## Contents

1. **Concept** .................................................................................................................................................... 4

2. **General Principles** ......................................................................................................................................... 7
   2.1 Using a SmartName ........................................................................................................................................ 7
   2.2 The SmartNames View ................................................................................................................................. 9
   2.3 SmartName Types (Overview) ..................................................................................................................... 11
   2.4 SmartName Categories .................................................................................................................................. 13
   2.5 SmartName Scopes ......................................................................................................................................... 15
   2.6 SmartName Tags ............................................................................................................................................. 16
   2.7 Creating a SmartName (General) ................................................................................................................... 18
   2.8 Testing SmartNames ....................................................................................................................................... 19
      2.8.1 Also Show Workflow Parameter SmartNames ......................................................................................... 20
      2.8.2 Also Show Product (Part) Parameter SmartNames ................................................................................... 20
      2.8.3 Test your SmartNames with ‘Resolve All Using’ .................................................................................. 21
   2.9 SmartName Relations and Dependencies .................................................................................................... 23
   2.10 Exporting and Importing SmartNames ...................................................................................................... 26

3. **System (Value) SmartNames** ..................................................................................................................... 28
   3.1 Default System SmartNames ....................................................................................................................... 28
   3.2 Creating System Value SmartNames .......................................................................................................... 33
      3.2.1 Concept ................................................................................................................................................. 33
      3.2.2 Creating a System Value SmartName - Generic Steps ......................................................................... 33
      3.2.3 Example of Looking for the Job URL .................................................................................................... 34
      3.2.4 Example of Looking for the Product Part URL ..................................................................................... 34
      3.2.5 Example of Looking for the Product Part Data Zone URL ................................................................. 36

4. **String Extract SmartNames** ...................................................................................................................... 38
   4.1 Creating a String Extract SmartName ......................................................................................................... 38
      4.1.1 Start From Character ............................................................................................................................ 39
      4.1.2 Use Separation Character ..................................................................................................................... 40
      4.1.3 Use Regular Expression (link) ............................................................................................................. 41

5. **CSV Text Extract SmartNames** ................................................................................................................ 42
   5.1 Creating a CSV Text Extract SmartName .................................................................................................... 42
   5.2 Example ....................................................................................................................................................... 43

6. **XPath Query SmartNames** .......................................................................................................................... 45
   6.1 Creating an XPath Query SmartName ......................................................................................................... 45
   6.2 The XPath Builder ....................................................................................................................................... 46
      6.2.1 Building XPath Expressions ................................................................................................................ 47
      6.2.2 Editing Example XML Content ........................................................................................................... 50
1. Concept

What is a SmartName?

A SmartName is a variable that refers to a value. You use them by inserting them in input fields all across Automation Engine. For example the SmartName [File] refers to the name of the task’s input file, and the SmartName [Date] refers to the date the task is launched.

SmartNames are shown in green and between square brackets. For example [File].

Where can SmartNames be used?

Note: Not all SmartNames can be used everywhere. To learn about limiting the places where they can be used see SmartName Scopes on page 15.

- In Task Tickets, Workflow Steps and Access Points

  See an example of a ReportMaker ticket:

  ![ReportMaker Ticket Example](image)

  The output file will be written in the Job Folder, in a subfolder 'Reports'. The output file name will start with the name of the input file name plus the suffix '_reportcard_' and ending with the date when this task is launched.

  Tip: You here also see how you can make combinations of SmartNames and regular (static) text.

  The report template that will be used starts with the customer's ID number and ends with '_Report_Template'. This makes the ticket smart enough to select the template that this customer prefers.

- In some fields of the Job Setup

  In this example a SmartName defines the code of the bar code in the Job Setup (the code is a variable that will be coming in from an XML file):
• **As Workflow Parameters in Workflow Steps**
  When your workflow contains *Workflow Parameters*, you can pick them up as *Workflow Parameters SmartNames* in task tickets of tasks used in that workflow. Learn more about *Workflow Parameters* in *Workflow Parameters*.

• **Inside SmartMarks**
  See this example of how a *SmartMark* (here in *PackEdge*) uses a *Smart Text* that also picks up 2 SmartNames:
• **Inside the setup of other SmartNames**

  Some examples:
  
  • You can create a custom SmartName that extracts the first 5 characters from the *[Job Name]*.
  
  • In below example, a conditional SmartName 'EXA_Which_Site_Is_Closest' compares the value of two other SmartNames.

![SmartName setup example](image)

• **Inside some fields in the Customer setup**

• **In Configure, in some setup fields of Jobs and Products**

  **Advantages**

  SmartNames are a crucial part of Automation Engine’s power. They bring speed and error reduction to your workflow through these effects:

  • **Consistency.** Folders and file names will be consistent, no typing errors.
  
  • **Automation.** You will rarely need operator intervention to edit workflows.
  
  • **Integration.** SmartNames can pick up their value from other systems without human intervention.
  
  • **Organisation.** Making workflows smart means you need to prepare less workflows. Smart workflows adapt to the input files and the variable job instructions.

  When you first install or update Automation Engine, you already get many Default SmartNames. You can also make your own.

  In the next chapters, we first give an overview of the general principles. Then we describe the different types of SmartNames in more detail. We both explain the default ones and then show you how to create your own.
2. General Principles

2.1 Using a SmartName

There are 2 ways to use an existing SmartName:

- click on \[ \] to open the SmartName Browser. Choose a category, select the SmartName and click Insert.

- When you want to insert multiple SmartNames, press ALT while clicking Insert. Then, the SmartName Browser will not close.

- You can filter the shown SmartNames by entering text in the search filter top-right. Use double quotes (") when an exact match is required.

- You can even combine search filters. For example when you type '123 exa', you ask to see SmartNames that contain both 123 and exa in their name or Tag (learn about tags here). Add a semicolon '123 ; exa' to see SmartNames that contain 123 or exa.
• Click **Refresh** to make sure that the most recent created SmartNames are also shown (or press CRTL-R or CMD-R).

• Click **New SmartName** if you want to create a new one from here, while keeping this dialog open.

• right-click in the field, choose **Insert SmartName...** and choose a SmartName from the offered categories.

**Important:** In both ways, make sure you first position your cursor in the exact place where you want to insert the SmartName.

**Tip:** It is possible to select and cut, copy and paste SmartNames.
2.2 The SmartNames View

The SmartNames View in your Pilot is where you

- see all your SmartNames, both the default ones that came with Automation Engine and the ones you created extra
- add filters to help find and organize them
- create, modify or delete SmartNames (if you have the User Access Right. Read more in Defining User Access Rights)
- test your SmartNames ("resolve").

As usual, you can control the columns in this view. To do this, go to the menu View > Select Columns in 'SmartNames'... (by default they are all shown).

You can add a Description to your custom SmartNames. This quickly reminds you what they are for, without the need to open and analyse them.

Tip: The SmartNames View is not available when your Pilot is in Jobs Overview mode.

Tip: When you choose Tools > SmartNames, the SmartName View opens as a separate window.
Type - Category - Scope - Private - Tags (Overview)

We here briefly describe what these columns mean. Find more detailed descriptions on some of them in the next pages.

- **Type.** The type is a very important distinction. It defines what kind of technique the SmartName will use to get (resolve) its value. Will it get the value from a database inside Automation Engine? Or by asking the database of an external system? Or from an XML file or from another SmartName? All types are documented in *SmartName Types (Overview)* on page 11 and later in even more detail.

- **Category.** Categories are an old way of helping you filter and group SmartNames. We recommend using Tags instead (see further).

- **Scope.** By default the Scope of a SmartName is global. This means you can use it everywhere. Defining a different (smaller) Scope for a SmartName means that you want to limit the zone where it can be used. For example: if you define the Scope 'Step and Repeat' then you indicate that this SmartName makes no sense to be used outside that concept of Step and Repeat. More on this in *SmartName Scopes* on page 15.

- **Private.** You can save a SmartName as Private. Private SmartNames are not visible when applying SmartNames in a ticket. They do not appear in the list or the SmartName Browser. Two example use cases:
  - the SmartName is only used as a sub-component of one or more other SmartNames (that users will see and use)
  - you are still testing a new SmartName in the SmartName View and you do not want other users to already use it.

- **Tags.** Adding custom tags to a SmartName helps you filter and group them. Read more on this in *SmartName Tags* on page 16.

The right panel

On the right side of the SmartNames View, you can add more SmartNames to the list and you can test them extensively. Read more in *Testing SmartNames* on page 19.

Advanced Search Criteria for SmartNames

As in many other Views in the Pilot, you can use Advanced Search Criteria. To activate this, type in the Find field or press CMD-F (Mac) or CTRL-F (Windows). You see the criteria of the current filter. Click the + or - to add or remove extra lines of criteria.

This screen shot is an example of a filter that shows all SmartNames with exam in their Tag.

Click Save to save this filter. Define if this filter is for yourself or for all users. Decide if you want to Add to my views bar.
See an example of custom **Smart Views**:

### 2.3 SmartName Types (Overview)

Later in this document we describe each type in detail. Here is an introduction:

**Database Query SmartNames**

These retrieve a value from an external database. This is useful when there is no other way that the external system can send the information to Automation Engine (via XML or CSV or E-mail ....)
String Extract SmartNames

These SmartNames extract a part of a string of characters (text or numbers). For example: when the input is '123-abc', then extracting all characters before the '-' will result in '123'.

**Note:** The string from which the value of the SmartName is extracted can contain other SmartNames. For example: extracting the first 5 characters of the [JOB ID].

Read more on this type of SmartNames in *String Extract SmartNames* on page 38.

System Value SmartNames

These retrieve a value from one of Automation Engine's own databases. You get these when you install or update your Automation Engine server software. For example [Access Point E-mail Sender] (only valid in the context of an Access Point)

Read more on this type of SmartNames in *System (Value) SmartNames* on page 28.

Script SmartNames

These use JavaScript to define a value.

Read more on this type of SmartNames in *Script SmartNames* on page 64.

XMP Path Query SmartNames

These retrieve a value from the XMP (metadata) section of a PDF. For example when it was last modified or the PDF version number.

Read more on this type of SmartNames in *XMP path Query SmartNames* on page 51.

Xpath Query SmartNames

These use XPath Queries to retrieve a value from an XML. The *Xpath Builder* is a great help to construct these Xpath expressions.

Read more on this type of SmartNames in *Xpath Query SmartNames* on page 45.

Conditional SmartNames

These return a value based on a condition. For example a SmartName 'Trapping size' has these conditions: if the workflow parameter 'Printprocess' is 'flexo', then return '0,2' and if the workflow parameter is 'digital' then return '0'.

These SmartNames are also often used to route workflows in one or the other direction.

Read more on this type of SmartNames in *Conditional SmartNames* on page 54.

CSV Text Extract SmartNames

These extract a value from a CSV file. For example take the value from field 5 on line 2.

Read more on this type of SmartNames in *CSV Text Extract SmartNames* on page 42.
2.4 SmartName Categories

SmartNames are sorted into categories, so you can find them more easily. The SmartName View and the SmartName Browser show you these categories and their members:

![SmartName Categories Diagram]

**Date**

Use SmartNames of this category to insert the [Date] or [Time] of their execution.

**File Information**

These SmartNames return file-related information from within the file itself (for example [Media Box Width]) or from within the Automation Engine database ([File Type], [Folder of File], [Smart ID]...). The most popular one is [File], representing the name of the ticket’s input file.

**Global**

This is a full list of all the SmartNames that can be used anywhere throughout Automation Engine. The ones with a limited Scope will not be listed here.

**JDF**

These SmartNames are only used in tasks where JDF constructs are used. Read more about JDF in *What about JDF?*. 
Job Information
These SmartNames refer to most of the fields in a Job setup, including the details of Customer and CSR for that Job. Information on the location of the Job is part of the category Path Information. Any custom Job Parameters are also a separate category.

Job Parameters
When you are in a Job context of a Job that has custom Job Parameters, you will see them here. Their name automatically starts with 'jp. '

Miscellaneous
• Some SmartNames return the name of the Automation Engine server(s). They are useful when you use Assistant servers.

• [Auto-Number1] will add a sort-number to a group of output files. This helps sorting files. Find more information here.

Page Information
These SmartNames offer information on the (PDF) page attributes, typically height and width. You typically use these in the (page) imposition related tasks like the Export Imposition to PDF File.

Path Information
These SmartNames correspond to the Job’s location. Find more details in System (Value) SmartNames on page 28.

Tip: If these SmartNames are greyed out (unavailable), it means that the field you want to insert them into is not a path field.

Plate Information
These SmartNames offer information on the plate attribute of a (page) imposition. You typically use these in the task Export Imposition to PDF File.

Product Information
These SmartNames are only useful when you use the Products tool in Automation Engine. Much like the category Job Information, this category relates to the many fields in the Products database of Automation Engine. Read more about setting them up in Product (Part) Properties.

Task Information
This category of SmartNames allows to insert task related information (Task Name, Operator, Application Version...). Find more details in System (Value) SmartNames on page 28.

Workflow Context
These SmartNames pick up specific names or paths of folder or files in a workflow context. They help avoiding that you have to make complex workflows or create complex custom SmartNames to achieve the same result. Find more details in System (Value) SmartNames on page 28.
2.5 SmartName Scopes

Depending on where you are in Automation Engine, different SmartNames may be available for use.

Global SmartNames
These SmartNames are available everywhere where SmartNames are offered.

SmartMarks SmartNames
These SmartNames are only available when using SmartMarks. In Concept on page 4, find an example where a SmartName was used to decide the Smart Text in a SmartMark.

Step and Repeat SmartNames
These SmartNames are only available in Step and Repeat tasks.

Mark Information
These default SmartNames are only available in the task Add Marks to Imposition (page workflows only).

Stripping SmartNames
These default SmartNames are only available when retrieving values from JDF files (with the Import JDF Stripping task).

Imposition SmartNames
These default SmartNames are only available when working with impositions (for example, in the Export Imposition to PDF File ticket).

Unit Text Field SmartNames
SmartName enabled unit text fields are available in tasks where units have to be entered. You can do unit conversions and simple arithmetic using the operators +, -, *, and / . You can also use parenthesis () if necessary while calculating. PowerTrapper, Change Imposition Layout and Create CAD sheet and a few examples of tasks that offer such unit text fields.

Some examples:

```
[Width] inch
```

```
9 inch + 3 mm
```
2.6 SmartName Tags

Creating a Tag

The Scope of a SmartName defines where it can be used. The Category of a SmartName helps filtering. Adding Tags to a SmartName is an extra tool that enables you to organize them via custom filters.

Note: Tags are also shown in a separate column in the SmartNames View.

You can add one or more Tags when you save a new SmartName. When the Tag already exists, it will appear in light blue:

Tip: In the Tags field, press the space bar to get a drop-down list of all the Tags that were already created on your Automation Engine.

To add a Tag to an existing SmartName, select it, right-click and choose Info to edit and add one or more Tags.

To remove a Tag from a SmartName, select it and press the Delete button.

Tagging multiple SmartNames

1. In the SmartNames View, (filter and) select the SmartNames you want to tag.
2. Right-click and choose Tags... A dialog will show what Tags they already have in common: Tags that are not assigned to all selected SmartNames are shown in grey.

3. To add a Tag to all selected SmartNames, add the Tag and click OK.
4. To remove a Tag from all selected SmartNames, delete the Tag and click OK.
Filtering SmartNames on Tags

To filter SmartNames on Tags,

- in the **SmartNames View**, select the search field and type the name of a Tag.

**Tip:** It is also possible to add a search row by pressing the **Add** "+" button in the **Scope** bar, and choosing **Tags** from the attributes list (that pops up on the left).

- when selecting a SmartName from the SmartNames browser, enter the Tag in the search field.

To filter on **multiple** Tags, separate the names by a **space**. For example to filter on SmartNames that have both **examples** and **OLH** as Tag, type "examples OLH" into the search field.

![SmartNames View](image)

To filter on SmartNames that have **either** **examples** or **OLH** as a Tag, then separate the Tags by a **semicolon** ";" (see screen shot).

Renaming a Tag

To rename a Tag, follow these steps:

1. Go to the **SmartNames View**, filter on the **Tag**.
2. Select all SmartNames that contain that Tag.
3. Right-click and choose **Tags**....
4. Double click the Tag to edit it and type the new name.
5. Click **OK**.
2.7 Creating a SmartName (General)

These are the general steps to create a new SmartName. Find more detailed instructions per SmartName type further in this document.

1. In your Pilot, select **Views > SmartNames**.
2. Click on the **Create a new SmartName** icon in the toolbar. Alternately, choose **File > New SmartName**.

This opens the SmartName setup dialog:

![SmartName setup dialog](image)

**Note:** Click to select the attribute from a drop-down list.

3. Select the **Type**. We introduced the available types in *SmartName Types (Overview)* on page 11. See further in this document for a detailed setup instruction per type. Depending on which **Type** you choose, the dialog adapts to offer different options.
4. Select the **Scope**. We introduced the available scopes in *SmartName Scopes* on page 15.

**Note:** You can easily change the **Scope** later; this does not affect the setup options.

5. Set all the required attributes. This is a general introduction; see the specific settings per type further in this document.
6. Check the **Result** value (bottom left in the dialog).

![Result dialog](image)

Depending on the example input data (file), you can already here see the example result with the already chosen settings.

7. **Formatting Options**... You can still change the formatting of that result (text, number, data and time). This is explained in detail in *Formatting SmartNames* on page 66.

For example your SmartName already extracts a date out of an XML, but you also want to change the format of that date from 25-Aug-14 to 25-08-14.

8. Click **Save** and enter a logical **Name** and a suitable **Description**.
When you use **Save As...**, the dialog stays open. This is very useful when creating multiple variations of a SmartName one after the other.

*SmartName Tags* on page 16 explains all about adding Tags.

Decide if your SmartName should stay **Private**. Read more about Private SmartNames this in *The SmartNames View* on page 9.

9. Click **OK**.

You can use **Save As...** in the **Add SmartNames** dialog to enter a new name for the SmartName. This is useful when creating multiple variations of a SmartName one after the other from the same dialog.

10. Check the result of your new SmartName in more situations or tasks. Read more about this in *SmartName Relations and Dependencies* on page 23.

### 2.8 Testing SmartNames

While creating new SmartNames you will want to test them. Instead of having to test them in a often complex workflow, you can use tools available in the **SmartNames View**.

This part of the View contains 2 main sections:

- **Include SmartNames From:** This part serves to show extra SmartNames that were not appearing in your list yet. Choose if you also want to show **Workflow Parameters** and/or **Product (Part) Parameters**. Read more in *Also Show Workflow Parameter SmartNames* on page 20 and *Also
Show Product (Part) Parameter SmartNames on page 20. The reason that you also want to see them here is to test them as well. The next feature enables that:

- **Resolve all using:** This part allows you to test your SmartNames. You can test them on a task or on a file. Read more in *Test your SmartNames with 'Resolve All Using'* on page 21.

### 2.8.1 Also Show Workflow Parameter SmartNames

Follow these steps to show extra **Workflow Parameters** in your list of SmartNames:

1. In **Include SmartNames From**, check **A workflow** and browse to the workflow that has the **Workflow Parameters** you want to see here. They will immediately appear in your list of SmartNames (unless you have a filter setting that prevents that).

2. Tip: When your list shows many SmartNames already, it is possible you do not easily see them appear. Because they are here all shown starting with the prefix *wfp*, you can easily check their presence by typing a name filter ‘*wfp*’. Alternatively, you could also use the **Category** filter Workflow Parameters.

### 2.8.2 Also Show Product (Part) Parameter SmartNames

**Note:** This feature is only useful when your are using Automation Engine’s **Products** tool.

Follow these steps to show extra **Product (Part) Parameters** in your list of SmartNames:
1. **In Include SmartNames From**, check **A product part**: and browse to the product file that has the Product (Part) Parameters that you want to see here. They will immediately appear in your list of SmartNames (unless you have a filter setting that prevents that).

2. **Tip**: When your list shows many SmartNames already, it can be hard to find them back in the list. Because they are here all shown starting with the prefix `pp`, you can easily see them appear by typing a Name filter 'pp' or 'ppp'. Alternatively, you could also use the Category filter **Product (Part) Parameters**.

### 2.8.3 Test your SmartNames with 'Resolve All Using'

The **Resolve All Using** tool in the **SmartNames View** enables testing your SmartNames. The basic principle is that you give an example input which is then tested on all SmartNames.

**Tip**: If the SmartName you want to test is a **Workflow** or **Product (Part) Parameter**, then first make it appear in the list by using the feature **Include SmartNames from**. Read more on how to this in **Also Show Workflow Parameter SmartNames** on page 20 and **Also Show Product (Part) Parameter SmartNames** on page 20.
"Resolve all using:" - An example:

1. Make sure the SmartName you want to test is visible in the list.
2. If you want to test the SmartName by using it in a task, then first make sure you have a finished task given a certain input. Then check A task and browse to the task in your task monitor dialog. If you want to test the SmartName by using it on a file, check A file and browse to that file.
3. Click Show to start resolving all your SmartNames with the selected file or task. This also adds an extra column to your SmartNames View. The example in this screen shot shows how the selected file 666-2712_ReportCard.pdf made the SmartName [Everything before the _ ] resolved (resulted) in the 666-2712.

Set example values

This is useful for SmartNames that have dependencies. For the selected SmartName, you can type in a manual value instead of needing to select a file or a task as input example.

Tip: Not all SmartNames allow you to enter an example value. This will for example work for most System type SmartNames and for the CSV Text Extract type SmartNames. This is because the value of these SmartNames is a "given" and can be overruled by your new example value. The value of SmartNames that are "calculated" cannot be overruled; the resolving mechanism will simply calculate them when needed.

See an example in the below screen shot.

- You made a custom SmartName [File - only part before _], which is a string extract with the _ (underscore) as separating character.
• This SmartName of course a depends on the system SmartName [File]. If you then select the main SmartName [File], you will be able to set an example value.

• In our example, we enter the value "coco_765869".
  • See how this value is shown as a manually entered example value: the ♀ icon indicates this.
  • See how the SmartNames (that depend on the SmartName [File]), also react and resolve a value: the SmartName [File - only part before _] results in 'coco'.

Remove all Examples Values
Click this to remove all the example values from your SmartNames View.

Show results as Plain text
In some exceptional cases it can be necessary to insert a SmartName although the user interface does not offer to insert this SmartName in this place. Use this function to see the SmartName as plain text so you can then select, copy and paste it anyway in the dialog or ticket where you really want it. Test carefully, because these cases are really exceptional.

A 'plain text' version of a SmartName looks a bit like XML code. For example [File URL] is in plain text written as "<<fileurl/>>".

Reload Resolved Values
This 'refresh' function is useful when the input for your test has changed in the meantime.
For example you are setting up SmartNames that are reading parts of an XML file and you already did a test here. You typically want to test again after you know that the content of the XML file changed (your test will show new values).

2.9 SmartName Relations and Dependencies

In SmartNames View, when you right-click on one or on a selection of multiple SmartNames, you get these extra functions:
Tip: Most of those are also available from the View menu.

Attention: How SmartNames relate to each other is very important when you need to manage many of them. Such checks are very important especially when you plan to rename or delete a SmartName!

Highlight Related SmartNames

Choose this function to highlight all SmartNames that need one or more of the currently selected SmartNames: ‘Need’ can mean that the one you selected is based on the other(s) or it can mean that the other(s) are based on the one you selected. This highlighting is a feature that stays active until you switch it off again.

For example: see this screen shot of the ones related to the SmartName [File]:
Select Related SmartNames

Choose this function to select all SmartNames that need one or more of the currently selected SmartName. In this example [Fuel Cost] also gets selected because it [Fuel Cost no km] needs it (to extract the unit 'km').

<table>
<thead>
<tr>
<th>Name</th>
<th>Result</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel Cost</td>
<td>--</td>
<td>Script</td>
</tr>
<tr>
<td>Fuel Cost no km</td>
<td>--</td>
<td>String Extract</td>
</tr>
</tbody>
</table>

Select Dependencies

Choose this function to select all SmartNames that are used in the currently selected SmartNames. If you select the SmartName [Fuel Cost no km] in the example below and then activate Select Dependencies, the SmartName [Distance] will also get selected. This is because [Distance] is used in the definition of [Fuel Cost no km].

<table>
<thead>
<tr>
<th>Name</th>
<th>Result</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>--</td>
<td>System</td>
</tr>
<tr>
<td>Distance</td>
<td>--</td>
<td>XPath</td>
</tr>
<tr>
<td>EXA_SN_DISTANCE_Extract_No_Units</td>
<td>--</td>
<td>String Extract</td>
</tr>
<tr>
<td>EXA_SN_XPATH_Distance_Ind_Unit</td>
<td>--</td>
<td>XPath Query</td>
</tr>
<tr>
<td>EXA_WFP_Distance_From_A_No_Units</td>
<td>--</td>
<td>String Extract</td>
</tr>
<tr>
<td>EXA_WFP_Distance_From_B_No_Units</td>
<td>--</td>
<td>String Extract</td>
</tr>
<tr>
<td>Fuel Cost</td>
<td>--</td>
<td>Script</td>
</tr>
<tr>
<td>Fuel Cost no km</td>
<td>--</td>
<td>String Extract</td>
</tr>
<tr>
<td>IVWV distance to france</td>
<td>--</td>
<td>XPath</td>
</tr>
</tbody>
</table>

Display Selected SmartNames Only
This function is no longer about selecting but about limiting the SmartNames that are displayed. This function helps when the selected ones are spread far out over your long list of SmartNames. See how the limited focus is also shown at the top.

![SmartNames table]

When you want to see all your SmartNames again, then simply choose the function **Display All SmartNames**.

**Display Unused SmartNames**

When it’s time to cleanup your SmartNames you will want to know which ones are not being used (any more). Choose this function to display only the unused SmartNames. You will first get this warning dialog:

![Warning dialog]

Press continue to start the search. When done, decide what to do with them (delete? tag as ‘unused’?).

### 2.10 Exporting and Importing SmartNames

SmartNames can be exchanged by exporting and importing (workflow) tickets that use those SmartNames.

Learn more about exporting and importing (workflow) tickets in *Exchanging Workflows*.

**Caution:** SmartNames of *custom Job Parameters* ([jp.]) will not be imported. When you import a workflow that uses them, they will not be added into the job setup tab ‘Parameters’.

**Why Exchange SmartNames?**

Exchanging SmartNames is typically done
- with other people (during an Esko training or to send to Esko Customer Service).
• with other servers (colleagues that use another Automation Engine server in another plant of your company).

How to Recover a SmartName that was Deleted?

Your SmartNames are a standard part of the Automation Engine backup data. When you want to restore SmartNames that were (accidentally) deleted, you can get them back by restoring a backup of that configuration. The deleted SmartName of course has to be part of that backup.

Read more about backups and how to restore them in The Automation Engine Server Web Page.
3. System (Value) SmartNames

We will first list the default SmartNames of this type (System SmartNames), then show you how to create your own (System Value SmartNames) and end with some examples.

3.1 Default System SmartNames

We here describe the System type SmartNames that come with an install or update of your Automation Engine server.

Remember: In SmartNames View, the column Description already describes the SmartName briefly.

Note: Default System type SmartNames can not be changed. You can not double click and open them. You can not copy them.

- **Access Point Customer**, **Access Point E-mail Sender**, **Access Point E-mail Subject**, **Access Point Host**, **Access Point Name**: these can be used within Access Points.

- **Access Point Path**: the path to the folder on the FTP server that will be scanned.

- **Access Point Protocol**: describes the used File Transfer protocol in an FTP Access Point.

- **Application Name** and **Application Version**: mentions the name and or the version of your Automation Engine software.

- **Assembly**: Assembly method as defined in the JDF file (Assembly/@order), only used in page imposition tasks.

- **Auto-Number1**: the main reason to use this SmartName is to add control over sorting the files in the workflow. This SmartName is therefore only valid when the task has multiple input files. If you choose as output file name `[File]_[Auto-Number1]` then each output file will get an (increasing) number. For example: your task input files are banana, lemon and mango.pdf. Then the output file names will be banana_1.pdf, lemon_2.pdf and mango_3.pdf.

  Note: Notice that this SmartName is not related to version numbering!

- **Back**: the distance between the center of the back of the section and the trim box of the page (typically used to add collating marks to a page imposition).

- **Backing up**: the “backing up” method of the press (turn or tumble) as defined in the JDF file.

- **Book Name** and **Book Number**: name and number of the book (ImposeProof only).

- **CSR for the Job**: name of the customer service representative of the Job.

- **Customer’s Product Reference**: name that your customer gives to the Product item.

- **Customer’s Job Reference**: name that your customer gives to "his" job order.
• **Date**: the current date, formatted YYYYMMDD.

• **Device ID, Device Model Description, Device Model name, Device Model Number, Device Type**: as defined in the JDF file, used for page imposition "Stripping" tasks.

• **E-mail Address of the CSR for the Job**: the e-mail address of the customer service representative for the Job.

• **Extension of File**: the extension of the input file.

• **File**: the name of the input file without extension.

• **File Type**: the file type of the input file, for example PDF or PDFSC (= Esko Normalized PDF).

• **File URL**: the URL of the folder of the input file, the full path.

• **First Folder of File**: the name of the first folder of the input file. For example: the File URL is `file://aeserver01/ShareThatYouMadeContainer/FirstFolder/SecondFolder/banana.pdf` here, the First Folder of File is 'FirstFolder'.

• **Folder of File**: the name of the folder of the input file. In above example this is 'SecondFolder'.

• **Folder of First Input of Master Workflow**: name of the folder of the input file of the most outer workflow.

• **Folder of Input**: the name of the folder of the input file (same as **Folder of File**).

• **Folio**: the page name of the current page (relates to **Mark Information**).

• **Folios**: a range of page names (folios) containing prefixes and suffixes (for example ~A~,~C~--~E~,~G~).

• **Imposition Name**: the name of the imposition (page workflows).

• **Ink Name**: the name of the ink(s) (relates to **Plate Information**).

• **JDF Device ID** and **JDF Device Name**: the ID and name of the machine or software (RIP or workflow) that interprets JDF.

• **JDF Sheet Name** and **JDF Signature name**: the sheet and signature name in the JDF file.

• **Job Category**: the category of the Job as defined in the Job setup, maximum 7 categories.

• **Job Colors**: the numbers of colors in the job (file) as found in the JDF file.

• **Job Colors Mode**: the Job color mode ('Black&White' or 'Full') as found in the JDF file.

• **Job Container Name**: the name of the **Container** where the Job has its **Job Folder**.

• **Job Container URL**: the URL of the **Container** where the Job has its **Job Folder**.

• **Job CT URL**: the URL of the "ct" folder of the Job (= where image files can be created during normalization).

• **Job Customer Contact 1, 2, 3**: the name of the 1st, 2nd and 3rd contact of the **Customer** of the Job.

• **Job Customer Description**: the description of the **Customer** of the Job.

• **Job Customer E-mail 1, 2, 3**: the e-mail address of the 1st, 2nd and 3rd contact of the **Customer** of the Job.
• **Job Customer ID** and **Job Customer Name**: the ID and name of the **Customer** of the Job.

• **Job Customer Info 1, 2, 3, 4, 5**: the 1st etc., info of the **Customer** of the Job.

• **Job Description**: the description of the Job.

• **Job Due Date**: the due date of the Job (YYYYMMDD).

• **Job Due Time**: the due time of the Job (HHMMSS).

• **Job File Format**: the file format of the Job (standard format is Esko normalized PDF).

• **Job ID** and **Job Name**: the ID and name of the Job.

• **Job Temp URL**: the URL of the “temp” subfolder of the Job Folder.

• **Job URL**: the URL of the Job’s **Job Folder**.

• **Layer Name**: the name(s) of the layer(s).

• **Layer Names**: the names of the layers used in this sheet (page imposition workflows).

• **Layer Number**: the number(s) of the layer(s).

• **Local Server**: the name of the server where the task runs.

• **Mark Height** and **Mark Width**: height and width of the current mark.

• **Master Server**: name of the Automation Engine master server. This is the one that runs the “task manager”.

• **Media Box Height** and **Media Box Width**: the height and width (mm) of the Media Box of the first page of the input file.

• **Name (without extension) of originating Input of (enclosing) Workflow**: the name of the first input of the (enclosing) workflow.

• **Name of first Input of Master Workflow**: the name of the first input of the (most outer) workflow.

• **Name of Folder of Job**: the name of the **Job Folder**.

• **Name of Folder of originating Input of (enclosing) Workflow**: the name of the folder of the first input of the (enclosing) Workflow.

• **Name of originating Input of (enclosing) Workflow**: the name of the first input of the (enclosing) Workflow.

• **n-th Input File**: the index number of the input file (the how many’th it is).

• **Number of Pages**: the number of pages of the input file.

• **Number of Pages (RunList)**: the number of pages in the **RunList**.

• **Operator**: the Automation Engine user that launched the task.

• **Order ID**: the Order ID of the Job.

• **Padded Page Label**: the page label padded with zeros (extra zeros in front).

• **Padded Page Number**: the page number padded with zeros (extra zeros in front). Only used in the **Split Pages** task.
• **Page Label**: the label of the page. Only used in the **Split Pages** task.

• **Page Height** and **Page Width**: the vertical and horizontal trim size of the page.

• **Page Number**: the page number (starting from 1) or page range if the ‘One Ups’ contain more than one page.

• **Page Numbers**: a range of page numbers. For example: 1-5, 7, 9-11 .

• **Page Range**: the range of imposed pages available in the output file (**ImposeProof** only).

• **Paper Height** and **Paper Width**: the vertical and horizontal size of the paper (page workflows).

• **Plate Height** and **Plate Width**: the vertical and horizontal size of the plate (page workflows).

• **Plate Template Name**: the name of the plate template (page workflows).

• **Press**: the name of the press as defined in the JDF file (Device/@FriendlyName or Device/ @DeviceID).

• **Print Group Name**: the name of the print group (collection of sheets) or sheet. This is used in (Black & White) page workflows.

• **Product Customer Description**, **Product Customer ID** and **Product Customer Name**: the description, ID and name of the customer for that **Product**.

• **Product Custom Field 1, 2, 3**: the custom field 1, 2 and 3 of the **Product**.

• **Product Description**, **Product ID** and **Product Name**: the description, ID and name of the **Product**.

• **Product Ordered Copies**: the amount of ordered copies that were ordered of this **Product Part** in this Job.

• **Product Part Custom Field 1, 2, 3**: the custom field 1, 2 and 3 for the **Product Part**.

• **Product Part Data Zone**: the URL of the **Data Zone** of the **Product Part**.

• **Product Part Name**: the name of the **Product Part**.

• **Product Part Status**: the status of the **Product Part** (the Esko system name).

• **Project ID**: the Project ID of the Job.

• **RunList Name** and **RunList Number**: the name of the RunList and its number in the imposition.

• **Section Number**: the number of the section, starting from 1 (this is related to **Marks**).

• **Server**: the name of Automation Engine's (**Master**) Server.

• **Sheet Colors** and **Sheet Colors Mode**: the number of colors in the sheet and their color mode (‘Black&White’ or ‘Full’), as found in the JDF file (page workflows).

• **Sheet Descriptive Name**: a short description of the sheet as defined in the JDF file (StrippingParams/@DescriptiveName), used in page workflows.

• **Sheet Index**: the index of the sheet (starting from 0), used in page workflows.

• **Sheet Name** (scope ‘Stripping’): the name of the sheet as defined in the JDF file (StrippingParams/ @SheetName), used in page workflows.
- **Sheet Name** (scope 'Plate Information'): the name of the sheet, used in page workflows.
- **Sheet Number**: the number of the sheet (starting from 1), used in page workflows.
- **Sheet Side**: the side of the sheet: Front or Back (used in page workflows).
- **Sheet Side Smart ID**: the Smart ID of the sheet side (unique identification number).
- **Short Task ID** and **Short Workflow Task ID**: the short version of the task ID number and the (most outer) workflow task ID number. For example: '5439'.
- **Signature Name**: the name of the signature as defined in the JDF file (StrippingParams/@SignatureName), used in page workflows.
- **Smart ID**: the Smart ID of the file (unique identification number based on last modification dates of the file and all its references).
- **Stripping Part Number**: Stripping part number as defined in the JDF file, used in page workflows.
- **Sub Order ID**: the sub order ID of the Job.
- **System Defined**: this is used when outputting .imp files (page impositions). It is a combination of the imp name, sheet number, sheet side and layer name.
- **Task ID**: the ID of the task in Automation Engine, the long version. For example: 'afbdf6ac-8d82-4eee-af4d-b2408041640f'.
- **Task Name**: the Esko system name of the task (workflow) ticket. For example the **Add SmartMarks** task is called "batchbrix.smartmarks".
- **Task Owner E-mail**: the e-mail address of the owner of the task.
- **Ticket Name**: the name of the ticket of the task (the name you gave the ticket).
- **Time**: the current time (when the SmartName is calculated), formatted HHMMSS.
- **Trim Box Height** and **Trim Box Width**: Height and width of the Trim Box (mm) of the first page of the input file.
- **URL of first Input of Master Workflow**: the URL of the first input file of the (most outer) workflow.
- **URL of Folder of Input**: the URL of the folder of the input file.
- **URL of Folder of originating Input of (enclosing) workflow**: the URL of the folder of the first input of the (enclosing) workflow.
- **URL of Input**: the URL of the input file.
- **URL of originating input of (enclosing) workflow**: the URL of the first input of the (enclosing) workflow.
- **Variant Name** and **Variant Number**: the name(s) and number(s) of the variant(s) (used in page imposition workflows).
- **Web** (scope 'Imposition'): this is the index of the web in the print group (used in page workflows).
- **Web** (scope 'Plate Information'): this is the number of the web (for multi-web printing in page workflows).
- **Workflow Task ID**: the ID of the (most outer) workflow task.
3.2 Creating System Value SmartNames

As we introduced in SmartName Types (Overview) on page 11, a System SmartName is a query to one of Automation Engine’s own databases. Think of it as a question to Automation Engine.

As you can see in Default System SmartNames on page 28, there are already a lot Default System SmartNames. Still, there are still some cases where it makes sense to create your own extra System Value SmartNames.

3.2.1 Concept

When creating your own System Value SmartName, your question can be one of these 3:

- "Give me the Job URL, the full network path to the Job's Job Folder"
- "Give me the Product Part URL, the full network path to the Product Part file"
- "Give me the Product Part Data Zone URL, the full network path to the Product Part Data Zone"

These questions usually come up in these 2 cases:

- when you receive a file and you want to store it in a Job or Product related folder
- when you want to get a file from (under) those places.

In both cases, you need to add 1 or more parameters in your question that you do know. For example: "Where is the Job Folder if I tell you that the Order ID is 12345 and the Customer ID is 67888?"

Note: In a perfect world, you have a perfect data structure that is very logic and it never changes. Then you would maybe not have these questions. However, it can for example happen that your data Containers fill up and you may not be sure on which one the Job Folder was created. Or the files that you receive need more analysis to interpret where they belong to.

3.2.2 Creating a System Value SmartName - Generic Steps

We here only show the basic generic steps to create such a SmartName. See further for 3 specific examples.

1. In the SmartNames View, click on 
2. Choose the Type 'System Value'.
3. Choose a Scope. If you are not sure, choose Global. Learn more about SmartName Scopes on page 15.
4. Which System Value: choose the type you are you are looking for. A list of attributes will appear. These will help to find what you want.
5. Check the attribute(s) that you do know, one could be enough.
6. Enter the value of each checked attribute. This will probably also be a SmartName.
You obviously need to prepare these SmartName(s) first.

7. Check the **Result** value already shown below left. This will only be possible when you are in **SmartNames View** and are using the **Resolve all using** tool. Read more about these in *Test your SmartNames with 'Resolve All Using'* on page 21.

8. Click **Save as...** to save your SmartName.

### 3.2.3 Example of Looking for the Job URL

**Example: Finding the Job URL by using the name of an incoming design file**

- You have created a Job '1237' and you are waiting for the customers design file.
- You receive a design file via E-mail or FTP. Your customer does not know what Job name you already created. The name of the design file is '78999_Banana.pdf'. This name starts with how the customer names this job/design: the **Customer's Job Reference**, followed by a free name reflecting the design variant (here the banana flavor).
- The setup of your Job '1237' also includes this **Customer's Job reference '78999'** (this information came in earlier via your business system).
- You created a SmartName 'EXA FILE_ALLBEFORE_', a **String Extract SmartName** that extracts from a file name all characters before the '_'. Read more about such SmartNames in **String Extract SmartNames** on page 38.
- Your **System Value SmartName** SmartName now needs the settings as in this screen shot:

![Find Job via Cust Ref Nrs in Design File](image)

- **Save your SmartName with a logical name. For example 'Find Job via Cust Ref Nrs in Design File'.**

### 3.2.4 Example of Looking for the Product Part URL

**Example: Finding the full Product Part URL by asking the Product ID**

- Your business system sends you an XML with a list of product items on which you need to launch a workflow.
- Business systems often do not know the details of your prepress data structure. The XML mentions the **Product ID** but does not know the full file URL to that **Product Part**. So you need to find out that URL.
• For example, we look for the full URL of the Product ID '12345'. What the business system does not know is the folder and the name of the PDF. The PDF name is 'tomato_juice.pdf', it has no reference to the Product ID.

• You first need to create a SmartName that extracts the Product ID from the XML. See this example:

Learn more about extracting values from an XML file in *Xpath Query SmartNames* on page 45.

• Your System Value SmartName then uses that Xpath Query SmartName:
See (below left) how the **Result** shows the complete URL to the PDF. This is what we were looking for.

- Save your SmartName with a logical name. For example 'Find ProdPart URL via Prod ID in XML file'.

### 3.2.5 Example of Looking for the Product Part Data Zone URL

**Example: Finding the Product Part Data Zone URL by using the name of an incoming design file**

- You have a workflow where you business system creates new **Products** on your Automation Engine, even before the design file has arrived.
- One of those is the **Product** '6662712_Yolo_Juice' that is waiting for its design (and has no thumbnail display yet). The number before the underscore is the product ID.

- You have a **Folder Access Point** (or E-mail or FTP or WebCenter) that you scan for incoming designs.
- When a design comes in, you first want to copy it to the right destination: to its **Product Part Data Zone**.
• You need a **System Value SmartName** as in this screen shot:

![Image of System Value SmartName](image1.png)

• Save your SmartName with a logical name. For example 'Find ProdPart DataZone URL via ProdID in Design File'.

• Your **Copy or Move File** ticket now uses this SmartName:

![Image of Copy or Move File](image2.png)

• You probably want to continue your workflow with the **Manage Product Status** task to change the status of this **Product Part** to 'Design File Arrived'.
4. String Extract SmartNames

When automating your workflow, you will quickly feel the need to extract a part of a file name or a network path. **String Extract SmartNames** offer to do just this.

**Note:** A **String** is a series of characters. These can be numbers, text or even symbols.

4.1 Creating a String Extract SmartName

To create a **String Extract SmartName**, follow these steps:

1. In **SmartNames View**, click on ![SmartNames Icon].
2. Choose the **Type** ‘**String Extract**’.
3. Choose a **Scope**. If you are not sure, choose **Global**. Learn more about **SmartName Scopes** on page 15.
4. In **Extract part of...**, decide from what you will extract something. In the canvas, you can
   - type. This is mostly done to test your SmartName while creating it (you see the result below left)
   - insert **SmartNames**. This is the most typical use, for example to extract parts of a [File] or a [File URL].
   - right-click and choose an option: **Cut**, **Copy**, **Paste**, **Clear Contents** or **Insert SmartName**.
5. Define what you want to extract. Choose from these 3 methods:

- **Start from character.** Learn more in *Start From Character* on page 39.

- **Use separation character.** Learn more in *Use Separation Character* on page 40.

- **Use regular expression.** A "Regular Expression" is a standard language in computer science. It enables very specific ways to extract parts of a string. Learn more in *Using Regular Expressions* on page 73.

  **Caution:** Results can vary if your input string is considered as a text or as a number! This is decided in the *Formatting Options*. The default is that your string is assumed to be text. Learn more about this in *Formatting SmartNames* on page 66.

**Note:** The Result will update itself only after you changed a setting or clicked somewhere in a different field. Tip: press the 'Tab' button to make sure that the Result is re-calculated.

6. Click **Save** to save your SmartName and close the dialog. Click **Save As...** to save your SmartName but keep the dialog open (to create more SmartNames).

### 4.1.1 Start From Character

This method allows to decide the start and end character that you extract. Some examples:
Example 1: "First 5"
Your input string is '874095620banana' and you want to extract the first 5 characters.

Choose Start from character. For Start from Character choose '1' and for Number of Characters choose '5'. This will result in '87409'.

Note: This method is mostly used to extract the first characters of a file name. To extract the last part of a file name, you need to use Regular Expressions. Learn more in Using Regular Expressions on page 73.

Example 2: "All except the first 3"
Your input string is '8740956202537435'. You want to extract all of the characters except the first 3.

Choose Start from character. For Start from Character choose '4' and for Number of Characters leave the default setting to '999'. This will result in '0956202537435'.

4.1.2 Use Separation Character

This method allows to separate the string into parts every time a certain Separation Character occurs, and extract one of the parts.

You can choose a Separation Character from the list but you can also type in a different one.

The String Part is where you choose which part of the string you want to extract.

Example 1: "Everything before the (first) underscore"
Your input string is '12345_678901' and you want to extract everything before the underscore '_'.

Choose Separation Character '_ ' and set the String Part to '1'.

This will result in '12345'.

Example 2: "The 3rd part"
Your input string is '87409_5620_27923' and you want to extract the number '27' which is the 3rd part of the string (if you separate it with the '_').

Choose Separation Character '_ ' and set the String Part to '3'.

This will result in '27'.
Note: If you want the part after the last '_', then this is not possible using the **Use separation character**. To do that, you need to use the method 'Use regular expression'. Learn more in *Using Regular Expressions* on page 73.

Example 3: "Everything after the '@'"

Your input string is 'Franky.Fruit@FruitCo.com' and you want to extract everything after the '@'.

In **Separation Character**, type in '@' and set the **String Part** to '2'.

This will result in 'FruitCo.com'.

### 4.1.3 Use Regular Expression (link)

Regular expressions are not only available in **String Extract SmartNames**. There are documented in *Using Regular Expressions* on page 73.
5. CSV Text Extract SmartNames

A CSV Text Extract SmartName allows you to create a SmartName that finds a value in a specific field of a CSV file. We will first illustrate the generic steps to create such a SmartName and then show a concrete example.

Note: Although the name of this SmartName type contains the word 'text', the value that is extracted can also be a number or any other character type.

5.1 Creating a CSV Text Extract SmartName

To create a CSV Text Extract SmartName, follow these steps:

1. In SmartNames View, click on ![SmartName Icon].
2. Choose the Type 'CSV Text Extract'.
3. Choose a Scope. If you are not sure, choose Global. Learn more about SmartName Scopes on page 15.
4. Define where your CSV can be found and set some options:
   - **CSV Source**: Define the path and name of the CSV file. You typically use SmartNames here.
   - **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. If you're not sure, open your CSV in a simple text editor.
   - **Quote Character**: Indicate if the value in the CSV contains quotes that you do not want to keep. For example: in the CSV a 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. Use Select Field... to preview the effect of this.
5. Define which specific field in the CSV you want to extract.
   - Type the number of the field and line in Extract text from field ... on line ....
   - or click on Select Field... to get a preview of your CSV file and there click the field you want to extract. The field and line number will then be entered automatically.

   Note: A preview of your CSV file will only appear if the field CSV Source can already find your CSV input file.
6. Check the result in Result.
7. If needed, set any Formatting Options. Learn more in Formatting SmartNames on page 66.
8. Click **Save** to save your SmartName and close the dialog. Click **Save As...** to save your SmartName but keep the dialog open (to create more SmartNames).

5.2 Example

- You are working with the **Products** tool and have set up some integration with your business system. You have set up a workflow where your **Jobs** are created automatically.
- You also get (from another system) a CSV file with the ordered copies (amount) that a **Product** is to be produced in that Job context. The CSV is written in the **Job Folder**, in a subfolder **CSV** and with a specific name. The screen shot shows how the **CSV Text Extract SmartName** can be set up:

This SmartName [Find Amount in local CSV] can now be used in your automated workflow where you use the **Link Product to Job** task.
6. Xpath Query SmartNames

As mentioned in SmartName Types (Overview) on page 11, Xpath Query SmartNames use "XPath queries" to extract a value from an XML file. XPath, the XML Path Language, is a standard query language for selecting nodes from an XML.

In Automation Engine, there are many tasks that can read parts of an XML, often without the need to create a SmartName first.

Tip: Many of these tasks and tools are described in the chapter Integrating with External Systems.

To help you build such XPath queries, Esko offers the The XPath Builder on page 46. This tool is available throughout Automation Engine, wherever you can create XPath queries.

6.1 Creating an Xpath Query SmartName

To create an Xpath Query SmartName, follow these steps:

1. In SmartNames View, click on [ ]
2. Choose the Type 'Xpath Query'.
3. Choose a Scope. If you are not sure, choose Global. Learn more about SmartName Scopes on page 15.
4. The large canvas is where you define your Xpath Expression. Initially, there is an example expression about 'vsize'. It will be removed automatically when you have used the Edit button and created your own. In this canvas, you can
   - type an Xpath Expression (if you know how to do this).
   - click on Edit to open the XPath Builder and create your expression there. Learn more in The XPath Builder on page 46.
   - click on Insert SmartName to insert SmartName(s) into your Xpath Expression. This is in most cases not needed.
   - right-click and choose an option: Cut, Copy, Paste, Clear Contents or Insert SmartName.
5. **Separator**: If the *XPath Expression* returns multiple elements, decide what character you want to use as separator.

6. **XML Source**: Define the path and name of the XML. Or keep the default SmartName [URL of Input].

7. **Result**: See the result of your *XPath Expression*. This is possible when the XML Source can be found. The Result is also visible when you are in SmartNames View and are using the Resolve all using tool and have manually selected an example XML. Read more about this in *Test your SmartNames with 'Resolve All Using'* on page 21.

8. **Formatting Options**: Learn more in *Formatting SmartNames* on page 66.

9. Click **Save** to save your SmartName and close the dialog. Click **Save As...** to save your SmartName but keep the dialog open (to create more SmartNames).

### 6.2 The XPath Builder

#### Introduction

*XPath* is a language that describes how to locate elements and attributes in XML (Extensible Markup Language) files. You can find a full specification on [http://www.w3.org/TR/xpath/](http://www.w3.org/TR/xpath/) and a tutorial on [http://www.w3schools.com/xpath/](http://www.w3schools.com/xpath/).

Think of an *Xpath Query SmartName* as a *String Extract SmartName* but then to get a string from an XML file.

The *XPath Builder* is an XML editor available in the Automation Engine Pilot. Anywhere in the Automation Engine where there is an *XPath Expression*, you can access the *XPath Builder* by clicking **Edit**.

The *XPath Builder* panel
- On top of the panel, you see the resulting XPath Expression that is probably a result of what you have clicked below.
- The large canvas shows the Example XML Content.
  - When possible, the example XML is automatically taken from the context. For example when you open the XPath builder from the XPath Query SmartName dialog, then the input file is used as an example XML.
  - When no example XML is available automatically, right-click in the canvas to see these extra ways to get and manage XML content: Cut, Copy, Paste, Clear and Paste from File.
  - Use CTRL-F (Windows) or CMD-F (Mac) to open a Find tool. This is very handy when your XML is a large file and you quickly want to find the string you want to select.

Note: This area has XML syntax highlighting: specific colors indicate a specific kind of node.
- On the right you have tools that help you create the XPath Expression by offering smart selections and even help you edit the XML content.

6.2.1 Building XPath Expressions

Select XML node
To build an XPath Expression, make sure that the Select XML Node button is clicked.

Simply click on an XML node to automatically create an XPath to that node. The resulting XPath is shown in the XPath Expression field at the top.
Note:

- The created XPath will be an "absolute XPath" from the root node to the selected node.
- The selected node will be highlighted in yellow.
- When the selected node is used in a list, an index number will be used in the XPath. For example: `/PurchaseOrders/PurchaseOrder[1]/Address[2]/Street/text()`.
- The generated XPath takes namespaces into account by using the local-name function. For example: `/*[local-name()='PurchaseOrder']*[local-name()='PurchaseOrderNumber'='99503']/*[local-name()='Address']*[local-name()='Type'='Billing']/*[local-name()='Street']/text()`.

Note: By selecting Enforce namespaces, the XPath Builder will verify any used namespace.

Add Condition

To add certain conditions to the XPath, click Add Condition. Then click on the desired XML node to add it as a 'predicate' to the XPath Expression.

Note:

- You can also add a condition by clicking on the desired XML node while holding the Alt key.
- The predicate nodes are highlighted in green.
- When using predicate nodes, the indices in the XPath are replaced with predicates. For example: `/PurchaseOrders/PurchaseOrder[@PurchaseOrderNumber='99503']/Address[@Type='Billing']/Street/text()`.
- When using predicates, it is possible that the XPath Expression selects multiple XML nodes. The number of matches is displayed beneath the XPath Expression.
Options

- **Select siblings.** Select this option to select all occurrences of the selected element at this level.

  **Tip:** This option is especially relevant when working with ‘repeating content’ in the *Map data task*. You there need to **Select siblings** to avoid that only the first element would be repeated (you would so get a list of only 1).

- **Enforce namespaces.** Select this option to enforce XML namespaces (‘xmlns’).

  **Tip:** XML namespaces provide a method to avoid element name conflicts. In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML files from different XML applications.

- **Use prefixes.** Check this option if you want to use XML namespaces in your Xpath Expression in a shorter way.

  **Note:** This option overrules the option **Enforce namespaces**.
Tip: This option is especially useful when querying XMP.

6.2.2 Editing Example XML Content

Format XML

Under Example XML Content, click on Format XML to format the example XML. This can be used to make the XML more readable. It will typically remove unnecessary spaces and lines.

Edit XML

To edit the example XML, follow these steps:

1. Under Example XML Content, click on Edit XML.
   
   Note: This will make the XML content editable. Also, the buttons Revert and Apply will become visible.

2. Edit the XML in any way you want.

3. When finished making changes, click Apply.
   
   Note: The changes will be automatically validated. If the XML is not valid, an error message will appear and the location where the parse error occurred will be visible and highlighted in a red color.

4. Before you click Apply, you can click on Revert to undo all changes made to the XML content.
   
   Note: You can undo/redo the last edits using (on Windows) Ctrl-Z and Ctrl-shift-Z or (on Mac) Command-Z and Command-Shift-Z.

   Note: If you clicked Apply but did this in error, you can still click Cancel.

Note: It is also possible to edit the XPath Expression itself manually. When you do this, take the following into account:

- After editing the XPath, press Enter to validate the updated XPath Expression.

- If the updated XPath Expression is not valid, an error message will be displayed beneath the XPath Expression.

- The corresponding XML nodes will be highlighted in grey.
7. XMP path Query SmartNames

XMP stands for ‘eXtensible Metadata Platform’. It allows embedding data about a file, known as metadata, into the file itself. It is a standard defined by Adobe.

Esko also uses XMP as format for metadata in its processed files (Normalized PDF, View data, LEN and other RIP output).

**Note:** A full specification of how Esko uses XMP is available as part of the Automation Engine Online Help. Find the white paper in its section “Related Documentation”.

Because XMP is actually XML embedded in a file, you can also use SmartNames to query that metadata. **XMP path Query** SmartNames are very similar to **Xpath Query** SmartNames. The main difference is that the source file is not a XML but a file with embedded XMP. The way to that metadata is also an **Xpath Expression**.

### 7.1 Creating an XMP path Query SmartName

When creating an **XMP path Query SmartName**, it is typical is that we first need to indicate our example input file, the one that contains the example XMP.

To create an **XMP path Query SmartName**, follow these steps:

1. In SmartNames View, in Resolve all using, check A file and browse to your (example) file that has the XMP metadata.
2. In SmartNames View, click on ![Resolve All Using](https://example.com).
3. Choose the Type ‘XMP path Query’.
4. Choose a Scope. If you are not sure, choose Global. Learn more about **SmartName Scopes** on page 15.
5. The large canvas is where you define your **Xpath Expression**. Initially, there is an example expression about `vsize`. If `vsize` is mentioned in the XMP of the input file you selected, it will already return a Result value (below left). This example expression will be removed automatically when you have used the Edit button and created your own. In this canvas, you can
   - type an Xpath Expression (if you know how to do this).
   - click on Edit to open the XPath Builder and create your expression there. Learn more in *The XPath Builder* on page 46.
   - click on Insert SmartName to insert SmartName(s) into your Xpath Expression. This is in most cases not needed.
   - right-click and choose an option: Cut, Copy, Paste, Clear Contents or Insert SmartName.
6. **Separator:** If the **XPath Expression** returns multiple elements, decide what character you want to use as separator.
7. **Result:** See the result of your **XPath Expression**.
8. **Formatting Options:** Learn more in *Formatting SmartNames* on page 66.
9. Click **Save** to save your SmartName and close the dialog. Click **Save As...** to save your SmartName but keep the dialog open (to create more SmartNames).

### 7.2 Examples

**Remember the many System SmartNames on 'Size'**

There are many SmartNames that you no longer need to create yourself via XMP queries. Remember these default (system type) SmartNames with **Scope Page Information** that may already deliver exactly what you were looking for:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Scope Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folio</td>
<td>A range of page names (folios) containing prefixes and suffix...</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Layer Name</td>
<td>The name(s) of the layer(s)</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Layer Number</td>
<td>The number(s) of the layer(s)</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Number of Pages</td>
<td>Number of pages in the RunList</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Page Height</td>
<td>Vertical trim size of the page</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Page Numbers</td>
<td>A range of page numbers (numbers are going from 1, for 0...</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Page Width</td>
<td>Horizontal trim size of the page</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>RunList Name</td>
<td>Name of the RunList</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>RunList Number</td>
<td>Number of the RunList in the imposition</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Variant Name</td>
<td>The name(s) of the variant(s)</td>
<td>System</td>
<td>Page Information</td>
</tr>
<tr>
<td>Variant Number</td>
<td>The number(s) of the variant(s)</td>
<td>System</td>
<td>Page Information</td>
</tr>
</tbody>
</table>

Learn more about them in *Default System SmartNames* on page 28.

**What if my file does not have the XMP information I'm looking for?**

For example: you want to get the 'vsize' of a native PDF. Your workflow does not create a normalized PDF version of it (where you would automatically get that extra XMP). In that case, you can use the **LinkEdge** task to create an XML containing the same XMP as you would find inside a normalized file.

The next step is then to use an **Xpath Query SmartName** to get that specific XMP information from inside that XML file. Learn more on how to do this in *Xpath Query SmartNames* on page 45.

**Which application created this file?**

Most files with metadata will have a note about which application created the file. Follow these steps:

1. Follow the generic steps as explained in *Creating an XMP path Query SmartName* on page 51.
2. In the example XML, use CTRL-F or CMD-F to find for the string 'CreatorTool'.
3. Select the node. For example: see how the 'CreatorTool' is selected in the example below:
4. Click OK.
5. Check the Result and Save your SmartName.
6. Tip: to shorten this name, create another String Extract SmartName or a Regular Expression.

Which options were used when RIP'ing this file?

It can be useful to see with what resolution or distortion your RIP file was made. There are tools to quickly detect that in Automation Engine, but your remote print site may not have those tools. It can therefore be useful to extract that value and use it in a file name or (FTP) folder name.

Follow the similar steps as in above example. See the screen shot to detect even more RIP options in the XMP of a LEN file (made with FlexRip or Imaging Engine).
8. Conditional SmartNames

Conditional SmartNames base their value on one or more conditions. The condition can be very basic or can be built up of several criteria and sub criteria. You can also define the order in which the conditions need to be met.

After documenting how to create a Conditional SmartName, we illustrate their power with some examples.

8.1 Creating a Conditional SmartName (generic)

To create a Conditional SmartName, follow these steps:

1. In SmartNames View, click on 
2. Choose the Type 'Conditional'.
3. Choose a Scope. If you are not sure, choose Global. Learn more about SmartName Scopes on page 15.
4. The dialog initially contains 2 blocks.
   - A "conditional block". Each conditional block has one or more criteria and ends with the Return value for that block.
     - You can create extra conditional blocks using the buttons Add / Remove (Windows) or + - (Mac).
     - You can also right-click in the line If xxx of the following are true: and Duplicate that block. Use the functions Move Up / Down or drag and drop the blocks to define their order.
     - The order of the blocks is the order that they will be checked, from top to bottom.
• Read more detail in Setting up a Conditional Block on page 55.

• An "alternative block". When none of the above conditions match, this is what the SmartName's return value will be. Type in the value (or use another SmartName).

5. Check the result in Result.
6. If needed, set any Formatting Options. Learn more in Formatting SmartNames on page 66.
7. Click Save to save your SmartName and close the dialog. Click Save As... to save your SmartName but keep the dialog open (to create more SmartNames).

8.2 Setting up a Conditional Block

To set up a conditional block, follow these steps:

• If All - Any - None of the following are true. Choose the relation between the condition and your (one or more) criteria.

• Per criterion, you need to
  • select the attribute from the drop-down. Choose between the SmartName File or any Other SmartName.
  • select the text or number expression from the drop-down list. You can choose any of the following expressions from the drop-down, depending on whether it is a text or numeric condition:
• For a **text** criterion, choose from the following expressions:

  - **Contains, begins with, ends with, is and is empty.**
    Use ' ' (space) for 'AND' and ';' for 'OR'. For example, a text criterion specifying:
    - "'Inkname' contains 'Pantone C'": this will resolve all ink names that contain 'Pantone' and all that contain 'C'.
    - "'Inkname' is 'C; M; Y; K;'": this will resolve the ink names that are specifically any of these.
    - Use double quotes to specify the exact text corresponding to the attribute. For example: "Inkname contains "reflex blue"": this will only resolve for "reflex blue".

• For a **number** criterion, choose from the following expressions: **equals, is greater than, is less than, and is between.** Some important notes:

  - If the attribute does not correspond to a number, it will be ignored.
  - You can use standard unit symbols and unit conversions are done in background. For example: if you specify "equals 5 mm" then this is not a problem when the file's XMP is defined in in (inches).

    **Note:** Unit conversions only work with units of the same kind. For example if the specification is "equals 5 mm" then this will fail to resolve when the XMP mentions "6 ppi" or just "6".

• **Note:** Number expressions will not resolve to text strings. For example: "Job Name equals MyJobName" will be ignored because equals is used for numbers, not text.
8.3 Examples

**Plate distortion depending on cylinder size**

A Job Category is used to define size of the cylinder on which the product will be printed. This circumference has an effect on the distortion that is required for the plate. A Conditional SmartName defines the distortion setting for the RIP.

![Distortion based on Cylinder Size](image)

**ReportMaker Template depends on the Customer**

The ReportMaker ticket uses a SmartName to select its template. A conditional SmartName checks who the customer is.
First we check if it is our biggest customer 'FruitCo' (ID 777). Then we check if it is any customer of the group 'SuperToy' (any of those 3 IDs). If it is none of the above, we use a generic template for all other customers:

Dot Gain Compensation depending on the Press and type of flexo plates

If the press chosen for the Job is the flexo press Nilpeter, then the DGC file depends on the category where the type of plates is defined: conventional flexo plates or digital flexo plates (CDI):

Find the French only version (using a sub criterion to exclude some attributes)

Our SmartName is looking for Jobs that are in the French language, but not in combined language versions. See how the extra sub criterion (ALT+‘+’) excludes the categories that contain the Canadian and Belgian version Jobs that combine French with another language:
Smart preflight checking a workflow parameter 'Print Process' to check allowed line width

Our workflow includes PitStop preflight. It is Smart Preflight because it uses SmartNames. Our example shows how the preflight profile will check line width but have it depend on the print process that the file will be printed in. The print process is defined as a workflow parameter.

- This is our workflow parameter: [wfp.Print Process]

- This is how we set up our conditional SmartName (3 print processes each get a condition on line width (mm)): 
• This is how our SmartName is used inside the PitStop Profile (there named 'variable'):
# 9. Database Query SmartNames

## 9.1 Concept

Database Query SmartNames ask a question to an external database. They pick up information from other systems, typically business systems (planning, production...). They only ask information, they do not write information into that external system.

**Note:** Database Query SmartNames may not be used to pick up information from a database from Automation Engine itself. For such use cases, you use the System (Value) SmartNames. Learn more in System (Value) SmartNames on page 28.

Some typical examples when you use Database Query SmartNames:

- In the Tabular Step & Repeat task. You can fill its many fields with Database Query SmartNames and so pick up all the required values. Narrow-web label printers often use this and base their queries on press type and die-shape.

- As a RIP option that you pick up from an external system. For example your business system that created your Job may have informed you about main production decisions like the press, the amount of plates and their size. But the distortion factor may need to be picked up from another external system closer to the press room. In that case, you use a Database Query SmartName in your RIP’s setting for distortion.

How do Database Query SmartNames compare to the other tools that access databases?

- A Database Access Point also does a database query but its other main feature is that it triggers a workflow, and maybe you do not want that to happen at the moment you pick the information.

- The Interact with Database task can read and write into external databases and can be used as any step of a workflow. But it does not return you the values immediately, you get them in a separate XML file from where you need to pick them up again (then via XPath Query SmartNames).

**Note:** In cases where that XML is already available, it is advised to use the information from the XML. For example: the XML that was used to create the Job, is stored in the Job Folder and it also contains all the Step & Repeat information. In that case it is advised to pick up that Step & Repeat info locally instead of asking it again to an external system (which always means network traffic and small delays).

Basically, Database Query SmartNames is the preferred tool when you only want to pick up a value and when an extra return-XML has no added value.
9.2 Creating a Database Query SmartName

First set up the link to that external database

Setting up a link to an external database is done in **Tools > Configure > External Databases**. Learn more on this in *Configuring links to External Databases*.

Now create the Database Query SmartName

To create a **Database Query SmartName**, follow these steps:

1. In **SmartNames View**, click on ![ ]
2. Choose the **Type** 'Database Query'.
3. Choose a **Scope**. If you are not sure, choose **Global**. Learn more in *SmartName Scopes* on page 15.
4. Choose the **External Database** from the drop-down list. If you need to create, change or test the link to an external database, click **Configure**.
5. Write your **SQL Command** in the canvas. You can **Insert SmartNames** into your command.

**Note:** Esko does not provide training in SQL commands. Contact your own IT database specialist to help set these up.

```
<table>
<thead>
<tr>
<th>SQL Command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT Press FROM MyOrders WHERE OrderID = [Order ID]</td>
</tr>
</tbody>
</table>
```

6. Check the **Result**. The result will be (re-)calculated when you click any option outside the canvas.
7. If needed, set any **Formatting Options**. Learn more in *Formatting SmartNames* on page 66.
8. Click **Save** to save your SmartName and close the dialog. Click **Save As...** to save your SmartName but keep the dialog open (to create more SmartNames).

**Learn more about integrating with external databases**

Learn more about using SQL commands in the chapter *Integrating with External Systems*, in the sections

- *Database Access Point*
• *Interact with Database task*

• *An elaborate example.*
10. Script SmartNames

Script SmartNames use JavaScript to get a value. Typically, other SmartNames are inserted into the script.


**Attention:** Before you create a Script SmartName, check if it can be built using standard tools as
- a Conditional SmartName
- a Regular Expression (in a String Extract SmartName).

10.1 Creating a Script SmartName

To create a Script SmartName, follow these steps:

1. In SmartNames View, click on ![SmartNames](image)
2. Choose the Type ‘Script’.
3. Choose a Scope. If you are not sure, choose Global. Learn more in SmartName Scopes on page 15.
4. The canvas initially shows a JavaScript template that returns the value "hello".

**Attention:** This template is important. The script needs to be a function returning a value that is called from the body of the script. Learn more in this Esko KB article.

Replace it with your own JavaScript. You will probably Insert SmartNames into your script. See some examples in Examples on page 65.
5. Check the result in Result. The result will be (re-)calculated when you click any option outside the canvas.

6. If needed, set any Formatting Options. Learn more in Formatting SmartNames on page 66.

7. Click Save to save your SmartName and close the dialog. Click Save As... to save your SmartName but keep the dialog open (to create more SmartNames).

10.2 Examples

The following examples are just a few basic samples of what a JavaScript can do. If you need help, Esko has specialists available through Esko Solution Services (Not Esko Customer Service).

Calculating: Multiply

This JavaScript multiplies the values of 3 Job Parameters (number SmartNames):

Extracting a few lines from a text:

This JavaScript returns the first 2 lines of a SmartName, in this case the Job Description:
11. Formatting SmartNames

A lot of data that end up in SmartNames can come from external systems. And a lot of information that you send back out to external systems may also be decided by SmartNames. These systems can store their information in a different language, in a different regional format or even a specific custom format. That is why it may be needed to change the format of a SmartName.

The formatting options are different whether your data is a Text, a Number or a Date or Time.

**Note:** A data string that contains both numbers and text characters is seen as Text. For example: 789_87_banana is Text.

The option Formatting Options... is available when you create any SmartName.

11.1 Text Formatting

When, in Formatting Options..., you select the Data Format 'Text', you get the following options:

- **Remove leading and trailing spaces:** If your SmartName value includes space characters at the beginning or end, use this option to remove those spaces.

- **Text Case:**
  - **Keep** the original case of the SmartName value.
  - change the text case using Make Upper Case or Make Lower Case.

- **Specify any special Encoding** of the SmartName value. Choose
  - **None** to leave it like it is.
• **XML Encoding.** If your SmartName output will be used inside an XML file, your value can contain a character that has a special meaning in XML (a 'reserved character'), then this encoding will replace that character by a by an XML entity reference. For example ‘10<20’ will be formatted to ‘10&amp;lt;20’.

• **URL Encoding.** Similar to the above, this will replace the reserved characters for URLs with the typical ‘percent encoding’. For example ‘Digital Flexo’ will be formatted to ‘Digital%20Flexo’.

• **Custom Encoding:** Use it to define custom replacements of characters or strings. Click **Edit** and add one or more lines of rules. The rules will be executed from top to bottom. For example:

  - First all ' ' (spaces) will be replaced by a Dollar sign
  - then all '@' will be replaced by 'at'
  - then all words 'Limited' will be replaced by 'Ltd.'

The **Result** will update after a new setting as soon as you click in a different field.

### 11.2 Number Formatting

When, in **Formatting Options...**, you select the **Data Format ‘Number’**, you get the following options:

**First check or define the Input Format Options.**

Why first? Because this might already have an effect on the **Result** that you will be checking when you start the actual (output) formatting options.

The **Input Format Options** allow you to inform Automation Engine how your input is constructed, how it should be interpreted.
For example: Your input is '123456,789' and you want to indicate now that your Decimal Separator is the ',' (comma) and that there is no Thousands Separator (None). You will already see a change in the Result.

(Output) Formatting Options

Now define how you want this Result formatted.

Note: If the Result is already what you want, this means that the default settings in the Formatting Options panel may already be the ones you want.

- **Decimals**: Indicate how many decimals you want. Select Auto to keep the number of decimals like it was in the original value of the SmartName. See in above screen shot how the setting 'decimals: 2' takes one decimal away and rounds the number to '79'. Another example is to round values with too detailed numbers like '67,500000000 to shorter ones like '67,5' (one decimal left).

- **Decimal Separator**: Select the type you want. In above screen shot we confirm that the comma is the one we want (to keep).

- **Thousands Separator**: Select the type you want. See in above screen shot how it adds the point in between '3' and '4'.

Note: The decimal and thousands separators cannot be identical.

- You can choose to Add leading zeros to the value of the SmartName to get a certain number of Needed Digits. If the value of the SmartName has less digits in the integer part than the specified number of needed digits, leading zeros will be added until the number of needed digits is reached. For example, if the number of needed digits is 5, "123" becomes "00123".

- You can choose to Convert the value of the SmartName to another unit.
  - **Length**: Choose, in both directions, from Millimeters, Centimeters, Decimeters, Meters, Inches, Mils and Points
  - **Angle**: Choose, in both directions, from Radians and Degrees.
  - **File Size**: Choose, in both directions, from Bytes, Kilobytes, Megabytes, Gigabytes and Terabytes.
• **Resolution.** Choose, in both directions, from Pixels per Inch, Pixels per Millimeter, Pixels per Centimeter, Lines per Inch, Lines per Millimeter and Lines per Centimeter.

• **Time.** Choose, in both directions, from Milliseconds, Seconds, Minutes, Hours, Days and Weeks.

### 11.3 Date and Time Formatting

When, in **Formatting Options...**, you select the **Data Format 'Date and Time'**.

It is very important that you first check or define the **Input Format Options...**

### 11.3.1 Input Format Options

**Date and Time** descriptions appear in many different formats. Before a SmartName value can be re-formatted, it will be required to first explain Automation Engine how your input value is constructed. This is what you do in **Input Format Options**.

**Format: ISO 8601**

Select this when your input date and time is in this standard format. This will be the case when the date was constructed by Automation Engine. For example: your input data is '2014-09-27T16:30:00+02:00' (September 27 2014, 4:30 PM in the time zone UTC+2).
11.3.2 Output Formatting Options

Once your input format is well understood by Automation Engine, you define the output Formatting Options.

You here have the same options as in Input Format Options on page 69.

Note: If your Date and Time serves a field in any Automation Engine database, it needs to be formatted according ISO 8601.

11.3.3 Example

You have a workflow where your MIS sends you an XML every time a new Job has to be created on Automation Engine (see an example of such workflows in Examples of integrating with an external system).

The XML from the MIS mentions the Job’s Due Date but has its own particular way of mentioning the date and time. We will need to reformat it to the standard ISO 8601 (that Automation Engine requires in this example).

Follow these steps to get the result as shown in the screen shot:
• In the setup of the SmartName, you see our input value 'Wednesday, October 1 2014, 3:35 AM'. Normally this date would in this canvas not be written out like this but it would show up as another SmartName, one that picks up that date from the XML that we mentioned earlier.

• Click Formatting Options..., do not check or change any settings there yet.

• Click Input Format Options..., Choose Custom and English.

• In Custom Format, type the string that you see in the screen shot. This is what those codes mean:
  • 'EEEE': 'E' means 'day of the week. 4 times 'E' means the day is written out in full text.
  • then there is a comma and a space
  • 'MMMM': 'M' means 'month'. 4 times 'M' means the month written out in full (text).
  • then a space followed by 'd', meaning the day of the month (number)
  • then a space followed by 'y', meaning the year, in (full) number
  • then again a comma and a space
  • then 'h:mm', meaning the hour (in 12 hour system), followed by a colon, followed by the minutes
  • then a space and 'a', meaning 'AM'.
- Indicate that this time is one from the New York time zone.
- Click OK to confirm and close the Input Format Options dialog.
- Now set the (output) Formatting Options: Choose the ISO 8601 format and choose the same New York time zone.
- Check the Result: you should see a ISO 8601 format of the custom description that we received.
- Save the SmartName with a logical name.
- This SmartName (date) is one that we can now use in the Create Job task, for the Job Due date.
12. Using Regular Expressions

Regular Expressions are text patterns that are used to match strings. They can contain a mix of plain text and special characters to indicate what to match. Find more information about this standard tool on Wikipedia.

In Automation Engine, they are used for special cases where SmartNames do not offer a solution. You can use Regular Expressions in

- String Extract SmartNames
- Some tasks: the Select File, Wait for Files, ...
- Some Workflow Controls: Data Splitter, Sort and Router.

12.1 A Selection of Useful Characters in Regular Expressions

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Beginning of the string. ^abc will only lead to a match if the string begins with &quot;PANTONE 120 C&quot;.</td>
<td>^c matches &quot;Cyan&quot; but doesn’t match abc.</td>
</tr>
<tr>
<td>\</td>
<td>OR</td>
<td>cat</td>
</tr>
<tr>
<td>$</td>
<td>End of the string. xyz$ will only lead to a match if the string ends with xyz.</td>
<td>120$ matches &quot;PANTONE 120&quot; but doesn’t match &quot;PANTONE 120 C&quot;.</td>
</tr>
<tr>
<td>.</td>
<td>The . matches any single character.</td>
<td>ab.d matches &quot;abcd&quot; as well as &quot;ab9d&quot; or &quot;ab-d&quot;.</td>
</tr>
<tr>
<td>[abc]</td>
<td>Brackets enclosing a set of characters indicates that any of the enclosed characters may match the target character.</td>
<td>[abc] matches &quot;a&quot;, &quot;b&quot; or &quot;c&quot; but doesn’t match &quot;x&quot; or &quot;y&quot;.</td>
</tr>
<tr>
<td>[^abc]</td>
<td>The ^ inside the brackets means that none of the enclosed characters may match the target character.</td>
<td>[^abc] matches &quot;x&quot; or &quot;y&quot; but doesn’t match &quot;a&quot;, &quot;b&quot; or &quot;c&quot;.</td>
</tr>
<tr>
<td>[a-zA-Z]</td>
<td>The - inside the brackets is used to show a range of characters.</td>
<td>[a-zA-Z] matches all upper and lowercase characters, [0-9] matches all digits.</td>
</tr>
<tr>
<td>\d</td>
<td>matches a numeric character (digit) (exactly the same as [0-9]).</td>
<td>\d\d matches &quot;96&quot; but doesn’t match &quot;5a&quot;.</td>
</tr>
<tr>
<td>\D</td>
<td>matches any non-numeric character (exactly the same as [^0-9]).</td>
<td>\D\D matches &quot;A+&quot; but doesn’t match &quot;A4&quot;.</td>
</tr>
<tr>
<td>\s</td>
<td>matches a whitespace.</td>
<td>\d\s matches &quot;5 &quot; but doesn’t match &quot;5a&quot;.</td>
</tr>
<tr>
<td>\S</td>
<td>matches any non-whitespace character.</td>
<td>\s\S matches &quot;6&quot; but doesn’t match &quot;.&quot;</td>
</tr>
</tbody>
</table>
### Character Description Example

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>\w</td>
<td>matches an alphabetic (word) character (exactly the same as [a-zA-Z0-9]).</td>
<td>\w\w matches &quot;9d&quot; but doesn't match &quot;5+&quot;.</td>
</tr>
<tr>
<td>\W</td>
<td>matches any non-alphabetic character (exactly the same as [^a-zA-Z0-9] or [^\w]).</td>
<td>\W\W matches &quot;+&quot; but doesn't match &quot;5+&quot;.</td>
</tr>
<tr>
<td>?</td>
<td>The ? matches the character to its left 0 or 1 times.</td>
<td>esk\w? matches &quot;esk&quot; as well as &quot;esko&quot; but doesn't match &quot;eskonv&quot;.</td>
</tr>
<tr>
<td>*</td>
<td>The * matches the character to its left 0 or more times.</td>
<td>c* matches &quot;,&quot;, &quot;c&quot;, &quot;cccc&quot; etcetera. esk\w* matches &quot;esk&quot; as well as &quot;Esko&quot;.</td>
</tr>
<tr>
<td>+</td>
<td>The + matches the character to its left 1 or more times.</td>
<td>c+ matches &quot;c&quot;, &quot;cccc&quot; but not &quot;.&quot;. esk \w+ matches &quot;esko&quot; as well as &quot;Esko&quot; but doesn't match &quot;esk&quot;.</td>
</tr>
<tr>
<td>\n</td>
<td>The (n) matches the character to its left exactly n times.</td>
<td>a{3} matches &quot;aaa&quot; but doesn't match &quot;aa&quot; or &quot;abc&quot;. ab{3}c matches only &quot;abbbc&quot;.</td>
</tr>
<tr>
<td>\n,</td>
<td>The (n,) matches the character to its left at least n times.</td>
<td>a{3,} matches &quot;aaa&quot; or &quot;aaaaaa&quot; but doesn't match &quot;aa&quot; or &quot;abc&quot;.</td>
</tr>
<tr>
<td>\n,m</td>
<td>The (n,m) matches the character to its left at least n times, but no more than m times.</td>
<td>a{3,5} matches &quot;aaa&quot;, &quot;aaaa&quot; or &quot;aaaaaa&quot; but doesn't match &quot;aa&quot; or &quot;aaaaaa&quot;.</td>
</tr>
<tr>
<td>\</td>
<td>Escapes characters with a special meaning. The \ can precede a special character (like * or [ ) if you want to search for it literally.</td>
<td>esko\artwork&quot;nv matches &quot;esko \artwork&quot;nv&quot;. ?$ matches &quot;]&quot; at the end of a line.</td>
</tr>
</tbody>
</table>

### Some examples based on the above list

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Matches</th>
<th>Doesn't Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>\d{6}-\w{5}-\d{1,3}.pdf</td>
<td>312454-ESKO4-1.pdf</td>
<td>31245A-ESKO4-1.pdf</td>
</tr>
<tr>
<td></td>
<td>235682-USA12-23.pdf</td>
<td>312456-USA-23.pdf</td>
</tr>
<tr>
<td>PANTONE \d+ C$</td>
<td>PANTONE 120 C</td>
<td>PANTONE 120 U</td>
</tr>
<tr>
<td></td>
<td>PANTONE 15 C</td>
<td>PANTONE 15 CU</td>
</tr>
<tr>
<td>[Yy].*</td>
<td>Yes</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>no</td>
</tr>
<tr>
<td>(Cyan)</td>
<td>(Magenta)</td>
<td>(Yellow)</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>Cyan SpotColor</td>
</tr>
</tbody>
</table>

### 12.2 Using Parentheses to Extract

**Use ( ) in the regular expression to define what to extract**

Parentheses are used to define what you want to extract (to "capture a group").
Example:
Example input string: lemon
Regular expression: le(mo)n
Output: mo

What if I want to extract multiple groups?
Automation Engine does not support this. Only the first group in the string between ( ) will be extracted.

Example:
Example input string: lemonandmango
Regular expression: le(mo)na(nd)man(go)
Output: mo

What if the input string already contains ( )?
When the input string already contains a set of parentheses, you have to "escape" those characters, by adding a \ in front of them.

Example:
Example input string: ab(cd)e We want to extract the group between the ( )
Regular expression: .*\((.*)\).*
In human language: something, then an open parentheses, then the group we want to extract, until there's a closing parentheses, followed by something.
Output: cd

12.3 Examples from Users

These examples came from actual users. You may find more on Esko's Knowledge Base.

Extracting the last character of a string
The SmartNames function Start From Character on page 39 makes it easy to extract the first x characters of a string. With Regular Expressions, you can extract the last character(s). See this example (you can change the number to extract more (last) characters):

Example input string: banana
Regular expression: .*{1}
Output: a

Extracting the server name from the SmartName 'Operator'

Example input string: Chantal@AESERVER01
Regular expression: (.+)@.*
Output: Chantal
Extracting the domain from an E-mail address

Example input string: michael.jackson@esko.com
Regular expression: .+@(.+).+[a-z]+
Output: esko

Extracting the part of a file name after the last underscore

Example input string: Banana_v03_271265_p003.pdf

- with the file extension
  - Regular expression: .*_(.*)
  - Output: p003.pdf

- without the file extension
  - Regular expression: .*_(.*)\..*
  - Output: p003

Extracting the part of a file name between the 2nd and 3rd underscore

Example input string: Banana_v03_271265_p003.pdf
Regular expression: \[^\_\]*\[^\_\]*_(.*)_.*
Output: 271265

Extracting a number from a file name

Example input string: Cosmo453_Advertisement_4.pdf
Convention in human language: something_something_number.pdf
Regular expression: .+_+([0-9]+).pdf
Output: 4

Defining a specific file selection - containing an 'exclude'

In the Select File task, you can use regular expressions in the input fields that end with Name Matches: by ending your input with @regex. This creates possibilities to define very specific selections. Some examples:

- To select all files except files where the name contains _report:
  
  (?!.*_report).*@regex

- To select all files but exclude MAC files (starting with .):
  
  \(!\.).*@regex

- To select all ArtPro files, but exclude the automatically generated backup file:
  
  (?!.* BU).*\..@regex

Defining a specific file selection
- To select the file and the report of the file:

  \texttt{<file/>.pdf|<file/>\_report.pdf@regex}

- To select only files with specific extensions. For example only the ARD and MFG files:

  \texttt{.*\.ard|.*\.mfg@regex}
The chapter "SmartNames" describes the many ways how to create a SmartName that will resolve in a string like "flexo" or "123". But how do you create a SmartName that resolves in Parameter Values that are not a free text string? For example you want a SmartName to decide a button in the interface that indicates to rotate 90 degrees. Or you want a SmartName to decide an internal value that is shown in a drop-down list, for example "Any User"?

We illustrate both cases with an example:

**Creating a SmartName of a User Interface Button**

Some buttons in Automation Engine Tickets can be transformed into SmartNames. Follow these steps to create such a SmartName:

1. Right-click the button and see if the option Copy Parameter Value appears. If so, click the option.

   For example this option is available for the Orientation buttons of the Marks that you can set in the Add Marks to Imposition task. Notice how you can copy their Parameter Value and how the list of buttons also offers to be decided by a SmartName.

2. Go to the dialog where you create the SmartName and right-click and Paste the Parameter Value.

   In this example we create a Job Parameter (that will become available as a SmartName). We here show the manual creation but it would normally be created automatically based on input from an external system. In the Job Setup, we create the Job Parameter Text Mark Rotation and for
its Value we paste (Ctrl-V) the Parameter Value of the button. The screen shot shows that the resulting value is "270".

![Add Job Parameter dialog]

3. Now use that SmartName in the Ticket instead of the button. Click on the SmartName icon and select it.

![Ticket dialog]

Creating a SmartName of an Automation Engine internal value

This example shows how to create a SmartName that will return the value of "Any user" (a choice from a drop-down list):

1. Right-click the interface value and see if the option Copy Parameter Value appears. If so, click the option.

![Edit Ticket of Wait for Action (Checkpoint) dialog]

2. Go to the dialog where you are creating the SmartName and Paste the value there. You will see it appear as "Any User...@".

Here is an example workflow: your Jobs are created via XML files that you receive from your business system. The XML contains the name of the Esko user that should handle this Job and therefore should receive an item in his To Do list (from the Wait for Action (Checkpoint) task...
that is automatically started when the Job is created). The Create Job task maps that user to the custom Job Category 5. The Wait for Action (Checkpoint) task then picks it up as the SmartName Job Category 5.

For this to work, the business system needs to know exactly how to write the Automation Engine User names. In our example, the business system also needs to learn how to tell Automation Engine that the To Do of a Job is for Any User. In that case, the business system needs to mention in the XML that the Job is for "Any User...@", because only that value will match the drop-down choice Any user.

This screen shot shows how the Create Job task created the correct string in the custom Job Category named ForWhichOperator:

And the SmartName of this Job Category is then used in the To: field of the Checkpoint task: