Post to File

The *Post to File* stage applies an XSLT or a JSON transform to the input AspireObject and then writes the result to a file. "Post to File" can handle multiple AspireObjects, including the parent AspireObject and multiple child AspireObjects - *all written to the same file*.

Post to File (Aspire 2)	
Factory Name	com.searchtechnologies.aspire:aspire-post-file
subType	default
Inputs	An AspireObject with the metadata of each document to be posted.
Outputs	A transformed XML or JSON representation of the record, which is then written to a file. Also sets the "postToFileWriter" and the "postToFileName" variables on the job to hold a writer to the file and the file name of the file.

A sample pipeline for accomplishing this might look like this:

blocked URL

Note that there are multiple instances of Post To File in the pipeline. However, only the first one specifies the output file. It is assumed by Post To File that all remaining instances will write to the same file.

Configuration

Element	Type	Default	Description
outputFile	Simple Template	Prior File	Specifies the file to which the transformed AspireObject (and other text, such as headers and footers) will be written. If not specified, the file which was previously opened by an earlier Post To File (either in the same pipeline OR in a parent pipeline) will be used. Note that the file name can be specified as a Simple Template. This gives you the flexibility to specify the file name as a combination of metadata from the AspireObject and/or variables attached to the job. See Simple Templates (Aspire 2) for more information.
postXsl	string		The XSL transform file to be used to transform the incoming Aspire Object document into the XML which is written to the file. If a relative file name is specified, it will be relative to ASPIRE_HOME (config/xslt is a common location).
saxonProcessor	boolean	false	Set on true if you want to use SAXON processors (which support XSLT 2.0). Only useful for XML transforms.
postJsonTransform	string		The JSON transform file to be used to transform the incoming Aspire Object document into the JSON which is posted to the file. If a relative file name is specified, it will be relative to ASPIRE_HOME (config/xslt is a common location). For more information on writing JSON transforms, see Post JSON.
header	string		Specifies a fixed string to be written out before the transformed data is written. Often used to add a headers to ensure that the resulting file is well formed. NOTE: If "batchFiles" is set, the header will only be written once at the beginning of the batch (also requires that jobs have been batched by a prior Branch Handler).
footer	string		Specifies a fixed string to be written out after the transformed data is written. Often used to add a footer to ensure that the resulting file is well formed. NOTE: If "batchFiles" is set, the footer will only be written once at the end of the batch (also requires that jobs have been batched by a prior Branch Handler).
batchFiles	boolean	true	If true, then jobs are written out as batches. This means: 1) if <outputFile> is set then a single file will be opened per batch (not one per job), 2) The header is written once at the start of the batch, 3) the footer is written once at the end of the batch. In general, <outputFile> should always be set for this option or odd interleaving of batch headers and contents may result.

Example Configuration

```
  <component name="PostToFile" subType="default" factoryName="aspire-post-file">
    <outputFile>data/records-{XML:id}.xml</outputFile>
    <header><![CDATA[<records>]]></header>
    <postXsl>config/xsl/data-conversion.xsl</postXsl>
    <footer><![CDATA[</records>]]></footer>
 </component>
```

If Using the postToFileWriter

The PostToFile component will set the "postToFileWriter" variable on the job if the <outputFile> parameter is set. You can use this writer to write to the file, but you *must* synchronize on the writer before writing to it. For example:

```
 Writer w = (Writer)job.getVariable("postToFileWriter");
 synchronized(w) {
   // Use the writer here
   w.write("Hello World!");
 }
```

Or, in Groovy:

```
 synchronized(postToFileWriter) {
   // Use the writer here
   postToFileWriter.write("Hello World!");
 }
```

You can not depend on the writer methods to do the synchronization for you, since your write might show up in the middle of another thread's transform.

Transformation Parameters

Two helpful parameters will be set whenever the transformation is called. They are:

- **component** - Contains the name of the component which is running the transformation

This parameter will allow you to use a single XSLT transform in multiple components, and to determine which component is calling the template at which time. Using a single XSLT transform should simplify the process of maintaining your transform code.

- **first** - is "true" (the string) for the first job in a batch, and "false" otherwise.

The purpose of this parameter is to allow you to use an XSLT transform for the header information.(Unfortunately, it is impossible to set a parameter to identify the last job in a batch.)

XSLT Examples

In an XSLT transformation, you can use these parameters with the <xsl:param> directive.

Using the "component" Parameter

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

   <xsl:template match="/doc">
      <xsl:choose>
        <xsl:when test="$param = 'PostToFileParent'">
          <!-- Your parent code goes here -->
        </xsl:when>
        <xsl:otherwise>
          <!-- Your child code goes here -->
        </xsl:otherwise>
      </xsl:choose>
   </xsl:template>

 </xsl:stylesheet>
```

Using the "first" Parameter

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

   <xsl:template match="/doc">
      <xsl:if test="$first = 'true'">
        <!-- Code for writing the header goes here -->
      </xsl:if>
      <!-- other code goes here -->
   </xsl:template>

 </xsl:stylesheet>
```

Parameters in Groovy Transforms

The parameters in groovy transforms will be simple Groovy variables ("component" and "first"). Note that they will both be strings.