 - Initial Product Status and Job Setup

This is the starting configuration we need on Automation Engine:

• In Pilot > Products > Manage Product Statuses, create a Status 'Design arrived' with a blue color and create a status 'Waiting For Approval' with a yellow color.

• Create a Product 666-2712. Give it the status 'Design arrived' and link it to the Job 1237.

• The Job 1237 is for customer FruitCo.
8.7.3 Step 3 - Create the Workflow

In our example, we assume that the design was already preflighted and made ready for print. Our workflow starts by creating a proof report and e-mailing it to the customer. It then changes the Product status to Waiting For Approval and also updates the external database of this new status.

1. Create this workflow and save it as 'EXA - PRODSTAT':

2. In the Mark File step, set the tag 'Product file'.
3. In the Create Report step, we advise these logical settings:
   - choose any Report Template
   - choose to Export to an extra PDF (we will e-mail the report PDF so you do not want the PDF to refer to external files!)
   - choose a to Output the file somewhere in the Job Folder (because the report will probably contain Job specific info)
   - choose a logical File Name which will make sense to the customer when he receives it via e-mail.
4. In the **Send E-mail** step, we advise these logical settings:

5. In the **Select Mark File** step, use 'Product file' as the tag to be selected.

6. In the **Manage Product Status** step, select the **Status 'Waiting For Approval'**.

7. Save your workflow.
The **Interact with Database** step is described in the next step of this example.

### 8.7.4 Step 4 - Setup of the Interact with Database Step

These are settings we need:

![Edit Ticket of Interact with Database](image)

- **Output in** and **File Name**: because we are changing an existing database record, the type of **SQL** statement in this example is 'UPDATE'. As mentioned in the documentation of this task, this task only outputs an XML file for 'SELECT' statements.

- **External Database**: select the one you set up for the database in this example.

- **Statement**: use **Other**.

- **SQL**: use the query as shown in the screenshot. The 2 **SmartNames** make this query work for all **Product (Parts)**.

- Click **OK** to close.

- Save your workflow.

### 8.7.5 Step 5 - Test the Workflow

1. In **Pilot**, open the **Job** '1237'.
2. In **Products View**, select the linked in **Product** '666-2712'.
3. Launch the workflow 'EXA - PRODSTAT' that you created (using a **Favorite** or right-click **Launch with or...**).
4. This is the result in the Pilot; notice the Product’s change in status and color:

5. This is the resulting finished workflow:
6. This is how the external database should have changed; notice the new status 'Waiting For Approval' for item '666-2712':

<table>
<thead>
<tr>
<th>ItemNr</th>
<th>ItemName</th>
<th>ItemStatus</th>
<th>ItemCustoms</th>
<th>ItemColors</th>
</tr>
</thead>
<tbody>
<tr>
<td>666-9845</td>
<td>banana juice</td>
<td>In Stock</td>
<td>2712</td>
<td>5</td>
</tr>
<tr>
<td>666-2712</td>
<td>yolo Juice</td>
<td>Waiting For Approval</td>
<td>2712</td>
<td>5</td>
</tr>
<tr>
<td>667-8787</td>
<td>miky drink</td>
<td>In Prepress</td>
<td>6666</td>
<td>3</td>
</tr>
<tr>
<td>667-9951</td>
<td>spko loco</td>
<td>In prepress</td>
<td>6666</td>
<td>4</td>
</tr>
</tbody>
</table>