**Regular Expressions used in our XSLT (3/30/2018)**

This regex is used in the fits-to-master\_multifile.xsl and fits-to-master\_singlefile.xsl stylesheets and is located at the beginning and end of the stylesheets. In most cases, the regex defines the group(s) to select, which are in parentheses.

**Parameter, Variables, and Regex section:**

These are all matching a portion of the file path. Most matches start at the beginning of the string because the aip-id can be in the file path in more than one place and we only want the first occurrence.

**hargrett collection-id**

examples: harg-ms1234, harg-ua12-3456

regex: ^.+?/(harg-[mu][sa]\d{{0,2}}-?\d{{4}})

* **^.+?** from the beginning of the string
* **/(harg-** start the group with the first "harg-" that occurs directly after a forward slash
* **[mu]** m OR u
* **[sa]** s OR a
* **\d{{0,2}}** 0-2 numbers
* **-?** optional dash
* **\d{{4}}** 4 numbers
* **)** end of the group

**hargrett aip-id**

examples: harg-ms1234er5678, harg-ua12-3456er7890

regex: ^.+?/(harg-[mu][sa]\d{{0,2}}-?\d{{4}}er\d{{4}})

* **^.+?/(harg-[mu][sa]\d{{0,2}}-?\d{{4}}** start the group with the first collection-id
* **er** er
* **\d{{4}}** 4 numbers
* **)** end of the group

**hargrett aip-title**

examples: harg-ms1234er5678\_AIP Title/Folder/file.ext, harg-ua12-3456er7890 \_AIP Title/file.ext

regex: ^.+?/harg-[mu][sa]\d{{0,2}}-?\d{{4}}er\d{{4}}\_(.\*?)/

* **^.+?/harg-[mu][sa]\d{{0,2}}-?\d{{4}}er\d{{4}}\_** find the first aip-id followed by an underscore
* **(.\*?)/**  the group is everything between the underscore and the next forward slash

**hargrett file-id**

examples: harg-ms1234er5678\_AIP Title/Folder/file.ext, harg-ua12-3456er7890 \_AIP Title/file.ext

regex: harg-[mu][sa]\d{{0,2}}-?\d{{4}}er\d{{4}}.\*

* **harg-[mu][sa]\d{{0,2}}-?\d{{4}}er\d{{4}}** Start match with the aip-id...
* **.\*** ...and also include the rest of the string after the aip-id

**russell collection-id**

example: rbrl-123

regex: ^.+?/(rbrl-\d{{3}})

* **^.+?** from the beginning of the string
* **/(rbrl-** start the group with the first "rbrl-" that occurs directly after a forward slash
* **\d{{3}}** 3 numbers
* **)**  end of the group

**russell aip-id**

example: rbrl-123-er-456789

regex: ^.+?/(rbrl-\d{{3}}-er-\d{{6}})

* **^.+?/(rbrl-\d{{3}}** start the group with the first collection-id
* **-er-**  -er-
* **\d{{6}}** 6 numbers
* **)** end of the group

**russell aip-title**

example: rbrl-123-er-456789\_AIP Title/Folder title/other folder/file.ext

regex: ^.+?/rbrl-\d{{3}}-er-\d{{6}}\_(.\*?)/

* **^.+?/rbrl-\d{{3}}-er-\d{{6}}\_** find the first aip-id followed by an underscore
* **(.\*?)/** the group is everything between the underscore and the next forward slash

**russell file-id**

example: rbrl-123-er-456789\_AIP Title/Folder title/other folder/file.ext

regex: rbrl-\d{{3}}-er-\d{{6}}.\*

* **rbrl-\d{{3}}-er-\d{{6}}** Start the match with the aip-id...
* **.\*** ...and also include the rest of the string after the aip-id

**Template Match="created"**

Both of these are used in <xsl:when>'s to convert a date pattern into the required date format of yyyy-mm-dd. These two, in combination with an <xsl:when> that extracts dates already in the right format from a longer dateTime string, cover all the date formats we have observed in the FITS output so far.

**<xsl:when test="matches($apdate, '[a-zA-Z]+ \d{1,2},? [0-9:A-Z ]\*\d{4}')">**

Use when the month is a word, the day is 1 or 2 numbers, and the year is 4 numbers, in that order.

examples: May 5, 2004; Monday, January 7, 2001 11:22:33 PM; Wed Mar 01 11:22:33 EST 2003

regex: ([a-zA-Z]+) (\d{{1,2}}),? [0-9:A-Z ]\*(\d{{4}})

* **([a-zA-Z]+)** group 1: the month as a full word or letter abbreviation. Matches one or more upper and lower case letters that are followed by a space and a number, which means just match the first word before the day and not the day of week that can be in the beginning.
* a space
* **(\d{{1,2}})** group 2: the day. 1-2 numbers
* **,?** optional comma
* a space
* **[0-9:A-Z ]\*** zero or more numbers, capital letters, and colons: the time, AM/PM, and timezone
* **(\d{{4}})** group 3: the year. 4 numbers

**<xsl:when test="matches($apdate, '\d{1,2}/\d{1,2}/\d{2,4}')">**

Use when the month and day are 1 or 2 numbers and the year is 2 or 4 numbers, in that order.

examples: 12/01/99 12:01 PM; 1/5/2011 1:11:55

regex: (\d{{1,2}})/(\d{{1,2}})/(\d{{2,4}})

* **(\d{{1,2}})** group 1: the month. 1-2 numbers
* **/**  forward slash
* **(\d{{1,2}})**  group 2: the day. 1-2 numbers
* **/** forward slash
* **(\d{{2,4}})** group 3, the year. 2-4 numbers