**Process Notes – R2 (07/27/2017)**

**Contents:**

Process 0 - Name and Title Processing

Process 1 - Name Processing

Process 2 - Title Processing

Process 3 - Field 856, Electronic Location and Access

Process 4 - Authority 4XX and 5XX

Process 5 - Selecting Name/Title and Title Authority

Process 6 - Series Processing

**Process 0 - Name and Title Processing (5/18/2017)**

**0.1) Name and name/title field components**

 Name part: all subfields before $t except $e, $4, $h, $j(X11)

 Role part: $e, $4, $j(X11)

 Title part: all subfields after $t except $h,v,x,y,z,w,0-8

 Series part: $v (8XX), $x (7XX, 8XX)

 Subject components: $x (6XX), $y, $z, $v (6XX)

 Genre part: $h

 Relationship part: $e/$4 (6XX), $i (7XX)

 Other?

**0.2) Figuring out Relationship**

 If 1XX, then name/title is the resource being described

 If 6XX, then relationship is bf:subject

 If also $e or $4, then carry over content using bflc:relation property (see 0.3 below)

 If 700-730, then

 If I2=2, relationship is bf:hasPart

 Else relationship is bf:relatedTo

 If also have $i, then carry over $i content using bflc:relation property

 If 8XX, then relationship is bf:hasSeries

 If 760-788, then

 Relationship determined by tag and I1 (see spec)

 If also have $i, then carry over $i content using bflc:relation property

**0.3) Basic RDF Patterns for Names, Titles, and Relationships**

**0.3.1) RDF for names**

<resource> bf:contribution [ a bf:Contribution ;

bf:agent [ a bf: Person, Organization, etc.

 rdfs:label “label from **Process 1.3**”;

 identifiedBy [ a Identifier ….. ]; see **Subfield $0 spec**

 bflc:nameXXMatchKey “string from **Process 1.1**”;

 bflc:nameXXMarcKey “string from **Process 1.2**” ] ;

bf:role [ a bf:Role

[rdfs:label “…” ]; see **Process 1.4**

 bf:code “…” ] ]. see **Process 1.4**

 If URI from ID for role, then instead:

 bf:role URI for role

If name is from 1XX:

Use bflc:PrimaryContribution instead of bf:Contribution

Also add (needed?)

<resource> bflc:primaryContributorNameXXMatchKey “string from **Process 1.1**”

**0.3.2) RDF for titles**

Construct Title class from title subfield; keep Title subproperties in same order as in field.

bf:Work bf:title [

 a bf:Title

rdfs:label “label from Process 2.3” ;

 bf:mainTitle “content of $a (X30, 240) or $t (X00, X10, X11)” ;

 bf:partnumber “content of $n” ;

 bf:partName “content of $p” ;

 bflc:titleXXMatchKey “see **Process 2.1**” ;

 bflc:titleXXMarcKey “see **Process 2.2**” ;

 bflc:titleSortKey “see **Process 2.4**” ] .

 bf:identifiedBy [ a Identifier ….. ] **see Subfield $0 spec**

Convert content of other MARC title subfields listed in rdfs:label as specified in title spec; order not necessary to preserve.

**0.3.3) RDF for relationships**

<resource> bf:relatedTo\*\* URI

\*\*bf:relatedTo may instead be bf:subject, or bf:hasPart, or bf:hasSeries, or one of the other specific relationship properties

Or if need to express also a specific relationship:

If only have relation label (from 7XX $i or 6XX $e):

<resource> bflc:relationship [a bflc:Relationship;

 bf:relatedTo\*\* URI;

 bflc:relation [rdfs:label “name of relationship” ] ] .

If have relation label and/or relation URI:

<resource> bflc:relationship [a bflc:Relationship;

 bf:relatedTo\*\* URI;

 bflc:relation [ URI for relation;

[rdfs:label “name of relationship” ] ] ].

URI a bf:Work or bf:Instance;

 rdfs:label “label from **Process 2.3**”;

 identifiedBy [ a Identifier ….. ]; **see Subfield $0 spec** bflc:titleXXMatchKey “See **Process 1.1**”;

 bflc:titleXXMarcKey “See **Process 1.2**”.

**Process 1 - Name Processing (5/18/2017)**

**Conversion of X00, X10, X11 names**

Note on name keys: If the fields is a name/title field, include only the subfields before the $t subfield as part of the name. A few subfields can occur in titles and names and if they are after the $t they are part of the title.

**1.1) Making a name match key**

For all: Drop all indicators and subfield codes – keep order in field

 X00 - abcdjq - bflc:name00MatchKey

 X10 - abcdng - bflc:name10MatchKey

 X11 - acdengq - bflc:name11MatchKey

**1.2) Making name marc key**

For all: Keep all indicators and subfield codes – tack tag on to beginning – keep whole field as is even if it has a title in it also -- keep order in field : tagii$atext$btext$gtext

 X00 - bflc:name00MarcKey

 X10 - bflc:name10MarcKey

 X11 - bflc:name11MarcKey

**1.3) Making name rdfs: label**

For all: Substitute blank for each subfield code – keep order in field

 X00 - abcdjq - rdfs:label

 X10 - abcdng - rdfs:label

 X11 - acdengq - rdfs:label

**1.4) Figuring out name role**

- If no $e (X00, X10), $j (X11) or $4, role is “contributor” but use URI from ID: <<https://id.loc.gov/vocabulary/relators/ctb>>

- If $e or $j (X11)

bf:role bf:Role rdfs:label “content of $e (X00, X10) or $j (X11)”

Note: If subfield content has “and”, &, or”,” there are multiple roles in subfield. Separate and process each into a separate bf:role.

- If $4 (for each $4)

bf:role a bf:Role bf:code “content of $4”

or bf:role a bf:Role URI for role from ID

Note: If $4 subfield content has more than 3 characters, discard all in $4 after the first 3 characters. Process only the first 3.

- if tag of field is 1XX, then use class bflc:PrimaryContribution for name information (see Process 0.3).

**Process 2 - Title Processing (5/18/2017)**

**Conversion of X00, X10, X11, X30, and 240 titles**

Note: for subfield strings below that start with “t” include only the subfields that occur in the heading after the $t. A few subfields may occur before and after the $t and if they occur before they are part of the name, not the title.

**2.1) Making a title match key**

For all: Drop all subfield codes – keep order in field

 X00 – tfgklmnoprs - bflc:title00MatchKey

 X10 - tdfgklmnoprs - bflc:title10MatchKey

 X11 - tfgklnps – bflc:title11MatchKey

 X30 – adfgklmnoprs - bflc:title30MatchKey

 240 – adfgklmnoprs - bflc: title40MatchKey

**2.2) Making title marc key**

For all: Keep all indicators and subfield codes – tack tag on to beginning – keep whole field as is even if it has a name in it also -- keep order in field – convert delimiter to $ sign: tagii$atext$btext$gtext

 X00 - bflc:title00MarcKey

 X10 –bflc:title10MarcKey

 X11 - bflc:title11MarcKey

 X30 –bflc:title30MarcKey

 240 - bflc:title40MarcKey

**2.3) Making title rdfs:label**

For all: Substitute blank for each subfield code – keep order in field

 X00 – tfgklmnoprs - rdfs:label

 X10 – tdfgklmnoprs - rdfs:label

 X11 - tfgklnps - rdfs:label

 X30 – adfgklmnoprs – rdfs:label

 240 – adfgklmnoprs – rdfs:label

**2.4) Making title sort string**

 Make sort string from 2.3) string by removing the characters specified in Indicator 2.

 Name new string bflc:titleSortKey

**Process 3 - Field 856, Electronic Location and Access (5/18/2017)**

**3.1)**  **If no $u in field, then nac field 856**

**3.2)**  **If 856 Ind2 = # or 0 or 8**

 If the Instance is electronic (008/23= o or s)

 Instance – hasItem -

Item – electronicLocator – <uri from $u> or bnode (if there are $zy or 3 in field)

 bnode bflc:locator <uri from $u>

 bnode – note – Note – “rdfs:label “content of $z”

 bnode – note – Note – “rdfs:label “content of $y”

 bnode – note – Note – “rdfs:label “content of $3”

If the Instance is NOT electronic

 Create new Instance with title from analog instance and pointer to the Work,

Instance a Electronic

 - link to the Work

 - hasItem -

 Item – electronicLocator – uri or bnode (if there are $zy or 3 in field)

 bnode bflc:locator <uri from $u>

 bnode – note – Note – “rdfs:label “content of $z”

 bnode – note – Note – “rdfs:label “content of $y”

 bnode – note – Note – “rdfs:label “content of $3”

**3.3)**  **If 856 Ind2 = 2**

 Instance – supplementaryContent – <uri from $u> or bnode (if there are $zy or 3 in field)

 bnode bflc:locator <uri from $u>

 bnode – note – Note – “rdfs:label “content of $z”

 bnode – note – Note – “rdfs:label “content of $y”

 bnode – note – Note – “rdfs:label “content of $3”

**Process 4 - Authority 4XX and 5XX (7/27/2017)**

**4.1) Authority 4XX**

Tags 400, 410, 411 without $t (only a name)

* Process using Process 1 and Process 0 (names part) as applicable to make matching keys and then process like bib 1XX and make W – contribution.

(In other words just treat it like a bib 7XX that is a name, not a name/title.) If no role in MARC data then make role id.loc.gov/…/ctr.

Tags 400, 410, 411 with $t or 430

* Separate the part before the $t (name part) and the part after and including the $t (title part); for 430, the title part is all after the $a.
* Create a bflc:nameXXMatchKey with the name part before the $t. (Ignore for 430)
* Compare it to the bflc:nameXXMatchKey made for the 1XX. (Ignore for 430)
* If match then discard name part.
* Treat title part like a variant title: W – title – VariantTitle
* Use Bib spec for “X30, 240” to process variant title parts
* If have name part but it does **not** match the 1XX name part, process like a 4XX with a contribution element (id.loc.gov…/ctr)

**4.2) Authority 5XX**

Tags 500, 510, 511 without $t (only a name)

* Process using Process 1 and Process 0 (names part) as applicable to make matching keys and then process like bib 1XX and make W – contribution

Tags 500, 510, 511 with $t

* Separate the data before the $t (name part) and the data after and including the $t (title part)
* Process the data before the $t using Process 1.1, 1.2, 1.3, 1.5 to make matching keys and then process name parts using Bib spec “X00, X10, etc.”
* Process data after the $t using Process 2 to make matching keys and process title parts using Bib spec “X30, 240”
	+ To determine the relationships, see Process 0.3.3, “RDF for relationships”
	+ If 500, 510, 511 tag has $i, create

bflc:relationship [a bflc:Relationship ;

bf:relatedTo <related W uri>

bflc:relation rdfs:label “content of $i”

* + If 500, 510, 511 tag does not have $i, examine $w/0
		- If $w/0=f, create W –derivativeOf – W
		- If 500, 510, 511 tag does not have $i and $w/0 does not=f, create W – relatedTo -- W

Tag 530

* Process data after the $a using Process 2 to make matching keys and process title parts using Bib spec “X30, 240”
	+ To determine the relationships, see Process 0.3.3, “RDF for relationships”
	+ If 530 tag has $i, create

bflc:relationship [a bflc:Relationship ;

bf:relatedTo <related W uri>

bflc:relation rdfs:label “content of $i”

* + If 530 tag does not have $i, examine $w/0
		- If $w/0=f, create W –derivativeOf – W
		- If 530 tag does not have $i or $w/0=f, create W – relatedTo – W

**Process 5 - Selecting Name/Title and Title Authority (7/27/2017)**

**5.1) Record selection**

Select records to process from the Names file

 - If 130 – select

 - If 100, 110, 111 have $t – select

**5.2) Missing conversion**

Stash any FIELD that occurs in a record and does not get a convert as follows:

bflc: missingConversionSpec – literal (marc field with tag and indicators and subfields)

**5.3) Notes**

- There is not any way to identify a serial or integrating.

- Approximate number of title and name/title records in the LC Authority file (2017) 1,256,426, (2016) 1,178,841, and (2012) 1,055,887 - To identify Series records within the selected records: 008/16 = a

 Additional check: 008/12 not=n; 008/13 not=n

**Process 6 - Series Processing (6/09/2017)**

**6.1) Step 1**

Convert 490 and 8xx together

First break a 490 with repeating $a into multiple 490 with $a($x)($v) in each. Keep 490s in same order as $a’s in the field.

Then if more than one of 490/8xx, pair them assuming they are in same order, i.e., first 490 goes with first 8xx, etc.

**6.2)**  **Step 2**

For each pair make the following:

Instance hasSeries a Instance

 rdfs:label 490-a

 seriesStatement literal 490-concatenate av

 seriesEnumeration literal 8xx-v or 490-v; prefer 8xx-v identifiedBy Issn rdf:value 490-x or 8xx-x

 bflc:appliesTo bflc:AppliesTo rdfs:label 490-3

 instanceOf a Work

 title title construct 8xx but ignore $v

 contribution agent construct 8xx

 identifiedBy Issn rdf:value 8xx-x or 490-x

If the 490 has a $6, make a second hasSeries and apply the same process.

**6.3) Treatment of ISSNs in $x**

8xx $x or 490 $x -- the 490 and/or the 8xx tag may have the ISSN ($x) so get it from either. The ISSN ends up in both the Instance and the Work but that is intentional.

**6.4) Treatment of volume data in $v**

8xx $v or 490 $v (prefer 8xx $v) -- Either or both (but usually both) tags will have the volume number but prefer to get it from 8xx with 490 as fallback.

**6.5) Ignore $l**