BatchXSLT for InDesign

Exporter for InDesign Documents

Advanced Programmer's Information
Technical References
Distributed Processing
XML Elements and Attributes
InDesign Scripts Hooks
Transformer Control Through HTTP

Version 40.00.46 and newer

For

InDesign CC, CS6

Mac OS X 10.9 and newer
Windows 10

Contents

JobTickets

«BatchXSLT for InDesign»'s behavior is controlled by JobTickets.

JobTickets are plain text files located or placed into the communications folder path located at ~UserHome/BatchXSLT4InDesign_Comm/BatchXSLT. Such JobTicket files are automatically opened and processed by the main XML transformer engine «BatchXSLT».
The very first JobTicket is loaded when BatchXSLT starts and is named "autostart.jt". It is located in the applications main folder /Applications/BatchXSLT4InDesign/BatchXSLT and is responsible to set the transformer into an initialized state where it can accept further commands.
JobTicket files contain lines of keyword (parameter names) and value pairs.
Some of the parameters are important to the main engine only and other parameters will control the behavior of the XSL transformation: all parameters starting with "tp_#" - the transformer parameters. All these parameters are passed through all XML files encapsulated within a <call_parameters> element.
Below find a portion of the initializing JobTocket file "autostart.jt". You may open and read the complete file with a plain text editor.

autostart.jt

```
# comments are removed.

mode = 0

jt_triggerfile =
sourcePathName = ** Drop an XML file
xslPathName = XSL/IMXepaper.xse
outputPathName =
xslVersionName = 18
xslVersionDate = 20110506

# Parameters to pass to the XSL Style
Sheet.
# The following 'tp_X' (tp_1..tp_99)
entries ...
# Syntax: tp_X = name=value
# Example: tp_1 = myparameter1=any ...
value

tp_3 = outputMode=0
tp_5 = pageJPEGScale=330x
tp_7 = pageJPEGQuality=90
tp_9 = pageJPEGdpi=150
tp_11 = pageJPEGParams=
tp_13 = imageTYPE=1
tp_15 = imageQUALITY=90
tp_17 = imageSCALE=1.0
tp_19 = imageDPI=400,300
tp_21 = imagePARAMS=
tp_23 = catchRadius=0
tp_25 = allBoxesNoCatch=2
tp_27 = chainedBoxesNoCatch=0
...
tp_39 = overwriteCSSfiles=0
tp_41 = wwwLinkStyles=wwwLink
tp_43 = FOLDERINDEXNAME=index.html
tp_45 = DEBUG=0
tp_47 = DEBUGIMAGES=0
tp_49 = documentPDFname=
tp_51 = TABLE_CELLS_WIDTH_PRESERVE=1.0
tp_53 = TABLE_BORDER_COLLAPSE=1
tp_55 = imageCopyToOutput=0
tp_57 = ALLOW_PARACLASS_OVERRIDES=1
tp_59 = fontsizeMinimum=8
tp_61 = magnifyingGlass=10
tp_63 = outputModeCustomXSL=
tp_65 = imageEXCLUDE=excl_;555
tp_67 = imageCROP=1
tp_69 = imagesMORE=
tp_71 = TABLE_AS_BLOCK=0
...
```

JobTickets Execution Priority

Three types of JobTickets are processed by „BatchXSLT for InDesign" with different priorities.

Priority #1

"autostart.jt" is called once only when the transformer is started. This JobTicket is never deleted.

Priority #2

After the Transformer is started, the internal JobTickets handler looks for JobTickets named 'override.jt', 'override.jt1' ... 'override.jt9'.
The number in the file name extension '.jt#' again is a priority indicator. If a JobTicket named like this is found it is executed immediately. After execution has completed, this 'override.jt#' file is deleted.

Priority #3

If no JobTicket with Priority #2 could be found by the JobTickets handler, it looks for a file named 'override.que' – a queue file containing JobTicket names.
This is the current procedure used by the 'ExportCurrentDocument.jsx' to command BatchXSLT:

- a JobTicket is written with a name like '123456789override.jt' to command BatchXSLT to make the main transform
- a second JobTicket is written with a name like '123456789override.jt1' to command BatchXSLT to transform the exported XML to HTML
- the names '123456789override.jt' and '123456789override.jt1' are written into the queue file 'override.que'.

The JobTickets handler reads a JobTicket name from 'override.que' and and passes the name to the main transformer. After processing is done the JobTicket is deleted and the next name is passed to the main transformer.

Any process can write such JobTickets even when it runs on a different machine in the network but has read/write access to the folder stated at JobTicketOverrideQueuePath = path
By default, JobTicketOverrideQueuePath is set to ~UserHome/BatchXSLT4InDesign_Comm/BatchXSLT.
in the JobTicket autostart.jt

InDesign writes JobTickets

«BatchXSLT for InDesign»'s behavior, when exporting a document to XML, is controlled by JobTickets created by InDesign scripts.

The main InDesign exporter script "ExportCurrentDocument.jsx" prepares everything needed by BatchXSLT to create the new XML output:
a) a PDF of the whole document
b) a PDF and a JPEG of each page
c) an IDML file
d) a copy of all images which must be converted to JPEG
When everything is ready, two JobTicket files are written and their name is passed through a jobticket queue file to BatchXSLT. (More about this queue file later in this chapter).

The first JobTicket file below commands the transformer to create an XML file suitable for epaper applications, and to convert and link all images and page previews:

```
_____________________________________
init_previous_jobticket = 0
jobticketVersion = 10.0
applicationPurpose = InDesign_IMX_Transformer
mode = 1

jt_triggerfile =
sourcePathName = /Users/andreasimhof/Export/in/Untitled/2011/20111210/Untitled-2.idml
xslPathName = XSL/IMXepaper.xse
outputPathName = /Users/andreasimhof/Export/out/Untitled/2011/20111210/
externalProcessTimeout = 180000
continueXMLparseOnFatalError = 0
tp_3 = outputMode=0
.....
tp_5 = pageJPEGScale=400x
tp_7 = pageJPEGQuality=90
tp_9 = pageJPEGdpi=150
tp_13 = imageTYPE=1
....
```

After the above has been processed by BatchXSLT, the second JobTicket commands to transform the created XML file to plain HTML:

```
init_previous_jobticket = 0
jobticketVersion = 10.0
mode = 0

jt_triggerfile =
sourcePathName = /Users/andreasimhof/Export/out/Untitled/2011/20111210/Untitled-2.xml
xslPathName =
outputPathName =
newoutputFileNameExt = .htm
excludeSourceProcessingRunFileNameExts = .incx,_int.xml,_indb.xml,.xmi
sourceFileAction =
deleteSourceDirs = 0
nextJobTicketPath =
nextJobTicketFileName = autostart.jt
checkExtConverter = 5
```

Parameters Passed Through all Transform Steps

Parameters declared in JobTickets are passed through all transform steps.

Example: if a transform parameter like "tp_7 = pageJPEGQuality=90" is declared in a JobTicket, it can be found in the XML file as "par" element within "the "call_parameters" element:

```
<call_parameters>
...
<par name="pageJPEGQuality">90</par>
...
</call_parameters>
```

call_parameters	Contains par elements controlling the output. Parameters are defined in the main start file 'autostart.jt' or dynamically in 'overrideX.jt' files. JobTicket files are automatically detected and processed by BatchXSLT.	
par	The following parameters automatically are passed to the transform XSL by the BatchXSLT transformer engine:	
	name="XMLSRC_VERSION"	1.0
	name="XMLSRC_ENCODING"	UTF-8
	name="XMLSRC_DOCTYPE_DECLARATION"	empty for 'automatic'
	name="SYSTEM_OS_NAME"	Mac OS X or Windows
	name="SYSTEM_VM_VERSION"	Java VM version like: 1.5.0_16
	name="SYSTEM_DEFAULT_CHARSET"	like 'MacRoman' or 'windows-1252'
	name="TRANSFORM_ENGINE"	BatchXSLT VV.vv
	name="INPUT_PATH"	main source path
	name="INPUT_SUB_PATH"	subpath below INPUT_PATH
	name="INPUT_NAME"	name of IDML file
	name="OUTPUT_PATH"	path to output directory
	name="OUTPUT_NAME"	name of xml file
	name="STYLESHEET_PATH"	path to main xsl 'IMXepeper.xse'
	name="STYLESHEET_NAME"	IMXepaper.xse
	name="LOGFILE_WRITE"	1 to write messages into log file
	name="LOGFILE_PATH"	path to such log files
	name="LOGFILE_NAME"	name of a log file
	name="USER_NAME"	the user's login name
	name="USER_HOME"	the user's home folder
	name="USER_DIR"	current program's directory
	name="LOCAL_MACHINE_NAME"	the machine's name
	name="GS_VERSION"	Ghostscript version string
	name="GS_VERSION_NUM"	Ghostscript version number
	name="GS_PGM_PATH"	path to Ghostscript
	name="GS_ENVIR"	Environment for Ghostscript
	name="IM_VERSION"	ImageMagick version string
	name="IM_VERSION_NUM"	ImageMagick version number
	name="IM_PGM_PATH"	path to ImageMagick
	name="IM_ENVIR"	Environment for ImageMagick
	name="USERLICENSE_TYPE"	license type: 0 = normal, 1 = DEMO
par	The following parameters are passed to the transform XSL using 'tp_xx' statements.	
	Debugging and intermediary files	
	name="DEBUG"	1 to show DEBUG messages in console window. if set, 'INTERMEDIARY_XML_preserve' will be set automatically
	name="DEBUGIMAGES"	1 to show messages from image converters only

name="DEBUG_cssfile"	1 to show debug info for CSS file creation
name="INTERMEDIARY_XML_only"	1 to create the intermediary XML file only. default = 0
name="INTERMEDIARY_XML_preserve"	1 to not to delete the intermediary XML file. default = 0
Output type and final transform	
name="outputMode"	output view mode: 0 = flipbook 1 = as pages ePaper 2 = as article list 3 = as XML tree all output modes are created from the same main intermediary XML file
name="outputModeCustomXSL"	name of a custom transform XSL if outputMode > 3
name="OUTPUT_FEATURES"	0 = export elements and attributes of Standard version only, 2 = in addition export all PRO Elements and attributes
Page previews	
name="pageJPEGScale"	Scale for page JPEGs or fixed width in pixels. default = 330x means: 330 pixels in width.
name="pageJPEGQuality"	Quality of page JPEGs (1..100), default = 90. the lower this value, the poorer the image quality, the smaller the file size
name="pageJPEGdpi"	DPI of page JPEGs. default = 150 the lower this value, the poorer the image quality, the smaller the file size
name="pageJPEGParams"	custom parameters for page JPEG creation (see ImageMagick manual)
name="documentPDFname"	name of document PDF or empty for no document PDF
Image exports and conversion	
name="imageTYPE"	Type of image conversion: 0 = don't export images, 1 = JPEG, 2 = GIF, 3 = PNG, 4 = TIFF
name="imageQUALITY"	JPEG quality (1..100), default = 90
name="imageDPI"	JPEG DPI (default = 400,300). images are read in with 400 dpi, the output image will be 300 dpi. the lower these values, the poorer the image quality, the smaller the file size
name="imageSCALE"	JPEG scale: 1.0 = default = original size Any decimal number like 2.0, 3.32 as a scale factor or a fixed width like "300x" for 300 pixels in width a fixed height like "x250" for 250 pixels in height
name="imageDoROTATE"	rotate outout images: 0 = default = don't rotate, otherwise rotate them
name="imageCUT2imagebox"	cut images to containing image box: 1 = default = cut to image box, 0 = cut to original image
name="imagePARAMS"	JPEG custom parameters (see ImageMagick manual)
name="imageCopyToOutput"	1 to copy original images to output folder. Used also to extract text from PDF for full-text search
name="imageEXCLUDE"	semicolon separated list of patterns to exclude images. images whose file names start with one these patterns are not exported: default = excl_;555
name="imageCROP"	1 = default = crop images to box size, 0 = don't crop, use original image
name="imagesMORE"	Parameters for additional images
name="imageNOCONVERT"	1 to export imag boxes but not converted image

name="CROPBOXmode"	How to handle PDF Crop and Trim boxes. Addable flags (default = 124): 0 = convert PDFs 'as is' - don't touch them. 1 = try to replace ArtBox with CropBox if no CropBox and not TrimBox is defined. 2 = do crop PDFs to CropBox or TrimBox if defined. 4 = ad a crop command to trim to imagebox
name="PDFhasCropBox"	1 to indicate, that PDFs always have a CropBox. default = 0
name="PDFhasTrimBox"	1 to indicate, that PDFs always have a TrimBox. default = 0
name="COLORPROFILE_remove"	color profiles REMOVE command for ImageMagick default = empty to not to remove profiles
name="COLORPROFILE_TIFF"	Path to color profile to use for specific image types
name="COLORPROFILE_EPS"	
name="COLORPROFILE_AI"	
name="COLORPROFILE_PSD"	
name="COLORPROFILE_ALL"	
name="COLORPROFILE_OUTPUT"	the output color profile to use default = empty = Utilities/Profiles/ColorMatchRGB.icc
name="COLORSPACE"	ImageMagik command for output color space. default = empty = -colorspace RGB
name="MAX_IMAGENAME_LENGTH"	max length of image names, default = 31
CSS and XSL	
name="XSLCSSPath"	relative path or absolute URL to external files like XSL, JavaScripts, CSS and helpers. default = empty, which wil result in a path like ../../../XSLCSS/' depending on the nested output folder structure
name="CSSpath"	path to linked external CSS. default is the content of 'XSLCSSPath'
name="CSSname"	name of linked CSS or empty to auto-generate. if empty, a CSS is generated with a similar name like the document's name. to use a (manually) adapted CSS state the name of this CSS. existing CSS files by default are not overwritten.
name="CSSnameFinal"	the name of the final linked CSS. if "CSSname " was empty, a CSS was generated with a similar name like the document's name.
name="overwriteCSSfiles"	1 to always recreate the CSS. 0 = default = don't overwrite existing CSS
name="CSSexpanded"	1 to export enhanced CSS infos. all style attributes are exported into a commented section within a CSS declaration of a style.
Box catching and sorting	
name="catchRadius"	the box catch radius. default = 0 Zero means, that all boxes at least touching a neighbour box are captured into the same article. set to any value like for example 3 to also catch boxes into the same article even if the boxes don't directly touch each other. box catching can be turned of by the following 'allBoxesNoCatch' parameter.
name="allBoxesNoCatch"	combined flags for the following 4 catch types 0 = normal 1 = catch for all box types turned off 2 = catch for empty boxes types turned off 4 = catch for image boxes types turned off 8 = catch for chained text boxes types turned off 16= catch for text boxes types turned off
name="emptyBoxesNoCatch"	1 = empty boxes don't catch. default = 1
name="textBoxesNoCatch"	1 = text boxes don't catch. default = 0
name="chainedBoxesNoCatch"	1 = chained text boxes don't catch. default = 0
name="imageBoxesNoCatch"	1 = image boxes don't catch. default = 0
name="groupBoxesToArticles"	1 to create article groups from touching boxes. default = 1
name="sortBoxesByAreaSize"	1 to sort boxes by surrounding area box. default = 1
name="sortBoxesByYposition"	1 to sort boxes by Y position. default = 1

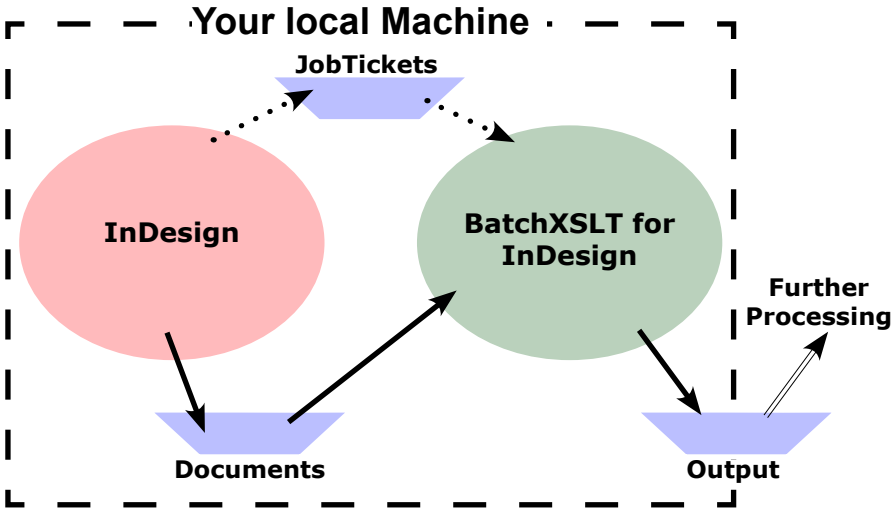
name="sortBoxYtolerance"	the tolerance for Y sorting. default = 1.5
Font handling	
name="replaceFont"	list of fonts to replace. default = empty
name="fontsizeFactor"	font size resize factor. default = empty
name="fontsizeMinimum"	smallest font size. default = 8
name="wwwLinkStyles"	list of styles acting as www links. default = 'wwwLink'
name="continuedArticleStyles"	style names linking to the next part of an article. default = empty
Tables handling	
name="TABLE_CELLS_WIDTH_PRESERVE"	non empty to resize cells by a factor
name="TABLE_BORDER_COLLAPSE"	1 = default to collaps cell borders
name="TABLE_AS_BLOCK"	1 to export tables always on a new line
Text and styles handling	
name="ALLOW_PARACLASS_OVERRIDES"	1 to allow overriding paragraph classes. 0 to suppress overridden styles of classes
name="preserverControlCharacters"	1 to preserve all control characters. dafault = 0
name="preconvertTextFlags"	see 'public.js' for a description of flags
Flip book look and behaviour	
name="addMissingPages"	1 to add missing pages in a flipbook
name="showArticleInNewWindow"	1 to open clicked article in new window
name="suppressSiteElements"	addable flags to suppress elements in output html flipbook web site 0 = default to show all flipbook site elements 1 to suppress the whole top head - page navigation 2 to suppress the bottom PDF toolbar 4 to suppress the loading progress 8 to suppress the info message text below pages add each flag with its own char like: '148'
name="suppressExportMouseOvers"	addable flags to suppress export and mouse overs in output html flipbook web site (processed by flipbook.xsl) 1 to completely suppress the mouse over function 2 to suppress all text and the mouse over text 4 to suppress all images and the mouse over images add each flag as it own char like: '14'
Layers handling	
name="includeLayers"	1 to include layers info in resulting XML
name="excludeHiddenLayers"	1 to not to export hidden layers. dafault = 0
Special information	
name="includeMetaData"	1 to include meta data info. dafault = 0
name="excludeNotes"	1 to not to export Notes. dafault = 0
name="includeDescription"	1 to include simple output description at end of output
name="XPLATFORM_NAMES"	0 or blank = do not encode filenames 1 = URI encode (%XX), 2 = default = Xplatform safe URI encode to (xXX)
name="FOLDERINDEXNAME"	non empty to create an output index.htm file. default = index.htm
name="INTERNET_AWARE_FONTNAMES"	1 to remove unsafe characters from font names. default = empty
User definable parameters	
name="userVar1"	user definable parameters
name="userVar2"	
name="userVar3"	

Distributed Processing

The Standard Installation

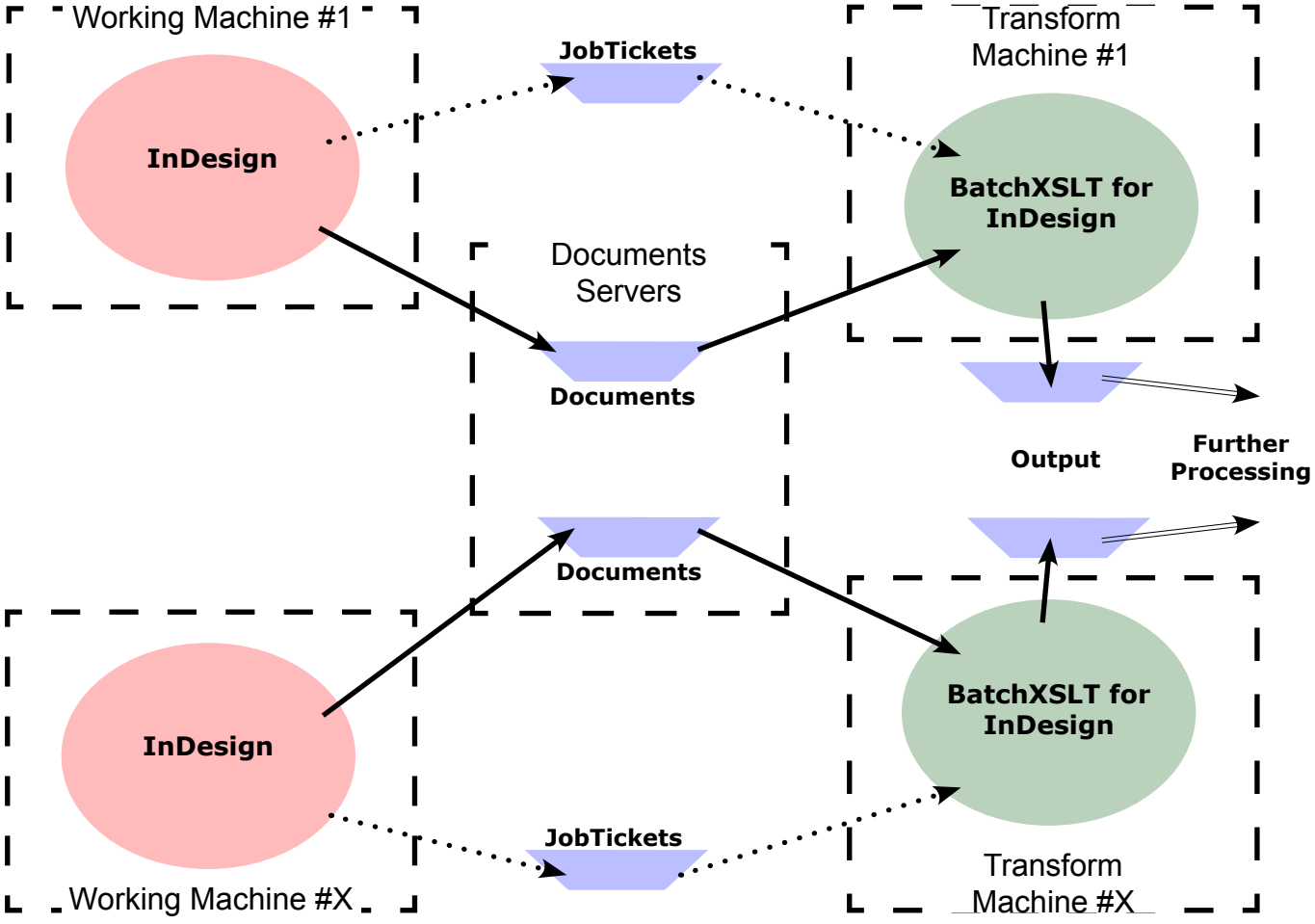
InDesign and «BatchXSLT for InDesign» are usually running on the same machine.

Most users will install «BatchXSLT for InDesign» on their layout machine beside the InDesign layout application.
The schematic workflow diagram for such a standard installation looks like this:



Advanced Installation

For 'heavy load systems', multiple licenses of «BatchXSLT for InDesign» may be installed on several, distributed transform-machines.



If you need help to create such a system, please, let us know.

License Needed for Distributed Processing

A data provider needs a «BatchXSLT» license for every customer whose InDesign data must be converted to XML output. A special "Web Master" license can be purchased where the same «BatchXSLT» transformer engine should process the data of many external customers.
The price of a "Web Master" license can be obtained by sending an email to ai@aiedv.ch.

If you need help in setting up a system matching your special needs we provide a **paid support service**.

Distributed Processing Example #1

The following example shows the step by step procedure to set up distributed processing for a widely spread requirement:
The printer or publisher produces his documents using InDesign, and wants to send the data to an external provider which creates and hosts the epaper and/or handles the exported XML data.

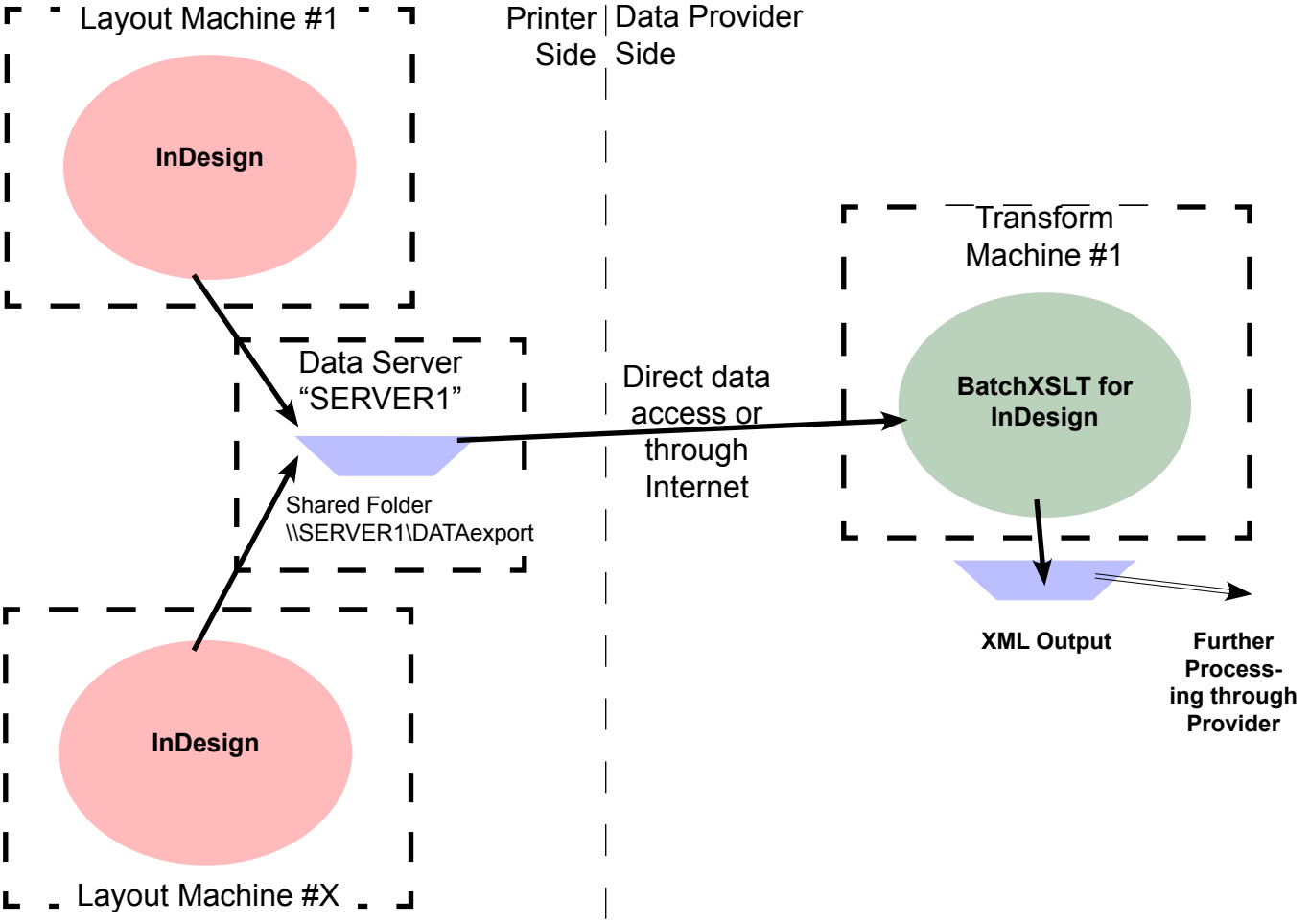
In the following example we use constant names for servers and folders. You might want to change them.

The Printer's Side

During the discussed production process, when a page is approved to be output for print, the base export for BatchXSLT for InDesign is created in a more or less automated way.
Once the production of the entire print product is complete, the base data from all InDesign pages is ready on the "Data Server" to be fetched by the Data Provider.

The Data Provider's Side

Sometime at night, the Data Provider will fetch all data from the Data Server to convert it to ePaper.



Distributed Processing Example #1 cont

Setting Up the Printer's Side

Prepare the Data Server

We assume a Windows Server.
Create a shared folder on the Data Server named **"DATAexport"**. Within this folder create the folder **"Export"** which will collect the exported data.
The access path seen from an external production machine in the network would look like this:
On a Windows machine, the UNC path looks like this:
\\SERVER1\DATAexport
The paths also can look like C:\whatever\
On a Mac OSX machine:
/Volumes/DATAexport/

Mount the volume of your Data Server on the production machine.

Install the Document Exporter Software

Install the full «BatchXSLT for InDesign» Software package on one of the production machines.
Install Java (if it is not). Java may be removed from the system after setup is complete.

Basic Export Settings

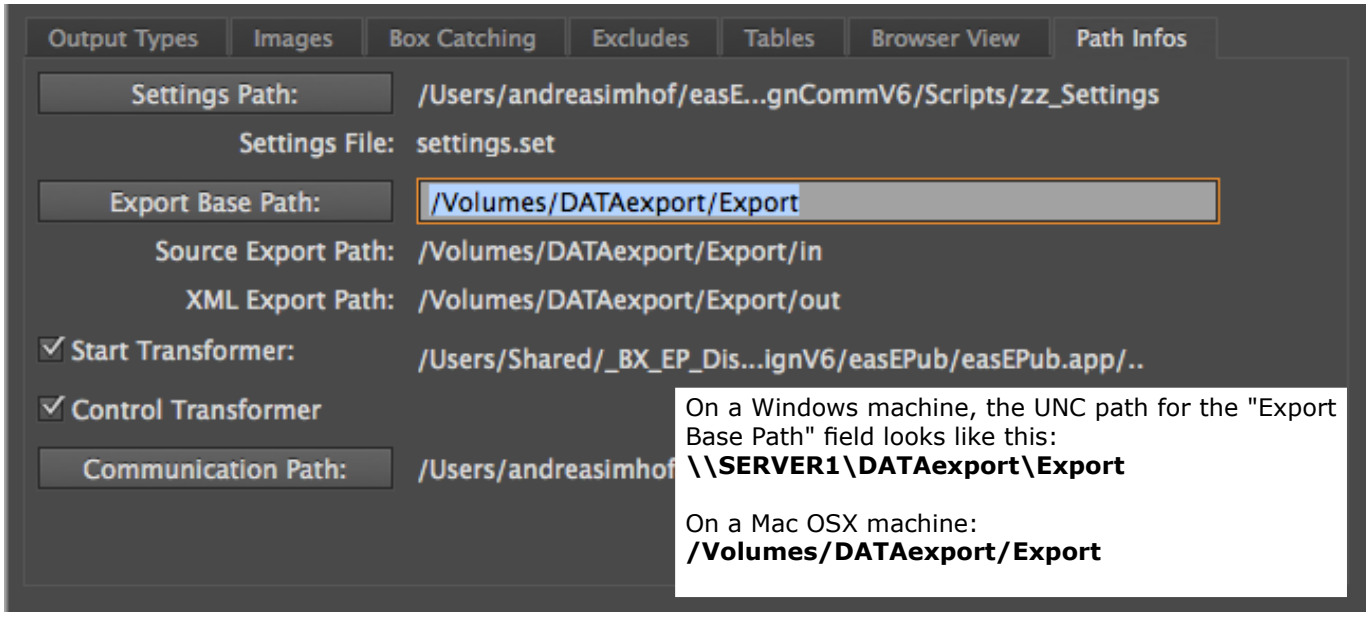
Open a document and call the Script:
"ExportCurrentDocument.jsx"
from InDesign's Scripting Panel.

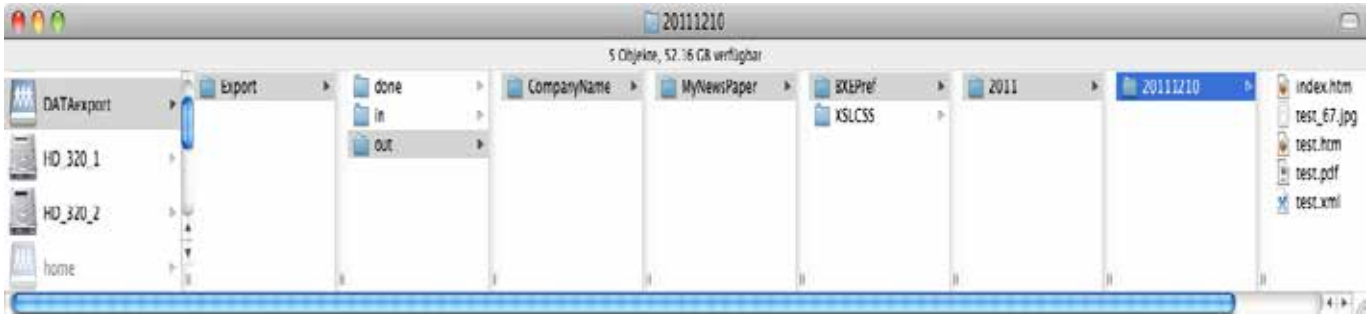
At the main exporter menu do this:

1. Fill in the fields "Company Name" and "Object Name".

2. At the "Path Infos" tab press the [Export Base Path] button and choose the "Export" folder from the Data Server.

To test if the export works fine: PRESS THE **EXPORT** BUTTON!
After a moment, you will find a full featured XML and HTML export within the Export folder:





Test the Export
The data of interest can be found at the deepest level within "Export/out/CpmpanyName/....
Open the .html file within your browser and you should see an e-Paper with one or more pages.
This is the data your Data Provider wants.

When ever you create a new export, you first should trash the 3 folders ‘done’, ‘in’ and ‘out’ within the ‘Export’ folder. Just to make sure to have the latest exported data for followimg exports.

Find out what the Data Provider needs
Let the Data Provider examine the created XML file and the content of the entire folder.
The XML file contains detailed information about the document content and text, page JPGs, page PDFs and linked images converted to JPEG

Adjust the Export Settings
Depending on what the result of such an export should be, the settings can be adjusted.

If the plain XML data and PDFs of single or double pages are needed:
In the main export menu at ‘Output Mode’ choose ‘XML Tree’, un-check ‘Page PDF’ and check ‘Document PDF’.
If no images are needed, at the ‘Images tab select ‘No’ Images.

There is an uncountable amount of possible settings. Find out what your Provider needs.
Once all the settings are clear, we can continue to automate and integrate the export process into the main production process.

Automate the Export
As all settings should be fine and we do want to see the main export dialog over and over again just to click the ‘Export’ button, we now will automate the call:
1. At InDesign’s Scripting Panel folder at the path
   `BatchXSLT4InDesign Scripts/ zz_Utilities` you can find the script ‘ExportAuto.jsx’
   Make a copy of it one folder up where ‘ExportCurrentDocument.jsx’ resides.
2. Open a document and double click the script ‘ExportAuto.jsx’. The Export should start without main dialog.
3. Examine the exported data

Disable Start and Control of the «BatchXSLT» Transformer
As we do not need the Transformer engine on the production machine, we can disable the transformer.
With an open document, call the script ‘ExportCurrentDocument.jsx’ and un-check the boxes ‘Start Transformer’ and ‘Control Transformer’ at the ‘Path Infos’ tab.
Cancel the main export dialog and again call the script ‘ExportAuto.jsx’ and then check the exported data.
As we do not transform the RAW data to XML any longer, we will find an ‘in’ folder only at the Export folder structure. From now on, the data provider’s transformer will handle the data and create the ‘done and ‘out’ folders.

Integrate the Export into the Production Process
May be, that you don’t want to manually call ‘ExportAuto.jsx’. It may automatically be called from the control scripts of your production system. It should be called at the end of the page production when it is ready to be printed: After the creation of the PDF to print and right BEFORE the document is closed. Ask your Production System vendor where to place a script call to ‘ExportAuto.jsx’.

Remove the «BatchXSLT» Transformer
If you have entered the license code at the «BatchXSLT» Transformer, ‘Transfer’ the license. Your Data Provider will need the license code to transform your data.
You then can delete the "BatchXSLT4InDesignVx" folder from the ‘Applications’ folder.

The Printer’s Side is done!

Distributed Processing Example #1 cont

Setting Up the Data Provider Side

Sometime at night, the Data Provider will fetch all data from the Printer's Data Server to convert it to ePaper or to 'what ever'.
This RAW data from the InDesign documents now has to be converted to readable and processible XML.

Let's now set up the transformer «BatchXSLT» on the Data Provider's machine.

Install the Software

We do NOT need the InDesign Scripts! We need the Transformer «BatchXSLT» only.

On a Windows machine:
Prerequisites: Install Java. On 64bit OS install BOTH Java 32 and 64 bit.

BEFORE you call the installer and if you have no InDesign installed:
On the desktop create a folder named 'Scripts' and within this folder create an other one named 'Scripts Panel'. This is needed because the installer will ask for such a folder.

Call the downloaded software installer and follow the instructions.
After installation is complete, the previously created folder on the desktop may be trashed.

On a OSX machine:
Copy the «BatchXSLT4InDesign» package folder to the Applications folder. Do not copy the Scripts folder if you have no InDesign installed.

Configure the Software

We assume, that the transform process will be started using a scheduled task sometime when one can be sure that all pages are ready.

As the transformer is not controlled by JobTickets created by an external application, we have to tell the transformer what to do in the initial loaded JobTicket 'autostart.jt.
A pre-configured JobTicket to control such a task may be found in the Applications (Programs) folder at the path:

```
BatchXSLT4InDesign/BatchXSLT/Utilities/Extras/JobTickets/
```

and, for Windows at the folder 'Win', or for OSX at the folder 'Mac'. It is named 'autostart_Once.jt'.
- Make a copy of 'autostart_Once.jt' beside the original 'autostart.jt'.
- Rename 'autostart.jt' to 'autostart_ORIG.jt'
- Rename 'autostart_Once.jt' to 'autostart.jt'

Open 'autostart.jt' with a plain text editor (best with a programmer's editor).
Locate the three lines:

```
sourcePathName = ...
outputPathName = ...
sourceFileAction = ...
```

For a Windows machine these lines look like this:

```
sourcePathName = \\SERVER1\DATAexport\Export\in\
outputPathName = \\SERVER1\DATAexport\Export\out\
sourceFileAction = \\SERVER1\DATAexport\Export\done\
```

and for OSX like this:

```
sourcePathName = /Volumes/DATAexport/Export/in/
outputPathName = /Volumes/DATAexport/Export/out/
sourceFileAction = /Volumes/DATAexport/Export/done/
```

Adjust the paths if you do not have the same server name and paths.
Save the JobTicket and manually start the transformer to test the process.
It should walk the 'in' folder and create the 'out' and 'done' folders.

If you want the transformer to quit after completion:
Open 'autostart.jt' again and locate the line:

```
mode = 1
```

and change it to

```
mode = 2
```

mode 1 means to work once after start and stay open.
mode 2 means to work once after start and then quit.

Starting the Transform Process

From a scheduled task (or manually) call the BatchXSLT-starter program.
On a Windows machine:
call BatchXSLT.bat

On a OSX machine:
call the application BatchXSLT.app

The 'in' folder provided by the printer will now be processed. The resulting XML data can be found at the 'out' folder.

BatchXSLT creates a LOG.

The log file is located at the user's HOME folder and is named 'BatchXSLT_log.txt'
Check it to see if everything went fine or if there were any ERRORs.
It can (automatically) be tested by searching the file for 3 hashes '###' followed by the ERROR text.
Especially useful when images could not be converted.

After processing is complete and the originally exported data is no longer needed, the folders 'in', 'out' and 'done' should be saved and trashed. In any case, the three folders should be removed from the 'Export' folder for the next export process can export into a clean folder structure.

HINT: sometimes it is wise to keep original export data at a safe place at least for a while...

There are many many more features included like calling an external process for pre- and/or post-processing.
Ask us, we provide help for special needs.

Exported XML Elements and Attributes

Exported XML Elements and Attributes

Complete list of elements and attributes exported by BatchXSLT for InDesign.
All units (where not marked) are in PostScript (72 pts/inch).

'PRO' marked elements/attributes are for checked "Export PRO Attributes".

BatchXSLT4InDesign XML

Element	Attributes	Function
indd_document		the all enclosing document element
call_parameters		info on how the transform was called (see below)
doctypeinfos		also see call_parameters
	xslbasepath	path to XSLCSS folder like: "../../../XSLCSS/"
	csspath	Path to CSS to link to document
	xslpath	Path to XSL for resulting view mode
	encoding	UTF-8 always
CONTENT		empty
header		the header element with headerfield childs
	type="layout"	the layout header
CONTENT		'headerfield' elements
headerfield		header fields within header tag CONTENT
	name="creationYear"	4-digits creation year like 2009
	name="creationDate"	8-digits creation date YYYYMMDD like 20090623
	name="creationTime"	creation time HH:MM:SS
	name="outputVersion"	output version
	name="filename"	this output XML file's name
	name="inputPath"	full path to export folder
	name="sourceINXfileName"	name of IDML file
	name="operatingSystem"	creator operating system
	name="vmVersion"	Java VM version
	name="transformEngine"	name and version string of transformer
	name="xsltVersion"	main transform version
	name="xsltVersionDate"	main transform version date
	name="creator"	InDesign document file infos
	name="description"	„
	name="title"	„
	name="rights"	„
	name="AuthorsPosition	„
	name="CaptionWriter"	„
	name="copyrightStatus"	„
	name="WebStatement"	„
	name="subject"	„

page			page elements
		Self	internal page ID
		page_sequence	physical page sequence in document. starts with1
		page_name	chapter page name
		pageOnSpread	number of this page on the spread
		page_side	page side: "lfth", "rgth" or "usex"
		w	page width
		h	page height
		page_left	position of left paper border from center spread
		page_right	position of right paper border from center spread
		spread	spread number of this page is located
	PRO	master	master this page is based on
	PRO	colcnt	column count
	PRO	colgutt	column gutter width
	PRO	coldir	column direction
	PRO	margtop	margin top
	PRO	margbott	margin bottom
	PRO	margleft	margin left
	PRO	margright	margin right
	PRO	numbtypesc	page numbering type shortcut
	PRO	sectprefix	section prefix
	PRO	numbtype	numbering type like "1,2,3,4..."
	PRO	autonumb	automatic page numbering [t..f]
	PRO	startpage	the page number if startnumb is ‘t’
	PRO	addsectionmarker	start with page number with section marker [t..f]
	PRO	sectmark	section marker
	CONTENT		elements: ‘pageJPEG’, ‘pagePDF’, ‘articles’, ‘group’

pageJPEG			the JPEG view of a page
		name	unencoded disk file name
		page_sequence	physical page sequence in document
		page_name	chapter page name
		sizefactor	the resize factor compared to original page size
		scale	same as sizefactor but in %
		original	name of the original size JPEG
		w	page JPEG width
		h	page JPEG height
	CONTENT		URI-encoded name of JPEG file without path

pagePDF			the PDF view of a page
		name	unencoded disk file name
		page_sequence	physical page sequence in document
		page_name	chapter page name
		fullpath	The path to the PDF during export time
	CONTENT		name of PDF file without path

articles			a container for articles on a page
		page_sequence	the page sequence
	CONTENT		element: ‘article’

article		an article constructed of one or several boxes
	idx	article index in document starting with 0
	page_sequence	physical page sequence in document
	page_name	chapter page name
	coords	the enclosing coordinates unscaled: left, top, right, bottom
	ls	left position scaled to page JPEG size
	ts	top position scaled to page JPEG size
	rs	right position scaled to page JPEG size
	bs	bottom position scaled to page JPEG size
	l	left position unscaled
	t	top position unscaled
	r	right position unscaled
	b	bottom position unscaled
CONTENT		elements: ‘boxchain’, ‘content’

boxchain		the boxes which build this article
CONTENT		element: ‘box’

box	type="empty" type="imag" type="text" type="chained_text" type="line" (PRO) type="push"	empty box image box text box chained text box line pusg button
	cont	content type of box: ‘unas’ = unassigned content (including line boxes) ‘empty’ = empty image or text box ‘text’ = a text box ‘anchored_text’ = an anchored text box ‘chained_text’ = a chained text box ‘imag’ = an image box ‘anchored_imag’ = an anchored image box
	Self	box id
	spread	spread number
	page_sequence	physical page number
	page_name	page section number
	groupid	group box id if box is in a group
	anchorid	anchor id. For anchored box non empty
	angle	angle if box is rotated
	coords	html coords rounded pixels: left,top,right,bottom
	bbox	bounding box as doubles: left,top,right,bottom
	shape	box shape points scaled/rounded to size of page jpeg
	shape_orig	box shape points
	pageJPEGScale	scale factor of output page jpeg
	y1 (px)	top rounded
	x1 (px)	left rounded
	y2 (px)	bottom rounded
	x2 (px)	right rounded
	matrix	box transform matrix of scaled box: (scale, rotation, displace)
	matrix_orig	box transform matrix unscaled original
	layerID	layer id the box is placed on
	layerName	name of this layer
	layerVis	visibility of layer: ‘t’ = visible
	label	box label text

		backgroundColor	box background color
	PRO	backgroundColorID	box background color ID pointing into the color table: extended_infos/colors/color/@Self
	PRO	backgroundColorFilt	box background color tint value
		frameWeightX	frame border weight unrounded
		frameWeight	frame border weight
		frameColor	frame border color
	PRO	frameColorID	box background color ID pointing into the color table: extended_infos/colors/color/@Self
	PRO	frameColorFilt	filter / shade value -1...100, (-1 is 100)
	PRO	frameStyleName	name of frame style
	PRO	frameStyleHTML	html style frame: solid, double, dotted, dashed
	PRO	frameGapColor	frame gap color RGB
	PRO	frameGapColorID	frame gap color ID
	PRO	frameGapColorTint	frame gap color tint / filter / shade value -1...100, (-1 is 100)
	line PRO	leftFrameEnd	line ending left: attribute not present if no line ending ‘brah’ =Bar ‘ssah’ = SquareSolid ‘sqah’ = Square ‘csah’ = CircleSolid ‘crah’ = Circle ‘cvah’ = Curved ‘barb’ = Barbed ‘twah’ = TriangleWide ‘trin’ = Triangle ‘swah’ = SimpleWide ‘simp’ = Simple
	line PRO	rightFrameEnd	line ending right (see line ending left)
box @type=’text’ cont.			additional attributes for text boxes
		insetTopR	text inset top rounded
		insetLeftR	text inset left rounded
		insetBottomR	text inset bottom rounded
		insetRightR	text inset right rounded
		insetTop	text inset top as double
		insetLeft	text inset left as double
		insetBottom	text inset bottom as double
		insetRight	text inset right as double
		chainidx	0-based index in box in a chain
		firstflowid	id of first box in a chain
		lastflowid	id of last box in a chain
		textflowid	id of this box in a chain points to it’s content: content/@id
		previousflowid	id of previous box in chain or ‘n’ if none
		nextflowid	id of next box in chain or ‘n’ if none
		colcnt	number of columns in textbox
		colwidth	width of a column
		colgutter	width of column gutter
		vjust	vertical justification of text within box: top = Align Top middle = Align Center bottom = Align Bottom vjust = Justify Vertically
		paraspacelimit	maximum paragraph space
	CONTENT		empty

box @type= 'anchored_text' or 'anchored_imag' cont.		additional attributes for anchored text and image boxes
	PRO ao_NoManualPos	prevent manual positioning: [t | f]
	PRO ao_Position	positioning: AOPi = inline, AOPa = Above Line, AoPa = custom
	PRO ao_yOffset	inline y offset or 'above line: space after'
	PRO ao_yOffsetAfter	above line: space before
	PRO ao_xOffset	custom x offset
	PRO ao_horizAlign	left, cent, rght, txal (also see ao_relSpine)
	PRO ao_relSpine	relative to spine [t | f]
	PRO ao_refPoint	reference Point of Anchored Object: ANtl = top left ANtc = top center ANtr = top right ANlc = left center ANbl = bottom left ANbc = bottom center ANtr = top right ANrc = right center ANct = center
	PRO ao_xRelTo	anchored position X relative to: txtf = text frame APan = anchor marker APco = column edge APmg = page margin APpg = page edge
	PRO ao_yRelTo	anchored position Y relative to: AVba = baseline AVxc = Cap Height AVbl = Top of Leading APco = column edge txtf = Text Frame APmg = page margin APpg = page edge
	PRO ao_keepColBound	keep with top/bottom coulumn boundaries [t | f]
box @type='push'		additional attributes for push button boxes
	buttonname	name of this button
	onaction	the action event
button	all box attributes as well as...	... the content and action of buttons
	type	push
	cont	push
	buttonname	name of this button
	onaction	the action event
	moviewidth	width of the movie frame
	movieheight	height of the movie frame
CONTENT		action, buttonstate
button/action @type="Movie"		push button actions for movies
	event	event actions: muev=onMouseUp mdev=onClick meev=onMouseOver mxev=onMouseOut ofev=onFocus obev=onBlur
	operation	what to do on event: play, stop, paus, resu

	movieID	ID of the movie
	name	name of the movie file
	path	path to the movie file
CONTENT		empty
button/action @type="Sound"		push button actions for sound
	event	event actions: muev=onMouseUp mdev=onClick meev=onMouseOver mxev=onMouseOut ofev=onFocus obev=onBlur
	operation	what to do on event: play, stop, paus, resu
	soundID	ID of the sound
	name	name of the sound file
	path	path to the sound file
CONTENT		empty
button/ buttonstate		push button states
	id	the id of this state
	name	‘Up’ or ‘Rollover’
CONTENT		content
content		the content of article / box
	type	‘text’ or ‘image’
	Self	a reference
	idx	content id (usually same as Self) points to box/@ textflowid
	anchor_id	non empty if containing box is anchored
	exist	for image boxes set to 1 if image exists
	label	box label text
	insetTopR	box inset real values
	insetLeftR	
	insetBottomR	
	insetRightR	
	insetTop	box inset rounded pixel values
	insetLeft	
	insetBottom	
	insetRight	
	backgroundColor	the box background color if any
image	angle	angle of image
image	scaleX	image scale X
image	scaleY	image scale Y
PRO image	displaceX	image displace X within image box
PRO image	displaceY	image displace Y within image box
CONTENT		element ‘div’ for text boxes
		‘img’, ‘img2’ for image boxes
div	if content @type=’text’	paragraph attributes
	class	CSS class name
	style="margin-left:#px;"	combinable HTM style attributes: left margin

	style="text-indent:#px;"	first line indent
	style="margin-right:#px;"	right margin
	style="margin-top:#px;"	top margin
	style="margin-bottom:#px;"	bottom margin
	style="text-align:xxx"	where xxx: left, center, right, justify
PRO	pstyID	paragraph style id. points to //extended_infos/stylesheets/paragraphs/@Self
PRO	cstyID	character style id. points to //extended_infos/stylesheets/characters/@Self
PRO	fontSize	font real size in pt
PRO	fontColorID	font color ID (points into <colors> section)
PRO	fontColorFilt	font color tint
PRO	fontStyle	font style: Roman, Regular, Italic..., 1,2...
PRO	fontSkew	font skew/angle (false italic) in degrees
PRO	underline	underline = 't' or 'f' for true or false
PRO	capsMode	caps mode: 'alcp' = All Caps, 'smcp' = Small Caps
PRO	charPos	position: 'spsc ' = Superscript, 'sbsc' = Subscript
PRO	strike	strikethrough: 't' = strike
PRO	fontFamily	font family like 'Times', 'Verdana', 'Mignon'...
PRO	fontName	font name incl. style: 'Times Roman', 'Mignon Italic'...
PRO	fontNamePS	postscript font name: 'Times-Roman', 'Minion-Italic '...
PRO	baseShift	base line shift value
PRO	scaleHoriz	horizontal font scale value in percent
PRO	scaleVert	vertical font scale value in percent
PRO	leading	line leading in pt
PRO	kerning	kerning in 1/1000th em
PRO	nobreak	non breaking text section = 't'
PRO	ligatures	ligatures: 't' if ligatures should be used, 'f' for no ligatures
PRO	tracking	tracking in 1/1000th em
PRO	margtop	margin top
PRO	margleft	margin left
PRO	margbott	margin bottom
PRO	margright	margin right
PRO	firstLineIndent	first line text indent
PRO	conditionID	if conditional text, this ID points into element: extended_infos/textconditions/condition[@Self=conditionID]
CONTENT		elements: 'span', 'br', text
PRO paraopts		paragraph options containing elements: parasets, pararules, paralist, paratabs
PRO parasets	paraStyleID, charStyleID, insetLeft, insetRight, insetFirstline, insetLastLineRight, lineLeading, dropCapsChars, dropCapLines, hyph, adjustBaseLine, aligneToSpine	general paragraph settings
PRO pararules	colorID, size, tint, offset, leftIndent, rightIndent, overprint, on, keepInFrame	paragraph rules elements: paraRuleAbove, paraRuleBelow
PRO paralist	type, styleID, align, textAfter, numberStyle, contStyle, numberFormat, level	paragraph list definistions
PRO paratabs	align, alignChar, leaderChar, pos	tab attributes for each set tab
span		text attributes change
	class	CSS class name of character style

		style="font-family:'Arial',..."	font name
		style="font-family:...,'Arial Bold Italic';"	Mac style name
		style="font-style:italic; "	font style (normal, italic...)
		style="font-weight:bold;"	font weight (normal, bold...)
		style="color:#ff0000; "	font color RGB
		style="font-size:xxpt; "	size in pt
		style="vertical-align:super;"	supper script
		style="vertical-align:sub;"	sub script
	PRO	cstyID	character style id. points to //extended_infos/stylesheets/characters/@Self
	PRO	fontStyle	font style: Roman, Regular, Italic..., 1,2...
	PRO	fontSkew	font skew/angle (false italic) in degrees
	PRO	underline	underline = 't' or 'f' for true or false
	PRO	capsMode	caps mode: 'alcp' = All Caps, 'smcp ' = Small Caps
	PRO	charPos	position: 'spsc ' = Superscript, 'sbsc ' = Subscript
	PRO	strike	strikethrough: 't' = strike
	PRO	fontFamily	font family like 'Times', 'Verdana', 'Mignon'...
	PRO	fontName	font name incl. style: 'Times Roman', 'Mignon Italic'...
	PRO	fontNamePS	postscript font name: 'Times-Roman', 'Minion-Italic '...
	PRO	baseShift	base line shift value
	PRO	scaleHoriz	horizontal font scale value in percent
	PRO	scaleVert	vertical font scale value in percent
	PRO	leading	line leading in pt
	PRO	kerning	kerning in 1/1000th em
	PRO	nobreak	non breaking text section = 't'
	PRO	ligatures	ligatures: 't' if ligatures should be used, 'f' for no ligatures
	PRO	tracking	tracking in 1/1000th em
	CONTENT		elements: 'span', 'br', text
img, img2			image and 2nd image element
		src	name of converted JPEG
		alt	original image name
	CONTENT		empty
table			InDesign tables tagged linke HTML
		class	table css class name
		style="border-left-style:html style"	solid, double, dashed, dotted
		style="border-right-style:html style"	
		style="border-top-style:html style"	
		style="border-bottom-style:html style"	
		style="border-left-width:rounded px"	up-rounded to full pixels
		style="border-right-width:rounded px"	
		style="border-top-width:rounded px"	
		style="border-bottom-width:rounded px"	
		style="border-left-color:#rgb"	RGB color
		style="border-right-color:#rgb"	
		style="border-top-color:#rgb"	
		style="border-bottom-color:#rgb"	

	style="padding-left"	Text inset left
	style="padding-top"	
	style="padding-right"	
	style="padding-bottom"	
PRO	Self	this table's ID
PRO	anchor_id	anchor ID
PRO	tablestyleID	table style ID applied to this table
PRO	rows	number of rows in table
PRO	cols	number of columns in table
PRO	headerrows	number of head rows in table
PRO	footerrows	number of foot rows in table
PRO	rowfillFirstcolorID	row fill first color ID
PRO	rowfillFirstcolorTint	row fill first color tint
PRO	rowfillFirstcolorCount	row fill first color count
PRO	rowfillStartcolorSkip	how many rows to skip at top until color fill
PRO	rowfillSecondcolorID	row fill second color ID
PRO	rowfillSecondcolorTint	row fill second color tint
PRO	rowfillSecondcolorCount	row fill second color count
PRO	rowfillEndcolorSkip	how many rows at bottom to not to fill with colo
PRO	borderLeftStyle	name of left border style: solid, dashed....
PRO	borderLeftStyleID	left border style ID
PRO	borderLeftWidth	left border width
PRO	borderLeftColorID	left border color ID
PRO	borderLeftColorTint	left border color tint
PRO	borderLeftGapColorID	left border gap color ID
PRO	borderLeftGapColorTint	left border gap color tint
PRO	borderLeftColor	left border RGB color
PRO	borderTopStyle	name of top border style: solid, dashed....
PRO	borderTopStyleID	top border style ID
PRO	borderTopWidth	top border width
PRO	borderTopColorID	top border color ID
PRO	borderTopColorTint	top border color tint
PRO	borderTopGapColorID	top border gap color ID
PRO	borderTopGapColorTint	top border gap color tint
PRO	borderTopColor	top border RGB color
PRO	borderRightStyle	name of right border style: solid, dashed....
PRO	borderRightStyleID	right border style ID
PRO	borderRightWidth	right border width
PRO	borderRightColorID	right border color ID
PRO	borderRightColorTint	right border color tint
PRO	borderRightGapColorID	right border gap color ID
PRO	borderRightGapColorTint	right border gap color tint
PRO	borderRightColor	right border RGB color
PRO	borderBottomStyle	name of bottom border style: solid, dashed....
PRO	borderBottomStyleID	bottom border style ID
PRO	borderBottomWidth	bottom border width
PRO	borderBottomColorID	bottom border color ID
PRO	borderBottomColorTint	bottom border color tint
PRO	borderBottomGapColorID	bottom border gap color ID
PRO	borderBottomGapColorTint	bottom border gap color tint
PRO	borderBottomColor	bottom border RGB color

PRO - tablesettings		additional table setup infos element
PRO - tablesettings/ rowheights/row	rownum	row number
	minheight	minimum height (row height at export time)
PRO - tablesettings/ rowheights/total	height	total height of all rows (table height)
PRO - tablesettings/ colwidths/col	colnum	column number
	width	column width
PRO - tablesettings/ colwidths/total	width	total width of all columns (table width)
table/tr		table row
		HTML style attributes:
	style="border-left-style:html style"	solid, double, dashed, dotted
	style="border-right-style:html style"	
	style="border-top-style:html style"	
	style="border-bottom-style:html style"	
	style="border-left-width:rounded px"	up-rounded to full pixels
	style="border-right-width:rounded px"	
	style="border-top-width:rounded px"	
	style="border-bottom-width:rounded px"	
	style="border-left-color:#rgb"	RGB color
	style="border-right-color:#rgb"	
	style="border-top-color:#rgb"	
	style="border-bottom-color:#rgb"	
PRO	height	row height
PRO	backgroundColor	background color RGB
table/tr/td		table cell
	colwidth	width of cell
	colspan	number of columns to span
	rowspan	number of rows to span
	valign	vertical justification of text within cell: top = Align Top middle = Align Center bottom = Align Bottom vjust = Justify Vertically
		HTML style attributes:
	style="border-left-style:html style"	border styles: solid, double, dashed, dotted
	style="border-right-style:html style"	
	style="border-top-style:html style"	
	style="border-bottom-style:html style"	
	style="border-left-width:rounded px"	border width up-rounded to full pixels
	style="border-right-width:rounded px"	
	style="border-top-width:rounded px"	
	style="border-bottom-width:rounded px"	
	style="border-left-color:#rgb"	border RGB color
	style="border-right-color:#rgb"	
	style="border-top-color:#rgb"	
	style="border-bottom-color:#rgb"	
	style="background-color:#rgb"	cell color
	style="padding-left"	text inset left

	style="padding-top"	text inset top
	style="padding-right"	text inset right
	style="padding-bottom"	text inset bottom
PRO	textInsetLeft	cell text insets
PRO	textInsetTop	
PRO	textInsetRight	
PRO	textInsetBottom	
PRO	borderLeftWidthX	left cell border as unrounded double value
PRO	borderLeftWidth	left cell border as uprounded full pixels
PRO	borderLeftColorID	left cell border color ID
PRO	borderLeftStyleID	left cell border stroke style ID
PRO	borderLeftGapColor	left cell border gap color RGB
PRO	borderLeftGapColorID	left cell border gap color ID
PRO	borderLeftGapColorTint	left cell border gap color tint
PRO	borderTopWidthX	top cell border as unrounded double value
PRO	borderTopWidth	top cell border as uprounded full pixels
PRO	borderTopColorID	top cell border color ID
PRO	borderTopStyleID	top cell border stroke style ID
PRO	borderTopGapColor	top cell border gap color RGB
PRO	borderTopGapColorID	top cell border gap color ID
PRO	borderTopGapColorTint	top cell border gap color tint
PRO	borderRightWidthX	right cell border as unrounded double value
PRO	borderRightWidth	right cell border as uprounded full pixels
PRO	borderRightColorID	right cell border color ID
PRO	borderRightStyleID	right cell border stroke style ID
PRO	borderRightGapColor	right cell border gap color RGB
PRO	borderRightGapColorID	right cell border gap color ID
PRO	borderRightGapColorTint	right cell border gap color tint
PRO	borderBottomWidthX	bottom cell border as unrounded double value
PRO	borderBottomWidth	bottom cell border as uprounded full pixels
PRO	borderBottomColorID	bottom cell border color ID
PRO	borderBottomStyleID	bottom cell border stroke style ID
PRO	borderBottomGapColor	bottom cell border gap color RGB
PRO	borderBottomGapColorID	bottom cell border gap color ID
PRO	borderBottomGapColorTint	bottom cell border gap color tint
CONTENT		element: ‘div’

Note		hidden notes
	user	name of user who created this note
	created	cration date and time
	modified	modification date and time
	stof	id
Note/content		
	type="Note"	type of content
Note/content/div		
	class	CSS class name
Note/content/div/.		the note’s text

footnote		
footnote/footnote_num/.		footnote number
footnote/div/.		the footnote text

documentPDF		the PDF of the whole document
	name	unencoded disk file name
	fullpath	the path to the PDF
CONTENT		URI encoded name of PDF file without path

text variables are not marked but directly inserted into the text

layers		layers descriptions
	name	layer name
	visible	‘t’ for visible
	locked	‘t’ for locked
	print	‘t’ to print this layer content
	showguides	‘t’ to show guides on this layer
	lockguides	‘t’ to lock guides on this layer
	supptextwrap	suppress text wrap when layer is hidden
CONTENT		empty

metadata		XML content with meta data
		image references

```
<rdf:li rdf:parseType="Resource">
<stMfs:linkForm>ReferenceStream</stMfs:linkForm>
<stMfs:reference rdf:parseType="Resource">
 <stRef:lastURL>file:///Users/andreasimhof/BatchXSLT4InDesign/BatchXSLT/_TestData/in/28_03chirac.eps</
stRef:lastURL>
</stMfs:reference>
<xmpMM:placedXResolution>72.00</xmpMM:placedXResolution>
<xmpMM:placedYResolution>72.00</xmpMM:placedYResolution>
<xmpMM:placedResolutionUnit>Inches</xmpMM:placedResolutionUnit>
</rdf:li>
```

color references

```
<rdf:li rdf:parseType="Resource">
<xmpG:swatchName>C=100 M=0 Y=0 K=0</xmpG:swatchName>
<xmpG:mode>CMYK</xmpG:mode>
<xmpG:type>Process</xmpG:type>
<xmpG:cyan>100</xmpG:cyan>
<xmpG:magenta>0</xmpG:magenta>
<xmpG:yellow>0</xmpG:yellow>
<xmpG:black>0</xmpG:black>
</rdf:li>
```

font references

```
<rdf:li rdf:parseType="Resource">
<stFnt:fontName>Times-Italic</stFnt:fontName>
<stFnt:fontFamily>Times</stFnt:fontFamily>
<stFnt:fontFace>Italic</stFnt:fontFace>
<stFnt:fontType>TrueType</stFnt:fontType>
<stFnt:versionString>Times-Italic5.0d10e1</stFnt:versionString>
<stFnt:composite>false</stFnt:composite>
<stFnt:fontFileName>Times.dfont</stFnt:fontFileName>
</rdf:li>
```

Extended information elements PRO

All following elements are exported only when you have checked “Export PRO Attributes”.

extended_infos	contained sub-elements	The extended information element block
	colors	detailed information on a document’s colors
	fonts	detailed information on a document’s fonts
	stylesheets	detailed information on a document’s stylesheets
	textconditions	detailed information on a document’s text conditions
colors	contained sub-elements	
	color	information for: - color type - color space - tints - gradients
fonts	contained sub-elements	
	font-family	The family containing the font
	font	- font name - postscript name - font style - font version and more
stylesheets	contained sub-elements	contained information
	paragraphs	- class name - InDesign name - style id it is based on - font name - font color - font size - insets - hyphenation info - rules and more
	characters	- class name - InDesign name - style id it is based on - font name - font color - font size and more
	tables	- background color - row and column fills and strokes and more
	objects	- background color - borders - position and more
	strokes	available stroke types
textconditions	contained sub-elements	contained information
	condition	an element describing the text condition Self="ID of this condition" pnam="Condition name" pvis="[t][f]" visibility true or false pCTm="highlight mode" ctTc="highlight color"

Extending Exporter Scripts with Hooks

The following InDesign exporter scripts can be extended through script hooks:
– ExportCurrentDocument.jsx
– ftpToWebArchive.jsx
– pushToWebDatabase.jsx

Script Hooks are written in InDesign's scripting language "Extend Script".
Such hook scripts are placed into sub folders within the exporters scripts folder "BatchXSLT4InDesign ScriptsVx" which is contained in the main InDesign "Script Panel" folder.

At several points during the run of an exporter script, at a named hook-event, the defined hook scripts are called using InDesign's script call doScript(). As this doScript() can not pass parameters or return values to/from the called script, we have to use a global accessible object named 'hooks'.

As many different publications can be exported using different settings, a called hook script must be able to determine if it should do something or not: It must be able to get the current loaded JobTicket file name which should be unique for a certain publication export.

The parameters passed to a called hook script is bound to the current settings file name in a global variable named 'hooks' and looks like this:

```
var hooks = {
        "settingsFileName1.set" : {  /* the call parameters object */ },
        "settingsFileName2.set" : {  /* the call parameters object */ },
        ...
        "settingsFileName3.set" : {  /* the call parameters object */ },
};
```

Where the "the call parameters object" is:

```
{"callerscriptname":callerscriptname, "hook":whichhook,"args":args, "retval":0}
```

where:
property "callerscriptname" is the name of the calling script like "ExportCurrentDocument.jsx"
property "whichhook" is the name of the current hook event like "afterInit"
property "args" is null or the parameters object specific to a hook event like "afterShowMainDialog", {"go":go}
property "retval" is the return value a called hook can set like 0 for ok or -1 for abort

A called hook script would access this object like :
hooks["settingsFileName1.set"]
and can read the properties specific to a setting and and hook event.

As several export runs with different settings can be done in a single main export run a hook script would test if it should act on certain current run or not.
Example:

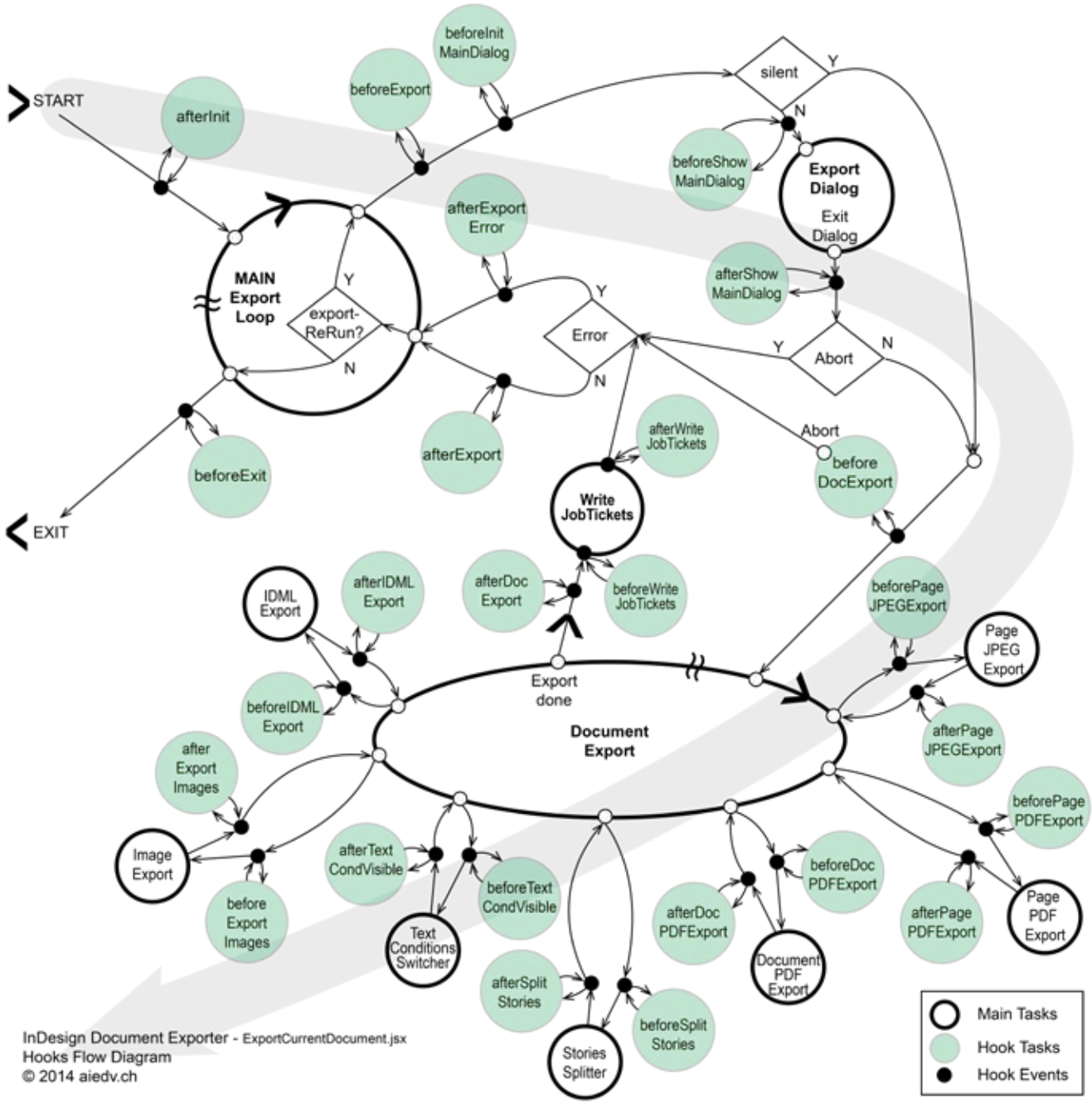
```
if (hooks["settingsFileName1.set"] || hooks["settingsFileName2.set"]) {
    if (docExportRun == 1) {
        // we only act in the very first document export run
        // get the passed args if needed
    }
}
```

A called hook script can access and set all globally defined variables of the calling main script like
"docExportRun " is 1 for the first export, then 2, 3 ...
"exportReRun" can be set to 'true' to redo a next export after having loaded a new JobTicket. Default is 'false'.

A programmer would have to dive into the main scripts to learn which variables can be accessed.

"ExportCurrentDocument.jsx" Hooks Flow Diagram

The following diagram shows the hooks event calls during an entire export process.
The green circles are the hooks (if any available) called on a certain program state.



"ExportCurrentDocument.jsx" hook folder and events

The main hooks folder checked by this script is: zz_Plugins/**hooks**/

The hook scripts folder structure would look like this:
zz_Plugins/**hooks**/

- beforeDocExport/
  - myCustomHookScript.jsx
  - anotherScript.jsx
  - whatevername.jsx
- beforeDocPDFExport/
  - myownPDFsettings.jsx
- afterExport/
  - reDoNextExport.jsx

The hook scripts are called in the sequence like listed below.

Hook event names (folder names)	Call point / Description	Usage / Example
afterInit	After all main initialization is done.	args : null Set new access paths, load a JobTicket matching the document name. Ex. actions: modify settings matching a publications export requirements
beforeExport	Before every document export is initialized (also before the main menu)	args : null If the retval is set to -1, this export is skipped - next hook is 'afterExport' Ex. actions: skip certain documents which should not be exported
beforeInitMainDialog	Before the main dialog is initialized	args : null If the retval is set to -1, the dialog is silently skipped
beforeShowMainDialog	After the main dialog is initialized but before it is shown	args : null
afterShowMainDialog	after the main dialog has been closed	args : {"go":go} Ex. actions: check the 'go' property to determine if the user has cancelled the export or if an export will stat
beforeDocExport	before the actual Document Export Task starts	args : null Ex. actions: turn on/off conditional text, set text variables, check for overflow text
beforePageJPEGExport	before the page JPEG preview image is created	args : { "theFilePath": theFilePath, "jpegExportPreferences": jpegExportPreferences } If the retval is set to -1, the page JPEG creation is skipped Ex. actions: modify the JPEG export options
afterPageJPEGExport	after a pge JPEG preview image has been created	args : { "theFilePath": theFilePath } Ex. actions: check if the JPEG file exists, create another page JPEG with different settings
beforePagePDFExport	before the page PDF preview image is created	args : { "theFilePath": theFilePath, "usepagepdfpresets": usepagepdfpresets } If the retval is set to -1, the page PDF creation is skipped Ex. actions: modify the PDF export options
afterPagePDFExport	after a page PDF has been created	args : { "theFilePath": theFilePath } Ex. actions: check if the PDF file exists, create another page PDF with different settings
beforeDocPDFExport	before the document PDF is created	args : { "theFilePath": theFilePath, "usepagepdfpresets": usepagepdfpresets } If the retval is set to -1, the document PDF creation is skipped Ex. actions: modify the PDF export options
afterDocPDFExport	after a document PDF has been created	args : { "theFilePath": theFilePath } Ex. actions: check if the PDF file exists, create another PDF with different settings
beforeSplitStories	right before the check if all conditional text should be turned on	args : null If the retval is set to -1, this is skipped Ex. actions: switch the type of story splitting
afterSplitStories	after splitting is complete	args : null Ex. actions: restore the original story splitting flag

beforeTextCondVisible	right before the check if all conditional text should be turned on	args : null If the retval is set to -1, this is skipped Ex. actions: check/verify all conditional text
afterTextCondVisible	after splitting is complete	args : null Ex. actions: restore the original text conditions
beforeExportImages	before copying/preparing the original images	args : null If the retval is set to -1, image export is skipped Ex. actions: check if all images exist or do something
afterExportImages	after all original images have bee prepared	args : null
beforeIDMLExport	before the entire document ius exported to IDML	args : null Ex. actions: suppress some stuff in the XML
afterIDMLExport	after IDML export completed	args : null
afterDocExport	after an export of a document has completed	args : null
beforeWriteJobTickets	before the JobTickets for the transformer will be written	args : { "commpath":path } where 'path' is the folder path to the communications folder Ex. actions:
afterWriteJobTickets	after the JobTickets have been written	args : { "commpath":path } where 'path' is the folder path to the communications folder Ex. actions: write one or more JobTickets for the transformer to automatically create any number of dedicated XML output
afterExport	After a document export successfully is completed	args: null Ex. actions: set and load a new JobTicket, set the variable 'exportReRun' = true; to loop to a next export run. Hook 'beforeExport' will be the next hook event.
afterExportError	After a document export terminated with an error	args: {"success":success} where: success = errorcode -1 = aborted or other error codes Ex. actions: notify user, write to log file, check the error, correct it and redo the export by setting the variable 'exportReRun' = true
beforeExit	Before the exporter script terminates	args: {"success":success} Ex. actions: notify user, write to log file

"pushToWebDatabase.jsx" hook folder and events

The main hooks folder checked by this script is: zz_Plugins/**hooksDB**/

Various hooks can be set in the same manner as for "ExportCurrentDocument.jsx". See hint below..

"ftpToWebArchive.jsx" hook folder and events

The main hooks folder checked by this script is: zz_Plugins/**hooksFTP**/

Various hooks can be set in the same manner as for "ExportCurrentDocument.jsx". See hint below..

To: "See hint below":

As you are probably at least a very skilled programer or InDesign scripter, you will not have any problems to dive into the source code, check the hook points and make your own hook scripts.

Transformer Control Through HTTP

How To Configure Transformer Control Through HTTP

BatchXSLT for InDesign is not just an XML (and XHTML) Transformer and image converter, it is also a HTTP server. Not to drive a web site, but to accept commands from any client which is able to send GET or POST requests over HTTP.
As an example: you can start any XML transformation from a web browser.

The HTTP server is not started by default when the transformer starts. There are some declarations within 'autostart.jt' to setup the so called httpCommander.

A fairly heavy web app is included within the folder 'CommanderHTTP' (contained in the Utilities folder).
We have designed it a long time ago to convert old Newspapers to flipping pages ePaper. These Newspapers were scannd/OCR with OmniPage.

Move (or copy) this 'CommanderHTTP' folder one level up beside BatchXSLT.app and autostart.jt.
'CommanderHTTP' contains a folder named:
'zz Content to BatchXSLT4InDesignComm-BatchXSLT beside comm folder'
Copy its content to the folder ~/BatchXSLT4InDesignComm/BatchXSLT/

Then, add these declarations to the autostart.jt file to activate http mode as a starting point:

```
# ---- httpCommander stuff ----
# set httpCommander_Active=1 to start the HTTP Command Server, 0 to not to start it
httpCommander_Active=1
# Defines for the remote http commander ( default: httpCommander_Port=8180 )
httpCommander_Port=8180
# the http server context root path. default = /
# localhost:8180/
httpCommander_ContextPath=/
# the servers name. default = httpCommandServer
httpCommander_Name=httpCommandServer
# the document root. default=CommanderHTTP/
# this folder must exist on same level as BatchXSLT.app
# you may also give a full path to any folder on a disk (start path with a slash /  )
httpCommander_DocumentRoot=CommanderHTTP/
# Path/name to file to send on connection
httpCommander_HelloFile=OmniPage/omnipage.htm
# the file type of hello file
httpCommander_HelloFileType=text/html; charset=UTF-8
# Path/name to file to send to log in
httpCommander_LoginFile=login.htm
# the file type of login file
httpCommander_LoginFileType=text/html; charset=UTF-8

# set httpCommander_secure=1 to force a logged in user with user name and password.
# 0 for no login required
httpCommander_secure=0
# a user name and password: user,password
httpCommander_userpass=itsme,4711
httpCommander_userpass1=itsyou,0815
httpCommander_userpass2=itshe,0817

# Turn debug mode on or off: 1 to write debug information into the transformer log file
httpCommander_DEBUG=0

# where delivered jobtickets are stored in the application software package
jobticketsPackagePath=CommanderHTTP/default/OmniPage/
# where working jobtickets should be stored
jobticketsWorkingPath=~/BatchXSLT4InDesignComm/BatchXSLT/jobtickets/
# ---- END httpCommander stuff ----
```

To start the http commander, set
httpCommander_Active=1

Start BatchXSLT.app
When the transformer starts, it will show the message line:
httpCommandServer is listening on port 8180
and is ready to accept commands over http.

You now may use your browser to test the httpCommander.
Enter a command at your browser's address line.

Command syntax:
IPaddress:port/?command
or
IPaddress:port/?command=parameter1
or
IPaddress:port/?command=parameter1,parameter2

Examples:
Assume, that the transformer is running on a machine with IP address 192.168.1.99.

Enter the following URL into the browser's address line:
192.168.1.99:8180
If the httpCommander_HelloFile is set in the autostart.jt file, you will see the web app.
If this parameter is not set, the transformer silently will close the connection as no command is given. But.....

To ask for the status of the transformer:
192.168.1.99:8180/?ts
The answer will be:
idle or busy

To ask for the transformer's log file as pre-formatted text:
192.168.1.99:8180/?plfp
The transformer will return the entire content of the log file

If the httpCommander encounters a syntax error like missing parameters it will silently close the connection!

httpCommander Commands

Command	Parameters	Description
nothing	nothing	Call the Transformer without any parameters like 192.168.1.99:8180 Depending on the security setting httpCommander_secure=0[1] the httpCommander will return the following: if httpCommander_secure=0 Return: ready if httpCommander_secure=1 Return: the html login screen defined in the file 'login.htm' from the folder CommanderHTTP
User commands		
user	user name	If httpCommander_secure=1 is set in autostart.jt The user's name to log in. Must provide password too - see below
pass	the password	If httpCommander_secure=1 is set in autostart.jt The user's password to log in Example browser URL to login in: 192.x.x.x:8189/?user=myname&pass=mypassword
logout		log me out Ex: 192.x.x.x:8189/?logout
lun		get logged in user's name Returns: the name of the logged in user. if no login required: nothing Example call: 192.168.1.99:8180/?lun
luip		get logged in user's IP Returns: the IP address of the logged in user. if no login required: nothing Example call: 192.168.1.99:8180/?luip
Transformer status commands		
trans		start the transform process use this after having set: - the source path name (the XML file) (command spn) - the output path name (optional) (command opn) - the output file name extension (optional) (command oext) - the XSL path name to use (command xslpn)
quit		quit the transformer application Returns: nothing Example call: 192.168.1.99:8180/?quit
ts		get transformer status Returns: busy or idle busy means, that the transformer currently is transforming an XML file. Example call: 192.168.1.99:8180/?ts
debug	0 or 1	set transformer debug mode Returns: 0 or 1 Example call to turn debug mode on: 192.168.1.99:8180/?debug=1

Command	Parameters	Description
pdebug		get transformer debug mode Returns: 0 or 1 Example call to get current debug mode: 192.168.1.99:8180/?pdebug
pwd		print working directory Returns: the path of the current working directory Example call: 192.168.1.99:8180/?pwd
pcd		print communication directory Returns: the path to the folder where the transformer looks for JobTickets Example call: 192.168.1.99:8180/?pcd Returns: /Users/username/BatchXSLT4InDesignComm/BatchXSLT/
pcn		print communication queue file name Returns: the name of the JobTickets queue file name Example call: 192.168.1.99:8180/?pcn Returns: override.que
pcf		print content of the communication preferences file comm.prefs Returns: the content of the comm.prefs file Example call: 192.168.1.99:8180/?pcf
Transformer and Commander names		
hcn		print commander name Returns: httpCommandServer Example call: 192.168.1.99:8180/?hcn
hcappn		print commander application name Returns: BatchXSLT Example call: 192.168.1.99:8180/?hcn
appn		print commander application name and commander name Returns: BatchXSLT httpCommandServer Example call: 192.168.1.99:8180/?appn
hccp		print http commander context path Returns: / Example call: 192.168.1.99:8180/?hccp
hcdr		print http commander document root Returns: CommanderHTTP/ Example call: 192.168.1.99:8180/?hcdr
hcip		print commander local ip address Returns: /192.168.1.99:8180 Example call: 192.168.1.99:8180/?hcip
LOG file		
plf		print content of log file as is Returns: the content of the log file (unwrapped, plain file contents) Example call: 192.168.1.99:8180/?plf

Command	Parameters	Description
plfp		print content of log file as pre formatted text Returns: the content of the log file (wrapped in <pre></pre>) Example call: 192.168.1.99:8180/?plfp
plfh		print content of log file as HTML text Returns: the content of the log file (each line ending with) Example call: 192.168.1.99:8180/?plfh
plfn		print log file name Returns: the name of the log file Example call: 192.168.1.99:8180/?plfn
plfpn		print log file path and name Returns: the path/name of the log file Example call: 192.168.1.99:8180/?plfpn
clf		delete (clear) the log file Returns: OK for success, otherwise any error code Example call: 192.168.1.99:8180/?clf
Working path names		
spn	path or path/name	set the path or path/name of XML file(s) Returns: the path set Example call: 192.168.1.99:8180/?spn=/Users/transformer/XMLtransforms/in
pspn		print the path or path/name of XML file(s) Returns: the path set Example call: 192.168.1.99:8180/?pspn
opn	path or path/name	set the path or path/name of transformed output file(s) Returns: the path set Example call: 192.168.1.99:8180/?opn=/Users/transformer/XMLtransforms/out
popn		print the path or path/name of transformed output file(s) Returns: the path set Example call: 192.168.1.99:8180/?popn
oext		set the file extension of the output file Returns: the extension set Example call: 192.168.1.99:8180/?poext
poext		print the file extension of the output file Returns: the extension set (like .xml or .html or...) Example call: 192.168.1.99:8180/?poext
xslpn	path/name	set the path/name of XSL file to use Returns: the path/name set Example call: 192.168.1.99:8180/?xslpn=XSL/IMXepaper.xse
pxslpn		print the path/name of XSL file to use Returns: the path/name set Example call: 192.168.1.99:8180/?pxslpn

Command	Parameters	Description
JobTickets		
jt	path/name	load a JobTicket without executing it (no transform) Returns: -99 if successfully loaded (but no transform started) or any other code than 0 on error Example call: 192.168.1.99:8180/?jt=~/XMLtransforms/jt/autostart.jt
jtr	path/name	load a JobTicket without executing it (no transform) Returns: 0 if successfully loaded and executed or any other code than 0 on error Example call: 192.168.1.99:8180/?jtr=~/XMLtransforms/jt/aJob.jt
pjf or pjfh to get JobTicket content as HTML	name or subfolder,name or path/name	get the content of a JobTicket Returns: The content of the JobTicket if found, otherwise nothing. Where does the httpCommander look for JobTickets? If no path is given like: 192.168.1.99:8180/?pjf=override.jt the communications folder is searched at: ~/BatchXSLT4InDesignComm/BatchXSLT/jobtickets/default/ If no path is given like: 192.168.1.99:8180/?pjf=myfolder,override.jt the communications folder is searched at: ~/BatchXSLT4InDesignComm/BatchXSLT/jobtickets/myfolder/ If a path/name is given, this path is searched.
savejtfile	path/name,content	save a JobTicket Returns: 0K on success, otherwise a non zero error code Example call: 192.168.1.99:8180/?savejtfile=~/ajobticket.jt,The content of this file. Writes this file into the user's home folder. NOTE: If content is large, use a form to POST this content
File commands		
flist	path	get the file list of given path Returns: a list of file names separated by \n (new line) if no files available return *none* Example call: List files of 'Export' folder within home folder http://192.168.1.99:8180/?flist=~/Export
dlist	path	get the directories list of given path Returns: a list of directory (folder) names separated by \n (new line) if no directories available return *none* Example call: List directories of 'Export' folder within home folder http://192.168.1.99:8180/?dlist=~/Export
existsfile	path	check if a file or directory exists Returns: 'true' if exists 'false' if not exists Example call: http://192.168.1.99:8180/?existsfile=~/blabla.jt tests if file blabla.jt exists at home folder
writefile writefiledb64 (writefiledb64 is like writefile but the content is Base64 encoded - for binary content)	path/name,content	write to a file Returns: 0K on success, otherwise a non zero error code Example call: 192.168.1.99:8180/?writefile=~/ajobticket.jt,The content of this file. Writes this file into the user's home folder. NOTE: If content is large, use a form to POST the content

Usage Examples Of Transformer Control Through HTTP

Example #1: Perform a simple XML transform

Assume an XML file in a certain folder. It represents all the stuff we have on stock.
We need to send our stock list to a web browser.
This means, that the XML file must be transformed by a special XSL transform style sheet to get a valid html file for the browser.

This is our XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE stock>
<stock>
	<items type="Cars">
		<item name="Mercedes" numitems="2">
			<type>S-class</type>
			<color>black</color>
			<motor>6 cyl</motor>
		</item>
		<item name="Mercedes" numitems="3">
			<type>A-class</type>
			<color>blue</color>
			<motor>8 cyl</motor>
		</item>
		<item name="Mitsubishi" numitems="3">
			<type>Outlander</type>
			<color>white</color>
			<motor>4 cyl</motor>
		</item>
	</items>
	<items type="Bicycles">
		<item name="Runner" numitems="12">
			<type>Cross Country</type>
			<color>green</color>
			<wheel>26 inches</wheel>
		</item>
		<item name="Flyer" numitems="3">
			<type>Downhill</type>
			<color>white</color>
			<wheel>28 inches</wheel>
		</item>
	</items>
</stock>
```

This is what we want to see in a browser:

Our Stock

Cars

Total Cars on stock: 8

Mercedes		
On stock: 2		
S-class	black	6 cyl
Mercedes		
On stock: 3		
A-class	blue	8 cyl
Mitsubishi		
On stock: 3		
Outlander	white	4 cyl

Bicycles

Total Bicycles on stock: 15

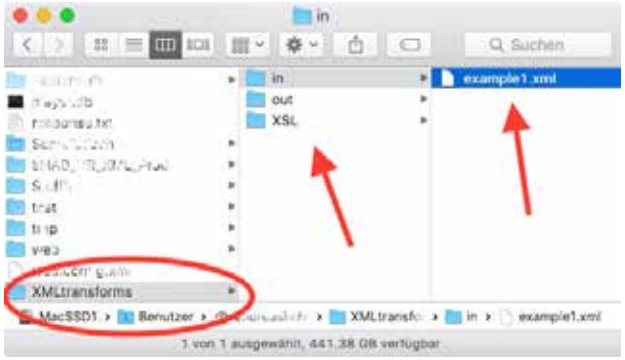
Runner		
On stock: 12		
Cross Country	green	26 inches
Flyer		
On stock: 3		
Downhill	white	28 inches

How this works:

We need an XML transformer. In this case, we use BatchXSLT :-)
We have to tell the transformer which XML file we want to transform using which style sheet to transform it.
And, the transformer must know where to store the output file.

That's it.
Create a working folder structure within your home folder: a folder named 'XMLtransforms' containg the folders 'in', 'out' and 'XSL'. Copy and past the XML and XSL from this example into text files. Name them 'example1.xml' and example1.xsl and place them into the 'in' and 'XSL' folder.

Looks like this:



The XSL transform style sheet to create HTML would look like this:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xsl:stylesheet xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”
                                                version=”1.0”>

<xsl:output method=”html” />
<xsl:output media-type=”text/html”/>
<xsl:output indent=”yes”/>
<xsl:output encoding=”UTF-8”/>

<xsl:output doctype-public=”” />
<xsl:output doctype-system=”” />

<xsl:template match=”/”>
        <html>
        <head>
        <style>
.content { display:inline-block;}
.header { font-size:18pt; text-align:center;}
.title { Margin-top:30px; margin-bottom:10px; padding:5px; background-color:green; font-size:16pt; color:white;}
.itemname { padding:5px; background-color:#ddd; font-size:14pt;}
.type { font-size:12pt;}
.numstock { margin-bottom:10px;}
.color,
.motor,
.wheel { font-size:10pt;}
td:nth-child(1) { width:150px;}
td:nth-child(2) { width:100px;}
        </style>
        </head>
        <body>
        <div class=”content”>
        <div class=”header”>Our Stock</div>
        <xsl:apply-templates/>
        </div>
        </body>
        </html>
</xsl:template>

<xsl:template match=”items”>
        <div class=”title”><xsl:value-of select=”@type”/></div>
        <div class=”numstock”>Total <xsl:value-of select=”@type”/> on stock: <xsl:value-of select=”sum(item/@numitems)”/></div>
        <xsl:apply-templates/>
</xsl:template>

<xsl:template match=”item”>
        <div class=”itemname”><xsl:value-of select=”@name”/></div>
        <div class=”itemstock”>On stock: <xsl:value-of select=”@numitems”/></div>
        <table class=”itemtable”>
        <tr><xsl:apply-templates/></tr>
        </table>
</xsl:template>

<xsl:template match=”type”>
        <td class=”type”><xsl:value-of select=”.”/></td>
</xsl:template>

<xsl:template match=”color”>
        <td class=”color”><xsl:value-of select=”.”/></td>
</xsl:template>

<xsl:template match=”motor”>
        <td class=”motor”><xsl:value-of select=”.”/></td>
</xsl:template>

<xsl:template match=”wheel”>
        <td class=”wheel”><xsl:value-of select=”.”/></td>
</xsl:template>
</xsl:stylesheet>
```

To control the Transformer we use the following commands:

spn=~/XMLtransforms/in/example1.xml (the location of the XML file)
opn=~/XMLtransforms/out/ (where to store the output file)
xslpn=~/XMLtransforms/xsl/example1.xsl (the location of the XSL file - the transform style sheet)
oext=.html (which file extension to use for the output file)
trans (to kick off the transform)

All these commands may be written on a single line into the browser's address line:

http://192.168.1.99:8180/?spn=~/XMLtransforms/in/example1.xml&opn=~/XMLtransforms/out/&xslpn=~/XMLtransforms/xsl/example1.xsl&oext=.html&trans

We will find the HTML file ‚example1.html' at the folder ~/XMLtransforms/out/

Happy Coding!

Disclaimer