mediainfo-to-master.xsl
Stylesheet explanation

Updated October 2018

The most recent version of the stylesheet added the following functionalities:

1) Use of global and local variables for caching values/filepaths
2) Use of matched templates for metadata fields in the filelist section
3) Use of conditional testing to combine multiple stylesheets into a single one, called mediainfo-to-master.xsl

Variables cache values or filepaths. Templates create the PREMIS/Dublin Core XML tags. When the variable is called in a template, it populates the fields with the correct information. Global variables (those located outside of a template) can be called in multiple places across the stylesheet.

Variables include:

$uri – locally-defined uri for the digital preservation system
$aip-id – unique identifier for the aip
$coll-id – unique identifier for the collection
$file-nodes – counts the number of <File> nodes in the MediaInfo; proxy for counting number of files
$aip-filepath – caches a frequently used filepath in the stylesheet
$mediainfo-version – caches the value of the MediaInfo version so it can be called in rest of stylesheet
$aip-size = sums the size of each file in the aip; total sum is in bytes

Named templates include:

aip-title
aip-id
collection-id
aip-version
object-category
aip-size
aip-unique-formats
relationship-aip

Match templates include matches for the following nodes:

//track[@type='General']
CompleteName
FileSize
Format_String
FileExtension

Explanation of conditional tests in the mediainfo-to-master.xsl

Generate the <filelist> tag

The <filelist> section and the applied templates contained within are wrapped by an if test that tests whether there is more than one file in the MediaInfo output.

```
<xsl:if test="$file-nodes > 1">
      <filelist>
      <xsl:apply-templates select="//track[@type='General']"/>
      </filelist>
</xsl:if>
```

$file-nodes is a global variable whose value is the count of <File> nodes in the Mediainfo XML.

Therefore, the filelist section is only displayed when there is more than one file in the AIP.

aip-id

The aip-id is cached as a global variable that can be called in multiple other places in the stylesheet. Because the information include in the aip-id differs based on whether the aip represents a file (1 item) or a representation (multiple items), an `<xsl:choose>` test is used.

Two `<xsl:when>` tests examines the text of the 'FileExtension' tag.

1) If the tag contains the words "pdf", "xml", or "doc", then the stylesheet identifies that it is a metadata aip. If the condition is met, then a regex is applied on the value of ‘FolderName’ element of the first file. The aip-id is pulled out, adding "_metadata" to the end.
2) If the tag contains the words "mov", "wav", "mp3" or "dv", then the stylesheet identifies it as a media aip. If the condition is met, then a regex is applied on the value of the ‘FolderName’ element of the first file. The aip-id is pulled out, adding "_media to the end.

Note: appending "_media" or "_metadata" to the end of the aip-id is done in order to differentiate aips of the same interviews/AV file from each other in the digital preservation system.

aip-title

The aip-title template has three test conditions and one default:

1) when the <FileName> tag contains the value "_pm.wav"
2) when the <FileExtension> tag contains any of the following values: "xml", "pdf", "doc"
3) when more than one file in the aip
4) default

First `<xsl:when>` tests for whether the first track element has a FileName child element that contains the string “_pm”. If the test is true, then the regex selects everything except for the “_pm”. Then it appends “_media” to the end. This is a media aip.

Second `<xsl:when>` tests for whether the FileExtension element contains the string 'xml' or 'pdf' or 'doc'. The presence of preservation documents with these file extensions indicates that this is a metadata aip. If the test is true, then the regex creates the aip-id and appends "_metadata" to the end of the aip-title.

Third `<xsl:when>` tests for whether the whether the $file-nodes variable is greater than 1. If true, then the aip contains multiple files. Therefore, the regex copies the literal string value of the FolderName tag (which is just the aip-id). This is done so that the title from only one file is not used as the representative title for the whole aip.

The `<xsl:otherwise>` default condition has the broadest regex. It analyzes the string value of the FileExtension tag, and copies everything but the file extension. This is done to capture titles with multiple words in them.

e.g. rbrl033short-bcam003_sanders-running-again-ad.mov

becomes

rbrl033short-bcam003_sanders-running-again-ad

object-category template

The required PREMIS objectCategory field can have one of two values: file or representation.

The `$file-nodes` variable, which counts the number of file nodes in the MediaInfo XML, is used for the test condition.

File denotes an aip that contains a single file.
```
<xsl:when test="$file-nodes = 1">
```

Representation an aip with multiple files.
```
<xsl:when test="$file-nodes > 1">
```

aip-unique-formats template

The first `<xsl:if>` test checks that there is more than one File node in the XML (only need to perform aip-unique-formats test if there are multiple files). If more than one file, then continues on to the choose/when test.

There are two `<xsl:when>` conditions:

1) When the CompleteName tag does not contain a string value of 'avchd'
    a. Uses `<xsl:for-each-group>` to select the Format child element of each track parent element.
    b. Then performs a sort using each track's Format element, and in this way, removes duplicates.
    c. Outputs the value of each of the unique Format matches into the <premis:format>.

2) When the CompleteName tag contains a string value 'avchd'
    a. Note: the follow steps are based on the assumption that if an aip does have an avchd directory, then the only other file in the directory will be a preservation master .mov file and therefore, there will be no duplicate files
        i. As a result, a `<xsl:for-each-group>` sort is not performed on the files in this test.

- Uses pre-filled <premis:format> tag for the avchd information with the Format Note reading that: "A complex directory format that is a composite of sub-component files, enumerated in the AIP-level file list."

Format match template in filelist section

Some of the subdirectory files that make up an AVCHD directory do not have Format tags generated in the MediaInfo XML.

- Therefore, `<xsl:if>` is used to test for when a 'FileExtension' element *does not* have a following-sibling element called ‘Format’.
- If the condition is true, then the value from the ‘FileExtension’ element is used for <premis:format> field, and <premis:formatNote> specifies that a format could not be found and so file extension was identified in its place.