pitney bowes

Customer Engagement

EngageOne® Compose

EngageOne® Designer

Utilities Guide

Version 6.6 Service Pack 9

Table of Contents

1 - Preface

This guide provides detailed information on the utilities that are supplied as part of the EngageOne® Designer licensed product.

In this section

Guide conventions

Typographical conventions

The following are used throughout this manual.

[...]	text between square brackets is optional.
{*opt1*| *opt2*}	parameters between curly braces represent a list of options, one of which must be chosen.
Textin italics	represents parameter data which should be replaced with customized values.

Skills and education

While every effort ismade to provide sufficient information in this and the otheruser manuals, we also recommend that users attend a product education course.Your product supplier will be pleased to advise on course availability.

2 - Importing application designs

Publication designs from other applications can be imported into the Designer in the format expected. To do this the external design must be defined in XSL (Extensible Stylesheet Language) format. This is the markup language used to describe how an XML document of a given type should be displayed. Specifically the import facility works with objects defined using the XSL-FO standard.

This chapter assumes that the reader has a working knowledge of XML and XSL-FO. Using examples it aims to outline how XSL-FO should be written to conform to the requirements of the import facility. It does not attempt to give a comprehensive tutorial for XML/XSL programming.

Detailed information about XSL can be found on the World Wide Web Consortium (W3C) website at **http://www.w3.org/TR/xsl/**

In this section

Glossary of terms

XML **Extensible Markup Language** – XML is a text based markup language that is used to describe data and its structure with a series of inter-related tags. Unlike other markup languages such as HTML, XML tags are not fixed and may be defined as required.

XSL **Extensible Stylesheet Language** – XSL is a language for expressing style sheets. An XSL style sheet is a file that describes how to display an XML document of a given type. XSL is a family of three Recommendations produced by the W3C's XSL Working Group: XSL Transformations, XML Path Language and eXtensible Stylesheet Language (often also referred to as XSL).

XSLT **XSLTransforms** – describe how XML documents are to be filtered and converted (i.e.transformed) into other XML documents, including XSL-FO files. This is also called a stylesheet although it is more like a sophisticated search-and-replace routine.

XPath **XSL Path Language** – this is used within XSLT to specify the parts of an XML document to which transformations are to be applied. XPath models an XML document as a tree of nodes with parent-child and next-previous relationships. These nodes are of different types: element nodes, attribute nodes, text nodes, etc.

XSL-FO **XSLFormatting Objects** – this is the third of the three recommendations – the eXtensible Stylesheet Language. An XSL-FO file contains the medium-specific and appearance-specific "formatting objects" that make up the page. For the print medium, formatting objects can include characters, blocks of text, images, tables, borders, master pages and so on.

ICF Importer Configuration File – this file is generated by the third-party exporter and contains the names and path of all the files related to an XSL-FO file to be imported.

XSL-FO structure

The import facility supports a subset of the full XSL-FO definition. The supported elements and attributes are listed on **Summary of supported XSL-FO elements** on page 37. You must not include unsupported elements or attributes in an XSL fileintended for use withthe import facility.

The import facility can work with all the regular units of measure supported in XSL.

Publications and documents

The *fo:root* element is used to define the overall publication structure. Within this, one or more *fo:layout-master-set* elements define the layout of individual document designs each of which must be partnered with an *fo:page-sequence* element which contains information about actual content. The following fragment demonstrates this structure:

```
<!--Start of publication -->
<fo:root>
<!--Start of document 1 -->
<fo:layout-master-set>
...
</fo:layout-master-set>
<!--Page content for document 1 -->
<fo:page-sequence>
...
</fo:page-sequence>
<!--Start of document 2 -->
<fo:layout-master-set>
...
</fo:layout-master-set>
<!--Page content for document 2 -->
<fo:page-sequence>
...
</fo:page-sequence>
</fo:root>
<!--End of publication -->
```

Defining page layouts

Within an *fo:layout-master-set*information about the pages to be generated must be defined within one or more *fo:simple-page-master* elements. Each element represents a page layout that will, if necessary, be conditionally selected within the Page Setup object in the Designer publication.

An *fo:simple-page-master* contains attributes that define page size and margin information. Additionally it can contain a range of sub-elements that define the layout sections of the page to which content can be added:

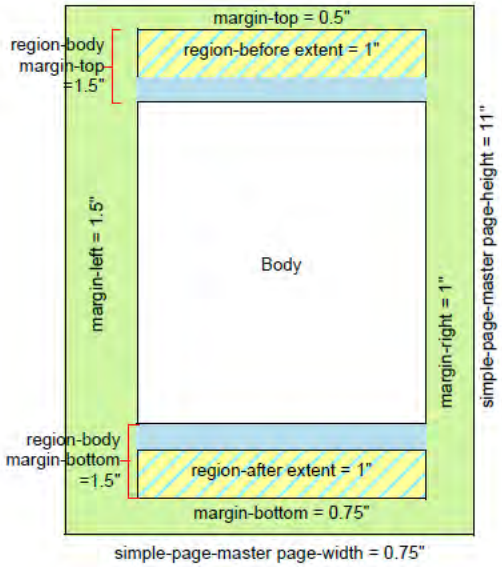
fo:region-before and *fo:region-after* become the publication header and footer sections respectively. A single attribute – extent – defines their height. If these are not coded default header and footer sizes are assumed from the top and bottom page margins.

fo:region-body becomes the body area in the Designer publication. It supports top and bottom margin attributes (independent of the page margins) which must take account of the header and footer spaces if defined.

*fo:region-start*and *fo:region-end*are ignored if coded.

This fragment below demonstrates these points and the code results in a page layout as shown:

```
<fo:layout-master-set>
<fo:simple-page-master master-name="Test"
page-height="11in"
page-width="8.5in"
margin-top="0.5in"
margin-bottom="0.75in"
margin-left="1.5in"
margin-right="1in">
<fo:region-before extent="1in" />
<fo:region-body
margin-top="1.5in"
margin-bottom="1.75in"/>
<fo:region-after extent="1in" />
</fo:simple-page-master>
...
</fo:layout-master-set>
```



Working with multiple page layouts

Where an *fo:layout-master-set* contains more than one *fo:simple-page-master* element this will result in the creation of multiple sections within the logic of the publication. The main use of multiple layouts with the import facility is to allow you to define a change to orientation or the size of headers and footers for different page designs.

> **Note:** that only one page size is supported for each designer document (although you can have different sizes if you are coding multiple document designs). if the *simple-page-master*elements contain differing attributes those from the first element will be used for all sections.

By default, the page layouts will be used in the sequence in which they are coded and a single page of content is expected for each layout.

When importing document designs that can contain varying number of pages for each page layout (e.g. those that use repeated data) you should include *fo:page-sequence-master*and *fo:repeatable-page-master-*

alternatives elements to specify a group which defines the order in which the page layouts/sections will be used. Within the group you will need to code

fo:conditional-page-master-reference elements to cross-reference the layout names and their relative sequences. In the following fragment three page layouts and their sequence are defined.

```
<fo:root>
<fo:layout-master-set>
<fo:simple-page-master master-name="front-page"
<!--page setup attribs. here -->
</fo:simple-page-master>
<fo:simple-page-master master-name="even-page"
<!--page setup attribs. here -->
</fo:simple-page-master>
<fo:simple-page-master master-name="odd-page"
<!--page setup attribs. here -->
</fo:simple-page-master>
<fo:page-sequence-mastermaster-name="duplex">
<fo:repeatable-page-master-alternatives>
<fo:conditional-page-master-referencemaster-reference="front-page"
page-position="first"/>
<fo:conditional-page-master-referencemaster-reference="odd-page"
odd-or-even="odd"/>
<fo:conditional-page-master-referencemaster-reference="even-page"
odd-or-even="even"/>
</fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
</fo:layout-master-set>
<fo:page-sequencemaster-reference="duplex">
<fo:flowflow-name="body">
<!--Include content here -->
...
</fo:flow>
</fo:page-sequence>
</fo:root>
```

Adding content

The *fo:page-sequence* element is used to define actual page content. Within this you should code the *fo:flow* element for content destined for the main document body and the *fo:static-content* element for header or footer content.

Both *fo:flow*and *fo:static-content* require you to code a *flow-name* attribute by which the content is linked to the relevant layout region. By default, the region names to be used with this attribute are:

- *xsl-region-before* - header
- *xsl-region-after* - footer
- *xsl-region-body* - main body

Note: you can redefine these names by specifying the *region-name* attribute in the *region-before*, *region-after* or *region-body* elements within *simple-page-master*.

This fragment adds some simple content to the header and main body regions:

```
<fo:root>
                <fo:layout-master-set>
                <!--page setup here -->
                </fo:layout-master-set>
                <fo:page-sequence
                master-reference="Example">
                <fo:static-content
                flow-name="xsl-region-before">
                <fo:block
               font-size="18pt" font-weight="bold" font-family="Arial">
 Textfor header
                </fo:block>
                </fo:static-content>
                <fo:flow
                flow-name="xsl-region-body">
                <fo:block
                font-size="16pt"
                font-family="Times NewRoman"> Text for main body area

                </fo:block>
                </fo:flow>
                </fo:page-sequence>
</fo:root>
```

Text

Within the *fo:page-sequence*element the *fo:block*element identifies text and its associated formatting attributes

Generally speaking, each *fo:block* element is converted into a separate paragraph object within the Designer publication and the attributes of the block are converted to the equivalent paragraph settings. The first paragraph is offset to the top left of the target region and other blocks follow down the page in the sequence defined in the XSL source.

Where *fo:block*s are nested within the XSL the child objects automatically default to using the attributes of the parent thus allowing you to set common attributes for a range of incoming paragraphs.

Character attributes such as font or color can be changed within a paragraph by coding *fo:inline*elements within an*fo:block*.

The following fragment illustrates these points:

```
 ...
 <fo:page-sequence
 master-reference="Example">
 <fo:block
 font-family="Helvetica" font-size="11pt"
 line-height="16pt" space-before="6pt" space-after="6pt"> Hereis some
text
 <fo:block
 font-family="Arial"
 Newparagraph using Arial. Other attributes as before
 <fo:inlinecolor="red">
 font-weight="bold"
 Boldred text in same paragraph
 </fo:inline>
 <fo:block
 </fo:block>
 </fo:page-sequence>
 ...
```

Tabs

XSL-FO does not natively support tabs but the importer has been adapted to expect a new element within an *fo:block* element that allows you to code tab settings. This is named *g1d1i:tab*. The element has the following attributes:

Name	Valid values	Default	Effect
spacing	Measurement	0.25 inches	Sets fixed tab spacing. Note that this cannot be used in combination with other properties
position	An offset position relative to page left margin	None	Sets a single tab stop relative to the left page margin
alignment	"left" "right" "centre" "character"	"left"	Sets how text is positioned in relation to the tab. If ‘character’ is specified the alignment-character property must be set.
alignment-character	Single text character	None	When alignment = ”character” this specifies the character which will be centered on the tab.

Name	Valid values	Default	Effect
leading-character	Single text character	None	Specifies a character that will be used to pad the space between the current tab stop and the previous offset.

The following fragment illustrates these points:

```
<!-- Use fixed tabs at .75 offsets -->
<fo:block
font-family="Times New Roman"
font-size="11pt">
<g1d1i:tab
spacing="0.75in"/>
Here is some tabbed text:
1#&0009;2#&0009;3
</fo:block>
<!-- Use various tab offsets -->
<fo:block
font-family="Arial"
font-size="10pt">
<g1d1i:tab
position="2in"
alignment="right"/>
<g1d1i:tab
position="4.5in"
alignment="character"
alignment-character="."
leading-character="-"/>
#&0009;See section.#&0009;10
</fo:block>
```

Note: the use of the escape character sequence (#&0009 in this example) to specify the tab character.

Images

Image files referenced by a document design are included by coding the fo:external-graphic element within an fo:block.

The importer will attempt to copy the actual image files referenced in such elements to the Designer repository.

If the element includes an src attribute and this contains a path name the importer will look for the file in this location. Otherwise the importer will look for the file in the location indicated by the Resource Path option which can be specified in the Application Import dialog.

Note that you can specify the action to be taken when the filename already exists as an image within the repository in the options specified when importing – see **Application Import options** on page 35.

XSL-FO does not natively support the full range of anchor positions used in Designer and, by default, all image references are inserted inline within the paragraph text created from the fo:block. However, the importer has been adapted to expect a custom position attribute within the fo:external-graphic element that supports the full range of anchor positions.

The full list of supported attributes for fo:external-graphic including the custom extension is as follows:

Name	Valid values	Default	Effect
src	Path name	None	The importer searches for the image file at this location.
position	"anchor" * "fixed" * "inline"	"inline"	Sets Designer anchor type. See the Designer User's Guide for details of these options.
text-flow	"square" "left" "right" "in front" "behind" "top bottom"	"top bottom"	For positions other than inline this determines how text flows around the image.
alignment	"center" "left" "right" "none"	"normal"	Sets image alignment when using anchored positioning.

The following fragment illustrates these points:

```
font-family="Times New Roman"
font-size="11pt">
An inline image follows:
<fo:external-graphic
position="inline"
src="url('img/icon.bmp')" />
This is text after the inline image
</fo:block>

<fo:block
<!-- Place an anchored image -->
<fo:external-graphic
position="anchor"
```

```
posX="2in" posY="1in" />
<!-- Place a fixed offset image -->
<fo:external-graphic
position="fixed"
posX="4in"
posY="0.5in"
text-flow="square" />
</fo:block>
```

Active Content

Active Content can be imported as an XSL-FO file by specifying it as the Output *Objecttype*. To call Active Content from within an imported XSL-FO file, the importer has been adapted to expect a new element – *g1d1i:call*.Return values areignored,so the element cannot be used, for example, to set a variable. Parameters are set using the child element *g1d1i:call-param*– see below.

If used within an *fo:block* or *fo:inline* element, the Active Content will be placed after any preceding text. To place it inline you must edit the Active Content in Designer to set its first paragraph to merge with the previous paragraph (in the Control tab of the Paragraph dialog box).

The element *g1d1i:call* has one attribute:

Name	Valid values	Default	Effect
name	Alphanumeric Designer file and path name	No default, must be set	Identifies the Active Content and where it is in the Designer

```
<fo:block>
<!-- call Active Content GetString -->
<g1d1i:call name="apps/GetString"/>
</fo:block>
<fo:block>
<!-- Hello & world will, by default, be on two lines -->
<fo:inline font-family="CG Times" font-size="8pt">
Hello <g1d1i:call name="apps/world"/>
</fo:inline>
</fo:block>
```

Parameters for Active Content are set using the *g1d1i:call-param*element – a child element of *g1d1i:call*. It has the following attributes:

Name	Valid values	Default	Effect
name	Alphanumeric name	No default, must be set	Provides the name of a parameter of the Active Content
select	An xpath reference or an	Optional	Sets the value of the parameter

Name	Valid values	Default	Effect
	alphanumeric string		

The value of the parameter can also be specified via the element content, or in a nested <xsl:value-of>.

Validation of the parameter name is not carried out by the importer, so you must ensure that the names match those of the Active Content. Any parameters in the Active Content that do not have a matching name will be passed a blank value. If more than one value is specified for a *g1d1i:call-param*element, only the last one will be used.

The following fragments illustrate Active Content calls:

```
<!-- Call Active Content CustID with a parameter of 123 -->
<g1d1i:call name="apps/CustID">
<g1d1i:call-param name="Parameter1">123</g1d1i:call-param>
</g1d1i:call>
<!-- Call Active Content CustID with a parameter of rhg2 -->
<g1d1i:call name="apps/CustID">
<g1d1i:call-param name="Parameter1" select="rhg2"/>
</g1d1i:call>
<!-- Call Active Content CustStatus with 2 parameters:
$TestVar,//apcm/Strf -->
<g1d1i:call name="apps/CustStatus">
<g1d1i:call-param name="p1"><xsl:value-of select="$TestVar"/>
</g1d1i:call-param>
<g1d1i:call-param name="p2"><xsl:value-of
select="//apcm/Strf"/></g1d1i:call-param>
</g1d1i:call>
```

System variables

All the Designer system variables can be accessed by the importer, which has been adapted to expect a new element – *g1d1i:env-var* – in an fo:block or fo:inline element. The element *g1d1i:env-var* has two attributes:

Name	Valid values	Effect	Default
name	Designer system variable name: ***author*** ***generation-date*** ***page-count*** ***page-number*** ***page-x-of-y*** ***publication-name***	Identifies the system variable to be set	No default, must be set

Name	Valid values	Effect	Default
	run-date ***run-time***		
format	Can be in any order. Upper case only. Single separators can be specified. All Designer date formats are supported. ***D*** - The day number, no leading zero, e.g. 8 ***DD*** - The day number with a leading 0, e.g. 08 ***DDD*** - One digit cardinal, e.g. 8th ***M*** - The month number, no leading zero, e.g. 8 ***MM*** - The month number with a leading 0, e.g. 08 ***MMM*** - Abbreviated month name (from locale), e.g. Aug ***MMMM*** - Full month name (from locale), e.g. August ***Y*** - The last digit of the year, 9 ***YY*** - Two digit year number, e.g. 09 ***YYYY*** - Full four digit year, e.g. 2009	Specifies formatting to be applied to run-date and generation-date only.	Default Designer date formatting

For example:

```
<fo:block>
Generated by <g1d1i:env-var name="author"/>
on <g1d1i:env-var name="generation-date" format="D-MMM-YYYY"/>
<!-- e.g 'Generated by John on 24-Feb-2009' -->
</fo:block>
```

References to input data

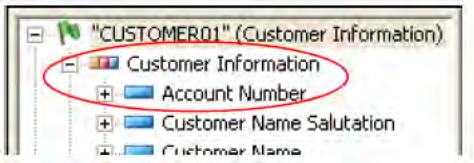
The importer cannot create new definitions for data elements in the Designer environment but it can add data aliases to a document design provided an appropriate data dictionary or data map has been created in advance. These can be specified as options in the *Application Import* dialog. The selected data dictionary or map must contain definitions of all data elements that are referred to by the XSL:FO source file.

The importer uses references in *xsl:value* elements to cross-reference to entries in the data dictionary or map. The select attribute of this element must contain an expression that represents a data

element as it would be displayed in the Designer Data Format Editor. This is coded using a limited XPath format. For example:

```
<xsl:value-of
select="//Customer/Account Number" />
```

Where *//Customer* identifies the record name and */Account Number* the field name as defined to Designer.



You can use the new field references directly within a paragraph by coding the elements within an *fo:block* structure.

Alternatively, you can specify that the data alias is to be used to populate a variable within the document logic. This is done by coding an xsl:variable element where the name attribute becomes the variable name within Designer.

The following fragment illustrates these points:

```
...
<!-- Output field in a paragraph -->
<fo:block
font-family="Arial"
font-size="14pt"
color="black"
Account No:
<fo:inline color="blue">
<xsl:value-of
select="//Customer/Account Number"/>
</fo:inline>
</fo:block>

<!-- Pass field to a variable -->
<xsl:variable
name="AccountNo >
<xsl:value-of
select="//Customer/Account Number"/>
</xsl:variable>
...
```

Repeating data

Code *xsl:for-each* elements to define repeating data structures that can cross-reference to existing repeating data definitions in the data dictionary or map. The select attribute of this element must contain an expression that represents the name of the repeating Designer data element. Within the

xsl:for-each you will need to code xsl:value-of elements to identify each field from the repeating data that you actually want to use in the document.

You can choose to use repeating data in two ways:

If the *xsl:for-each* is coded outside of a block element the importer will create a Designer repeating data object and include the data aliases referenced by the for-each within it. You will need to code fo:block elements within the *xsl:for-each* actually to output fields in a paragraph.

If the *xsl:for-each* is itself coded within an *fo:block* element the importer will build Designer objects that concatenate all instances of the data and/or text within the *xsl:for-each* and then output the concatenation within the paragraph created from the *fo:block*.

The following fragment illustrates these points:

```
...
<!-- 1. Create a repeating data object and output a field from it
iteratively -->
<xsl:for-each
select="//Service Detail">
<fo:block
font-family="Arial"
font-size="9pt">
<xsl:value-of
select="//Service Detail/Type"/>
</fo:block>
</xsl:for-each>
<!-- 2. Output a concatenation of a field from repeating data -->
<fo:block
font-family="Arial"
font-size="9pt">
<xsl:for-each
select="//Service Detail">
<xsl:value-of
select="//Service Detail/Type" />
<fo:inline> -- </fo:inline>
</xsl:for-each>
</fo:block>
...
```

For example, assume that iterations of the 'Type' field referenced above contain the following text strings when the document design is used with actual input data: “Long Distance”, “Mobile Service” and “Local”.

Example 1 would produce:

Long Distance
Mobile Services
Local

Example 2 would produce:

Long Distance--Mobile Service-Local--

Sorting

Within an *xsl:for-each* element you can instruct that the associated repeating data structure be sorted by including an xsl:sort element. Use the order attribute to specify the sort direction. For example:

```
<xsl:for-each
xsl:sort select="//Service/Date">
order=”descending”
<fo:block
font-family="Arial"
font-size="12pt">
<xsl:value-of
select="//Service Detail/Type"/>
</fo:block>
</xsl:for-each>
```

Including program logic

The importer supports the use of *xsl:if* and *xsl:choose/when/otherwise* elements in the creation of condition objects within the Designer environment.

Logic can be used in the following circumstances:

- within publication logic to select specific document designs
- within document logic to create and select document sections
- within the main document body.

You may only use such logic at fixed points in the XSL source if it is to be translated by the importer.

For document selection within publications only use the condition elements:

- between *fo:root* and first *fo:layout-master-set*;
- between the last *fo:page-sequence* of a document and the *fo:layout-master-set* of the next;

As in the following fragment:

```
<<fo:root>
<!-- Select doc by language -->
<xsl:variable name="Language"
select="//DOCINFO/LANGUAGE" />
<!-- English? -->
<xsl:if test="$Language = ‘EN’" -->
<fo:layout-master-set>
...
</fo:layout-master-set>
<fo:page-sequence>
...
</fo:page-sequence>
```

```
</xsl:if>
<!-- French? -->
<xsl:if test="$Language = 'FR'" -->
<fo:layout-master-set>
...
</fo:layout-master-set>
<fo:page-sequence>
...
</fo:page-sequence>
</xsl:if>
</fo:root>
```

For section creation and selection within documents only use the condition elements:

- between the *fo:layout-master-set* and *fo:page-sequence* elements;
- and between successive *fo:page-sequence* elements.

As in the following fragment:

```
...
<fo:layout-master-set>
...
</fo:layout-master-set>
<xsl:variable name="DocType"
select="//DOCINFO/TYPE" />
<!-- Normal account? -->
<xsl:if test="$DocType = 'N'" -->
<fo:page-sequence>
...
/fo:page-sequence>
</xsl:if>
<!-- Preferred account? -->
<xsl:if test="$DocType" = 'P'" -->
<fo:page-sequence>
...
</fo:page-sequence>
</xsl:if>
...
```

Logic intended for the main document body must be placed inside the *fo:flow* element within *fo:page-sequence* elements. You may not code conditions intended for header or footer sections.

The following fragment illustrates these points:

```
<fo:page-sequence>
<fo:static
flow-name="Header" >
<fo:block>Header</fo:block>
</fo:static>
<fo:static
flow-name="Footer" >
```

```
<fo:block>Footer</fo:block>
</fo:static>
<fo:flow
flow-name="Body">
<g1d1i:number
name="Current"
expression="0"/>
<xsl:for-each
select="//Service">
<g1d1i:number name="Current"
expression="var.Current + 1"/>
<xsl:variable name="Service"
select="//Service/Description"/>
<fo:block
font-family="Arial"
font-size="12pt"
color="black">
<xsl:choose>
<xsl:when
test="$Current = 1">
<fo:block>
First service is
'<xsl:value-of
select="$Service"/>'
</fo:block>
</xsl:when>
<xsl:when
test="$Current = 2" >
<fo:block>
Second service is
'<xsl:value-of
select="$Service"/>'
</fo:block>
</xsl:when>
<xsl:otherwise>
<fo:block>
Service number
'<xsl:value-of
select="$Service"/>'
</fo:block>
</xsl:otherwise>
</xsl:choose>
</fo:block>
</xsl:for-each>
</fo:flow>
</fo:page-sequence>
```

Manipulating numbers

Within program logic it may be necessary to manipulate number values, for example to provide a counter function as part of a repeated condition.

The importer has been adapted to expect a new element within the XSL source to support this requirement. This is named g1d1i:number. When such an element is encountered the importer will create a number variable within the Designer logic. The variable can be updated and queried by the XSL sequence.

The attributes of g1d1i:number are as follows:

ollowing attributes:

Name	Valid values	Default	Effect
name	Alphanumeric variable name	No default, must be set	Sets the variable name
type	"number", "counter"	"number"	Sets the variable type. These are according to the definitions within the Designer environment
expression	Any numeric expression or a Designer formula expression that computes to a number	"0"	Sets the variable value

If you need to update the variable with an expression that contains its previous value you will need to use the 'var.name' syntax which is part of the Designer formula language. For example:

```
<g1d1i:number name="pCount"
expression="var.pCount + 1"/>
```

You can also use the formula language to include references to other known values from the Designer environment in your equation. Details of all formula language syntax can be found in the Designer User's Guide.

To output the variable or refer to it within a condition, code the standard XSL variable reference in the format $name.

The following fragment illustrates these points:

```
...
<!-- Initialize -->
<g1d1i:number
name="Page"
expression="3"/>
<!-- Update -->
<g1d1i:number name="Page"
expression="var.Page + 10"/>
<!-- Query -->
<xsl:if test="$Page = 10" -->
...
</xsl:if>
...
```

Currency parameters - known issue

When dealing with Currency parameter types there is a known issue whereby you must use a Curr() cast when comparing them to a value in a condition.

The following will give you a warning and fail to import correctly:

```
<xsl:if test="$CurrencyParameter = 1987.12">
```

To avoid this issue use:

```
<xsl:if test="$CurrencyParameter = Curr(1987.12)">
```

Page numbering

The standard XSL *fo-page-number* element can be used to supply the current page number to Designer.

It is also a common requirement to place 'page x of y' text within the application, where 'y' is the total number of pages. To satisfy this the importer has been adapted to expect a new element in the XSL source that supplies the total. The name of this element is *g1d1i:publication-page-count*. It has no attributes.

The element can be used within *fo:block* and *fo:inline elements*. The following fragment demonstrates:

```
...
<fo:block
font-family="Arial"
font-size="16pt"
color="black">
Page number
<fo:page-number />
of
<g1d1i:publication-page-count />
</fo:block>
...
```

Importer configuration file

The importer configuration file (ICF) is an XML file containing a number of elements that tell the importer the name and location of the files required to import an XSL:FO document. It is used by both the GUI and Windows Service versions of the importer.

In the sections that follow, relative file paths are relative to the location of the ICF.

<G1IMPORT>

Description	ICF root element	
Attributes	[EXPORTEDDOC]	Exporter supplied name, not used by the importer
	VERSION ICF	Version number
Contents	DOCUMENT	one only
	DATAFORMAT	one only
	DATAMAP	one only
	PROJECT	one only and optional – import project path
	OPTIONS	one only
	LOCALE	any number
	SELECTION	any number
	RESOURCES	any number
Example	<G1IMPORT EXPORTEDDOC="DOCUMENT-9.fo" VERSION="1">	

<DOCUMENT>>

Description	Mandatory element with the path to the document or documents to be imported.	
Attributes	`FILE`	file path of an XSL-FO document to be imported, or the path to the directory if importing in batch mode.
Contents	None	
Example	`<DOCUMENT FILE="Doc\DOCUMENT-9.fo" />`	

<DATAFORMAT>

Description	Optional data format file	
Attributes	`FILE`	file path and name of the data format file
Contents	None	
Example	`<DATAFORMAT FILE="Doc\DataStructure.xml" />`	

<DATAMAP>

Description	Optional data mapping information	
Attributes	`MODE`	mapping mode – valid values are *MANUAL* and *NONE*. Default is *NONE*
	`[NAME]`	repository data map name – must be set if *MODE* is *MANUAL*
Contents	None	
Example	`<DATAMAP MODE="MANUAL" NAME="Test Data Map" />`	

<RESOURCES>

Description	Optional resource file group	
Attributes	None	
Contents	FONT	any number
	IMAGE	any number
	LOCALE	any number
	FONT	any number
Example	<RESOURCES>…</RESOURCES>	

<FONT>

Description	Optional font file	
Attributes	`FILE`	file path and name of Windows font to be imported
Contents	None	
Example	`<FONT FILE="fonts\bigcorplogo.ttf" />`	

Running the importer

The Application Import utility is run independently of the Designer and allows you to import publication files, along with their associated data structures, resources, Active Content and Public Document objects into the Designer. These can be supplied as XSL-FO, XML and MS Word doc/docx files. Note that you must have MS Word 2010 or 2013 installed on the machine running the Application Import utility if you wish to import Word based document designs.

> **Note:** Some bespoke users of the importer require an additional file and settings for evaluating input data. These should only be used if you have been instructed by your product supplier.otherwise, leave them blank.

There are a number of limitations you need to be aware of when importing Word documents into Designer through the Application Import utility. In particular, results may be unpredictable when:

- Nested lists are included;
- Numbered lists start at a value other than 1;
- Numbered lists contain characters proceeding the numeral value, for example (1);
- Lists are manually included, i.e. where a list style has not been applied;
- Multiple indenting methods within the same paragraph or list are used;
- Character spaced content is included;
- Strike through font styles are used;
- Soft carriage returns are included;
- Gutter margins have been defined;
- Tab stops are defined outside the document margin;
- Continuous section breaks have been defined;
- Table cells have been vertically merged;
- A document submitted for review has not been fully approved;
- Certain drawing objects are included, for example, horizontal and vertical lines.

Running the Application Import utility

Run the AppImport.exe program, by default this is installed at the following location:

```
c:\Program Files\{PBBI CCM | Group1}\DOC1\Designer\Client)
```

If Designer is not already running, you are prompted to connect to the repository. Use your usual Designer user name and password to log on. Note that you must have at least read and create/edit privileges on the relevant items in order to import an application.

Importing an application

To import an application: on the File menu, select Open – or use the Import File/Import folder browse button to search as required

If a publication with the same name already exists, this will not be overwritten – the newly imported publication will have the date and time appended to the name.

You can override certain attributes of the publication before importing such as the page size, default font, image handling.

XSL-FO Application Import Utility

Input		
Attributes	Import file/Import folder	When the Batch mode check box is not selected this field is labeled Import File. Enter or browse to the full path to the file to be imported. Otherwise, when the Batch modecheck box is selected, this field is labeled Import folder. Enter or browse to the root folder of the files to be imported.
	Batch mode	When this option is selected, all the files in the path specified in Import folder are imported.
	Confirm each file	Select this option to be prompted to confirm the input of each file found in the import folder when importing in Batch mode. You will also be able to cancel the import process.
	Resource path	The root path to any image, font and locale resources. All resources found in this directory are imported into the repository.
Output		

Object type	Select the type of object to be created from the imported application – either a Publication, Active Content or Public Document. When Active Content or Public Document is created you can mark it so that it can be published independently by selecting the Make publishable check box.
Name	The name given to the Publication, Active Content or Public Document created at the end of a successful import.
Project	Specify the project where you want the publication or Active Content to be created. Either select an existing project or create a new one.
Folder	Specify the folder in the project where the publication , Active Content or Public Document is to be created. Select None to import to the project level, or you can select an existing folder or create a new one. Select Use folder hierarchy to replicate the hierarchy of the import folder.
Data mapping	If the input file contains data references then you must specify a data dictionary or data map to resolve them.

If for any reason this connection is broken, use the Connect button to re-establish it.

The files are validated and any errors reported in the log window. If this is not displayed click the Show Log button. If there are no errors, click Import. The publication is imported into the Project root folder specified.

Changing import settings

To change the import settings: on the Tools menu, select Options and change the settings as required.

Application Import options

Presentation		
	Page	Override the page size and orientation as defined in the imported application.
	Default font	Define a font to be used when a font that is used in the imported file is not available in the Designer.
Resources		
	Ignore duplicate images	When selected, images are not imported if an image of the same name already exists in the repository. Otherwise duplicate images will have the date and time appended to the name.
	Prefix image name with publication name	When selected, the imported image names are in the format <Publication name>_<Image name>.
	Search sub-directories	When selected, any resource files in the resource path sub-directories will also be imported.
Data		
		This page contains settings for bespoke users only – refer to the note **Running the importer** on page 31.
Locale		
		This allows you to set a specific locale for the imported publication. Otherwise, the publication will be created using the default Designer locale.
Performance		

You can optimize the XSL:FO file being imported which will attempt to reduce the complexity of the resulting Designer publication. The original file will not be modified.

Summary of supported XSL-FO elements

The table below can be used as a guide to all the elements supported by the importer.

Note: unsupported elements and element attributes will be ignored and skipped. An appropriate warning will be placed in the import log.

fo:basic-link	
	external-destination
fo:block	
	break-before
	color
	font
	font-family
	font-size
	font-stretch
	font-style
	font-weight
	keep-together
	keep-with-next
	line-height
	margin-left

	margin-right
	orphans
	space-after
	space-before
	start-indent
	text-align
	text-align-last
	text-indent
	white-space-collapse
	white-space-treatment
	widows
fo:conditional-page-master-reference	
	master-reference
	odd-or-even
	page-position
fo:list-block	

	break-before
	provisional-label-separation
	provisional-distance-between-starts
	keep-with-next
	keep-together
fo:list-item	
fo:list-item-body	
	start-indent
	end-indent
	text-decoration
fo:list-item-label	
	start-indent
	end-indent
	text-decoration
fo:page-number	

	text-decoration
	font
	font-family
	font-size
	font-style
	font-weight
fo:page-sequence	
	master-reference
fo:page-sequence-master	
	master-name
fo:region-after	
	extent
fo:region-before	
	extent
fo:table-body	
fo:table-caption	
fo:table-cell	

	column-number
	ends-row
	height
	number-columns-spanned
	starts-row
fo:table-column	
	column-number
	column-width
	number-columns-spanned
	starts-row
	width
fo:table-footer	
fo:table-header	
fo:table-row	
	height
fo:wrapper	
g1d1i:call	
	name
g1d1i:call-param	

	name
	select
g1d1i:env-var	
	name
fo:external-graphic	
	src
	text-flow
	position
	alignment
fo:flow	
	flow-name
fo:inline	
	color
	font
	font-family
	font-size
	font-style
	font-weight
	text-decoration
	linefeed-treatment
fo:layout-master-set	

fo:leader	
	color
	font
	font-family
	font-size
	font-style
	font-weight
	leader-alignment
	text-decoration
fo:region-body	
	margin-top
	margin-left
	margin-bottom
	margin-right
	start-indent
	end-indent
	space-before
	extent
	space-after
fo:repeatable-page-master-alternatives	

fo:root	
fo:simple-page-master	
	master-name
	page-height
	page-width
	margin-top
	margin-left
	margin-bottom
	margin-right
fo:single-page-master-reference	
	master-reference
fo:static-content	
	flow-name
fo:table	
	height
	width
	table-layout
g1d1i:number	

	name
	type
	expression
g1d1i:publication-page-count	
	text-decoration
	font
	font-family
	font-size
	font-style
	font-weight
g1d1i:tab	
	alignment
	alignment-character
	leading-character
	position
	spacing
xsl:if	
	test
xsl:choose	
xsl:when	

	test
xsl:otherwise	
xsl:for-each	
	select

Example of an XSL-FO file

This code imports as a document consisting of three pages with varying headers and footers.

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

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
 <fo:simple-page-master master-name="front-page"
  page-height="148mm" page-width="105mm"
  margin-top="5mm" margin-bottom="5mm"
  margin-left="10mm" margin-right="10mm">
    <fo:region-body margin-top="0.5in" margin-bottom="0.5in"/>
 </fo:simple-page-master>

 <fo:simple-page-master master-name="odd-page">
  <fo:region-before region-name="odd-header" extent="0.75in"/>
  <fo:region-body margin-top="0.5in" margin-bottom="1in"/>
  <fo:region-after region-name="footer" extent="0.5in"/>
 </fo:simple-page-master>

 <fo:simple-page-master master-name="even-page">
  <fo:region-before region-name="even-header" extent="0.75in"/>
  <fo:region-body margin-top="0.5in" margin-bottom="0.75in"/>
  <fo:region-after region-name="footer" extent="0.25in"/>
  </fo:simple-page-master>
  <fo:page-sequence-master master-name="duplex">
   <fo:repeatable-page-master-alternatives>
   <fo:conditional-page-master-reference master-reference="front-page"
    page-position="first"/>
   <fo:conditional-page-master-reference master-reference="odd-page"
    odd-or-even="odd"/>
     <fo:conditional-page-master-reference master-reference="even-page"

    odd-or-even="even"/>
     </fo:repeatable-page-master-alternatives>
  </fo:page-sequence-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="duplex">
   <fo:static-content flow-name="odd-header">
    <fo:block text-align="center">
     This appears in the headers of the odd pages
    </fo:block>
   </fo:static-content>

  <fo:static-content flow-name="even-header">
   <fo:block text-align="center">
    This appears in the headers of the even pages
```

```
   </fo:block>
  </fo:static-content>

  <fo:static-content flow-name="footer">
   <fo:block text-align="center">
    This appears in the footers of all pages
   </fo:block>
  </fo:static-content>
  <fo:flow flow-name="xsl-region-body">
   <fo:block font-size="11pt" font-family="Arial">
    This text appears on the first page
   </fo:block>
   <fo:block font-size="11pt" font-family="Arial" break-before="page">
    This text appears on the second page
   </fo:block>
   <fo:block font-size="11pt" font-family="Arial" break-before="page">
    This text appears on the third page
   </fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

3 - Designer Audit Trail

User activity in the Designer can be logged if required, and extracted for audit trail purposes or similar. When enabled such tracking includes successful and failed login attempts, user management and security changes, creation and editing of design assets, version control activities and key workflow activities such as publishing. The log itself is created within the Designer SQL database, but you can create a report from it as required.

The activity log is disabled when a Designer environment is first installed and must be specifically turned on if required. The command line utility DesignerAudit is used to enable/disable logging, to perform extracts from the log and to clear the log as required. If you do enable audit trail logging it is strongly recommended that you perform regular housekeeping as the additional data can grow large over time and may affect overall performance of the Designer repository.

> **Note:** the Designer Audit utility must be run by a database user with a minimum of *db_owner* access rights.

In this section

DesignerAudit – Windows only

Description	To manage the Designer audit trail logging and reporting.
Preparation	DesignerAudit is run in the Designer server folder from a command prompt, for example: `C:\Program Files\PBBI CCM\DOC1\Designer\Server\DesignerAudit`

Syntax:

```
DesignerAudit {/start [<repos_options>] |
/stop [<repos_options>] |
/report:<file> [/from:<date>] [/to:<date>] [<repos_options>]|
/clean [/to:<date>] [<repos_options>] |
/status [<repos_options>] |
/help}
```

where <repos options> are:

```
/server:<server> [\<instance name>]
/port:<port>
/user:<username>
/pass:<password>
/repos:<repository>
```

Parameters	*/start:*	Start logging Designer activity. Use `<repos_options>` to specify a repository other than the default Designer repository.
	/stop:	Stop logging Designer activity. Use `<repos_options>` to specify a repository other than the default Designer repository.
	/report:	Create an audit report of logging that has taken place. Use `<file>` to specify the path and filename of the report – a comma-separated file (.csv). This option can be combined with /clean providing the from option is not used. Options are: `from` – can be used to specify the start date of the report. Otherwise the report will start from the earliest date. <date> is in the format yyyy/mm/dd or yyyy-mm-dd `to` – can be used to specify the end date of the report. Otherwise the report will end with the latest date. <date> is in the format yyyy/mm/dd or yyyy-mm-dd `<repos_options>` – specify a repository other than the default Designer repository.
	/clean:	Remove log entries from database. This can be combined with /report to remove the same log entries once the report has been generated. Options are: `to` – remove old log entries up to the specified date. Otherwise all entries will be removed. `<date>` is in the format yyyy/mm/dd or yyyy-mm-dd `<repos_options>` – specify a repository other than the default Designer repository.
	/status:	Show if the audit logging is running or not. Use `<repos_options>` to specify a repository other than the default Designer repository.
	/help:	Show the help for DesignerAudit. This is the default option.
<repos_options>	DesignerAudit will automatically connect to the default Designer repository. You can specify a different repository using these repository options.	

	/server:	The name of the Designer database server. Use \instance if there is more than one instance running.
	/port:	The port used for the database.
	/username:	A database user name. When specified a password must also be supplied. If not specified a Windows trusted connection will be used.
	/password:	The database password for specified user. This is only required when a username is specified.
	/repos:	The name of the repository. If not specified the default Designer repository will be used.

Examples:

Show help for all options:

```
DESIGNERAUDIT
```

Start logging Designer activity on the default repository:

```
DESIGNERAUDIT /start
```

Create a report of the complete log:

```
DESIGNERAUDIT /report:”c:\reports\drep-full.csv”
```

Create a report of the log up to a specified date and then remove them:

```
DESIGNERAUDIT /report:”c:\reports\dbrep.csv” /to:2012-02-01 /clean
```

Stop logging Designer activity:

```
DESIGNERAUDIT /stop /server:dbServ /user:dbadmin /password:dBa123X
```

4 - Batch Publishing

Publishing resources for production by Generate or EngageOne® is normally performed in the Designer using the resources available in the current repository. DOC1Publish is a command line utility that allows you to publish using a batch file or scheduler, without the need to run Designer. This means that you can publish large numbers of production resources.

> **Note:** any active content, production job or engageone template referenced by doc1publish must be available in the associated repository.

In this section

About DOC1Publish

DOC1Publish allows you to perform all publishing tasks related to publications, Active Content and EngageOne templates from the command line. Process logs are generated at a configurable location.

Parameters that control the publication process can either be provided to DOC1Publish on the command line or through an XML configuration file (doc1publish.xml). If you choose to use a configuration file to control DOC1Publish it is possible to publish multiple production jobs, Active Content and EngageOne templates from a single repository. However, when parameters are supplied on the command line only a single publishing task can be run at a time. Note that a sample configuration file and associated schema is provided with the product distribution material.

When using a production job the Location settings of the associated host object (as specified within Designer) will be used as default values for Output and Outputfilename when neither of these parameters is specified.

Designer configured for Single Sign-On (SSO)

When Designer has been configured for SSO, DOC1Publish can automatically use the currently logged in Windows user to log into the Designer repository. Note that the logged in Windows user must have the appropriate authorization.

To use DOC1Publish in SSO mode, omit the user credentials from the command-line or, if using configuration file, enter blank values for the user credentials elements. Refer to **DOC1Publish configuration file** on page 60 for further information.

DOC1Publish – Windows only

Purpose	To publish Designer resources ready for production without loading the Designer client.
Preparation	DOC1Publish is run from the Designer's client folder from a command prompt. The Designer repository containing the required production resources must be accessible to the program. The DOC1Publish executable is located at: `c:\Program Files\{PBBI CCM | Group1}\DOC1\Designer\Client)`

Syntax:

```
DOC1PUBLISH {/c:file|(/u:username /p:password
{/JOB:objname|/AC:objname|/TPT:objname}
[/OutputFilename:OutputFile] [/Output:path]
/OutMode:{Complete|Design|Resource|OAC]}
[/Verbose] [/LogFile:OutputFile] [/BypassOutputChannel]
[/TargetRepo:name]
[/ApplyLabel:LabelName] [/Label:LabelName] [/Graphics])}
```

Parameters	*/c:*	Use this option to indicate that parameters are specified in a configuration file rather than on the command line. File identifies the path/file name of the configuration file to be used with this job. The other parameters that follow in this section are not processed when this option is used. See “DOC1Publish configuration file” on page 47 for details.
	/u:	username identifies the user name for the repository
	/p:	password associated with the user name.
	/JOB:	objname is the full path and name of the production job to be published. Note that one of /JOB, /AC, or /TPT must be specified. This is in the form: //project[/folder]/jobname.
	/AC:	objname is the full path and name of the Active Content to be published in the form: //project[/folder]/jobname.
	/TPT:	objname is the full path and name of the EngageOne® template to be published in the form: //project[/folder]/jobname.
	/OutputFilename:	OutputFile is the full path and name of the HIP, Active Content or EngageOne template to be produced. If neither this parameter nor /Output is specified then the Location settings of the associated host object (as specified within Designer) will be used. Otherwise the default is the production job name.
	/Output:	path is a local Windows or UNC path and specifies the file location for the HIP, ZIP and log files. If neither this parameter nor /OutputFilename is specified then the Location settings of the associated host object (as specified within Designer) will be used. Otherwise the default is the current directory.

/OutMode:	this controls the type of publishing: Complete – publication or Active Content design and resources are published. Design – publication or Active Content design is published. Resource – publication resources are published. OAC – when used for: – a production job all the Active Content in the publications in the production job are published in the form of HIP files for Generate to process; – Active Content the specified Active Content is published in the form of a HIP file for Generate to process; – an EngageOne® template all the Active Content in the EngageOne template is published to a zip file suitable for the EngageOne environment. If the OutMode switch is omitted the defaults are as follows: Publication production job – the Complete option is invoked. Active Content production job – the value from the production job will be used. Active Content – a zip file is created suitable for the EngageOne environment. EngageOne template – this switch is not necessary for EngageOne templates.
/Verbose:	when included on the command line all warning and error messages are logged together with stack traces. If this switch is not included, only basic error messages are logged.
/LogFile	OutputFile is the full path and name of the log file. You can optionally create a log file to capture logging information during the execution of DOC1Publish.

/BypassOutputChannel	Facilitates a batch run of EngageOne Server without the need to run any pre-processing of data. This option is valid for non-interactive templates only and will be ignored for interactive templates. Note the following: • Device(s) must be explicitly enabled via the output device setting **Enable as EngageOne Device** in Designer. • Only DIJ and SYS_XPJ Journals are supported using doc1publish. For other journal types, use the Designer Publish Wizard, refer to the Designer User's Guide for details.
/TargetRepo:	used in a multiple repository scenario. Name indicates either a network or local repository alias. Note that this value is case sensitive. If this switch is not included then the default repository is used.
/ApplyLabel:	used to apply a label to the associated publishable object. LabelName is the user defined label for the object to be published. If LabelName is omitted then a system defined label is applied. A date and timestamp will be appended to the new label if the name already exists. Note that a description cannot be included with this switch.
/Label:	uses the publishable object with the label LabelName allowing earlier versions of the object to be used. Otherwise the latest version is used.
/Graphics:	includes resources that are required when Active Content is used with advanced graphics features, such as in a rotated text box.

Examples

Show help for all options:

```
DOC1PUBLISH
```

Basic form:

```
DOC1PUBLISH /U:admin /P:admin /JOB:pdftest
```

Basic form SSO:

```
DOC1PUBLISH /JOB:pdftest
```

Basic form Bypass Output Channel Processing:

```
DOC1PUBLISH /JOB:pdftest /BypassOutputChannel
```

Specifying a folder location for the production job:

```
DOC1PUBLISH /U:admin /P:admin /JOB:"//Master/BwProj/Boardwalk PDF"
```

Specifying a UNC output folder location:

```
DOC1Publish /u:admin /p:admin /Job://Master/MyJob
/Output:"\\Prod1\Broadwalk Project"
```

In a multiple repository environment specify a particular repository:

```
DOC1PUBLISH /U:admin /P:admin /JOB://OpenTest/pdftest
/TargetRepo:DOC_Rep4
```

Using a configuration file to supply parameters to DOC1Publish

```
DOC1PUBLISH /c:d:\multiOut\configfile.xml
```

DOC1Publish configuration file

The DOC1Publish configuration file is an XML file containing elements that supply DOC1Publish with the parameters during its execution. The tables below detail the structure of this file.

<DOC1PublishConfig>

Description	DOC1Publish root element	
Attributes	None	
Contents	*UserCredential*	one only
	OutputPath	one only
	Verbosity	one only
	LogFile	one only
	TargetRepository	one only
	JobInfo	one or more
Example	`<DOC1PublishConfig>…</DOC1PublishConfig>`	

<UserCredential>

Description	Mandatory authentication group	
Attributes	None	
Contents	DOC1UserName	one only
	DOC1Password	one only

<UserCredential>

Example	`<UserCredential>…</UserCredential>`

<DOC1UserName>

Description	User name for the required repository. Note that when Deisgner has been configured for SSO leave the value blank.
Attributes	None
Contents	User name data

Example	`<DOC1UserName>Admin</DOC1UserName>`
	SSO example:
	`SSO <DOC1UserName></DOC1UserName>`

<DOC1Password>

Description	Password associated with the user name for the required repository. Note that when Deisgner has been configured for SSO leave the value blank.
Attributes	None
Contents	Password data

<DOC1Password>

Example	`<DOC1Password>admin</DOC1Password>`
	
	SSO example:
	
	`SSO <DOC1Password></DOC1Password>`

<OutputPath>

Description	Optional output file location for HIP or zip files. If neither this element nor <OutputFile> is specified then the Location directory host setting configured in the Designer will be used. Otherwise the default is the current directory.
Attributes	None
Contents	File path, note that UNC paths are supported.
Example	`<Outputpath>…</OutputPath>`

<Verbosity>

Description	Optional logging level
Attributes	None
Contents	Either set to *ON* or *OFF*

<Verbosity>

Example	`<Verbosity>ON</Verbosity>`

<Logfile>

Description	Optionally specifies log filename
Attributes	None
Contents	Filename
Example	`<Logfile>MyJobRun.log</Logfile>`

<TargetRepository>

Description	Optional repository alias
Attributes	None
Contents	Alias for a network or local repository
Example	`<TargetRepository>Doc_Repository</TargetRepository>`

<TargetRepository>

Description	Optional repository alias

<TargetRepository>

Attributes	None
Contents	Alias for a network or local repository
Example	`<TargetRepository>Doc_Repository</TargetRepository>`

<JobInfo>

Description	Production job group, at least one must be defined	
Attributes	None	
Contents	*Jobname*	one only
	JobType	one only
	OutputFile	one only
	OutputMode	one only
Example	`<JobInfo>…</JobInfo>`	

<Jobname>

Description	Mandatory production job name
Attributes	None
Contents	Name of Generate production job to be published

<Jobname>

Example	`<Jobname>//LVS/Letter02p/PDFjob</Jobname>`

<JobType>

Description	Mandatory, specifies type of job to be run	
Attributes	None	
Contents	Name of Generate production job to be published	
	Set as follows:	
	DOC1	JOB
	Active Content	AC
	EngageOne Tmplt	.TPT
Example	`<JobType>Job</JobType>`	

<OutputFile>

Description	This is optional and specifies the name of the file generated from the publishing task. If neither this element nor <OutputPath> is specified then the output file name specified in the Designer host settings will be used. Otherwise the default is the production job name.
Attributes	None
Contents	Filename

<OutputFile>

Example	`<OutputFile>MyJob.hip</OutputFile>`

<OutputMode>

Description	Mandatory, specifies the type of publishing to be carried out	
Attributes	None	
Contents	Set as follows:	
	COMPLETE	publication or Active Content design and resources are published
	DESIGN	publication or Active Content design is published
	RESOURCE	publication resources are published
	OAC	when used for: • a production job all the Active Content in the publications in the production job are published in the form of HIP files for Generate to process; • Active Content the specified Active Content is published in the form of a HIP file for Generate to process; • an EngageOne® template all the Active Content in the EngageOne template is published to a zip file suitable for the EngageOne environment.

<OutputMode>

Example	`<OutputMode>OAC</OutputMode>`

<ApplyLabel>

Description	This is optional and specifies the label to be created and assigned to the publishable object. If an existing label is assigned, a date and timestamp is appended to the label name. Note that a system label will be assigned if the contents is empty.
Attributes	None
Contents	Label name
Example	`<ApplyLabel>Label-1503</ApplyLabel>`

<BypassOutputChannel>

Description	Facilitates a batch run of EngageOne Server without the need to run any pre-processing of data. Note the following: • Device(s) must be explicitly enabled via the output device setting **Publish for EngageOne** in Designer. • Only the Structured XML Journal is supported in Publishable Active Content used with Bypass output channel processing published templates. Any other journal type entries will not appear in the output.
Attributes	None
Contents	Set as follows:
	NO — Bypass Output Channel processing is not active.

<BypassOutputChannel>

	YES	Bypass Output Channel processing is active.
Example	`<BypassOutputChannel>YES</BypassOutputChannel>`	

<LabelToPublishName>

Description	This is optional and specifies the label of the publishable object to be used. This enables earlier versions to be used. Otherwise the latest version of the object is used.
Attributes	None
Contents	Label name
Example	`<LabelToPublishName>Test-1207</LabelToPublishName>`

<Graphics>

Description	This is optional and will include all resources that are required when Active Content is used with advanced graphics features, such as in a rotated text box.
Attributes	None
Contents	Either set to ON or OFF
Example	`<<Graphics>ON</Graphics>`

Examples

The following example connects to an accessible repository and controls the processing of three production jobs, generating detailed logging information.

Note: the bolded elements in the example below are set once for the complete run.

```
<?xml version="1.0"?>
<DOC1PublishConfig>
 <UserCredential>
  <DOC1UserName>Admin</DOC1UserName>
  <DOC1Password>admin</DOC1Password>
 </UserCredential>
 <OutputPath>C:\Doc</OutputPath>
 <Verbosity>ON</Verbosity>
 <LogFile>MyJobrun.log</LogFile>
 <TargetRepository>Doc_Repository</TargetRepository>
 <JobInfo>
  <Jobname>//LVS/Letter02p/PDFjob</Jobname>
  <JobType>JOB</JobType>
  <OutputFile>test1.hip</OutputFile>
  <OutputMode>COMPLETE</OutputMode>
 </JobInfo>
 <JobInfo>
  <Jobname>//Str0311/The length of the strings</Jobname>
  <JobType>AC</JobType>
  <OutputFile>test2.zip</OutputFile>
  <OutputMode> OAC </OutputMode>
 </JobInfo>
 <JobInfo>
  <Jobname>//Str0311/Common Business Logic/The length of the
strings</Jobname>
  <JobType>AC</JobType>
  <OutputFile>StateCodes</OutputFile>
  <OutputMode>OAC</OutputMode>
 </JobInfo>
</DOC1PublishConfig>
```

The following example connects to an accessible repository and controls the bypass output processing of a EngageOne non-interactive publication via a single production job, generating detailed logging information.

```
  <UserCredential>
    <DOC1UserName>Admin</DOC1UserName>
    <DOC1Password>admin</DOC1Password>
  </UserCredential>

  <!-- To set the output folder location -->
  <OutputPath>C:\doc1gen</OutputPath>

  <!-- To set the verbosity for log file -->
  <Verbosity>ON</Verbosity>

  <!-- To set the log file name -->
  <LogFile>C:\doc1gen\Boardwalk_With_Journals_DOC1Publish.log</LogFile>


  <!-- To set the name of the repository that will be searched/used for
 publishing process. -->
  <TargetRepository>DOC1_Repository</TargetRepository>

 <!-- Publish Publication to Generate using Generate Production Job -->


 <JobInfo>
 <Jobname>//Boardwalk/Boardwalk Project/Boardwalk with Journals</Jobname>

 <JobType>TPT</JobType>
 <OutputFile>Boardwalk with Journals from Command Line</OutputFile>
 <OutputMode>COMPLETE</OutputMode>
        <BypassOutputChannel>YES</BypassOutputChannel>
 </JobInfo>

</DOC1PublishConfig>
```

Notices

http:// github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md. The source code for this software is available from http://www.newtonsoft.com/json.

ICU License - ICU 1.8.1 and later Copyright (c) 1995-2006 International Business Machines Corporation and others All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

This product contains Elasticsearch, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/**. The source code for this software is available from **https://github.com/elastic/elasticsearch**.

This product contains Elasticsearch.Net, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://apache.org/licenses/**. The source code for this software is available from **https://github.com/elastic/elasticsearch-net**.

This product contains NEST, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://apache.org/licenses/**. The source code for this software is available from **https://github.com/elastic/elasticsearch-net**.

This product contains Antlr4cs Runtime, which is licensed under the BSD-3-Clause. The license can be downloaded from **https://raw.githubusercontent.com/tunnelvisionlabs/antlr4cs/master/LICENSE.txt**. The source code for this software is available from **http://www.antlr.org**.

This product contains Log4net. The license for log4net can be downloaded from **https://logging.apache.org/log4net/license.html**. The source code for this software is available from **https://logging.apache.org/log4net/download_log4net.cgi** .

Otherwise all product names are trademarks or registered trademarks of their respective holders.

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com/us