Existing Change Management Approaches

Overview

This document includes general discussions about the overall approach and descriptions of known existing change management approaches for various institutions.

General Discussion on Approaches

Question: Which might be easier... When data is changed, approaches to identifying changes...

- a diff between the new and old record - challenging with RDF because the graph is unordered, so doing the comparison can require extra processing. There would need to be some scaffolding to create some structure. Diffs are applied in order. Diff UI to show what has changed makes understanding the impact of the changes easier to understand.
- list the changes

How to distinguish the impact of changes? Determines how much info to present. Human users want to focus in on impactful changes.

Best practices questions:

- format of the notification and how granular it needs to be
- for systems that cache authoritative data, what would be updated based on the changes (partial changes vs. entire cached data)

Two audiences of the stream:

- machine/cache
  - cache wants everything up to date all the time
  - applications that cache may need to take action when a change occurs
- human
  - wants to avoid noise of non-impactful changes
  - wants attention drawn to impactful changes where they may need to take action

Action based workflows...

- What types of changes require significant action when they occur?

Change management documents produced by authoritative providers

Library of Congress

LOC currently implements ATOM feeds for each of its datasets. Examples:

- Subjects: http://id.loc.gov/authorities/subjects/feed/1
- Names: http://id.loc.gov/authorities/names/feed/1
- Organizations: http://id.loc.gov/vocabulary/organizations/feed/1
- ISO 639-2: http://id.loc.gov/vocabulary/iso639-2/feed/1

This provides information about whether and when *resources* have been created, updated, or deleted/deprecated.

LOC is in the process of implementing Activity Stream. This work will duplicate the information communicated with the current ATOM feeds, but will also permit LOC to offer more specific activity streams, such as one devoted solely to authoritative label changes. Some issues being considered:

- URI belongs to a scheme when active (e.g. subject scheme). This is identified by a triple. The scheme triple is removed when it is deprecated. Point is: Deprecation alters the resource in fundamental ways.
- Sometimes a term may move from one authority (e.g. subject) to another (e.g. genre). Adds a triple identifying the new term (e.g. use_instead predicate). This makes redirection or advertisement of the 'replacement' URI easy. But sometimes a single term may be split into two or more. No longer a one-to-one replacement, it is unclear what a suitable replacement would be for the old URI.

MeSH

- documents deletions

Getty

- uses Activity Streams
- mentions Activity Streams in ITWG Technical Update slide 12

3rd party vendor notification

When an entity is requested and is not available, some 3rd party vendor's allow the requestor to be notified when the authority provider adds the entity to the authority.

Example format:

Change management processing by downstream users

Traditional authority file in an ILS

For many libraries, "authority change management" is a paid service acquired from a third party. Because monitoring changes in an authoritative source and determining whether the change is relevant to one's own system is not a simple task, and requires specialized software that may or may not be available in an ILS, many libraries have opted to outsource the task to an authorities vendor (e.g. Backstage Library Works). Services that the vendor provides may include the following:

1. Keep a copy of the library's authority file. This file may be provided by the library at the beginning of the contract, or built by the vendor by matching headings in bibliographic records against one or more authoritative sources.
2. Keep track of all unmatched headings (i.e. headings used in bib records that had no match when last searched) and partial matches (e.g. only the name portion of a name/title heading has a match, or only the main subject heading of a main plus subdivision(s) heading has a match)
3. Monitor changes in the authoritative source (i.e. new additions, change of an existing record, deletion or deprecation of an existing record)
4. Based on this monitoring, provide an updated record when a record that is **in the library's authority file** has changed.
5. At a specified period (e.g. every three months) re-search all previously unmatched headings against the latest version of the authoritative source to see if there is now a match. If so, provide the matching "new" record to the library or have the library send the bib records with that heading for re-processing.
6. At a specified period (e.g. every three months) re-search all previous partial matches against the latest version of the authoritative source to see if there is now a full match. If so, provide the full match record or have the library send bib records with the partial match for re-processing

By using such a service, authority change management for the library is reduced to loading records provided by the vendor. The complicated task of monitoring the change stream of the authoritative source and determining what is relevant is performed by the vendor. And because the boundary of a MARC authority record is unambiguous (it begins with the first character of the leader and ends with the record terminator), maintaining a changed record can be done simply by swapping out the entire record. Knowing what has changed may be unnecessary or is already specified in the profile set up with the vendor that determines what kind of changed records should be supplied.

This same model could be applied, with some variations, to future entity-based cataloging (e.g. records created in Sinopia). In fact, Stanford is talking with Share-VDE right now to establish such a service. However, there are some major differences between the MARC environment and the RDF environment, namely:

- In MARC, what constitutes an authority record is clear and unambiguous. In RDF, you have an entity description in the form of a graph. As has already been pointed out elsewhere in this document, the boundary of such an authority "record" is not as clear. For example:
    - Should the "record" be all the triples that have that entity as the subject?
    - If a statement points to another node (either with URI or without), should that node be included? How to determine whether it should be or not?
    - Should all nodes that are not empty be included, and all nodes that are empty be considered the boundary of that "record"?
    - What if you have an authoritative entity that is the logical combination of multiple entities, e.g. a series could be construed as the combination of a bf:Work and a bf:Instance, in that case do you include two graphs as one authority "record"?
    - When would it be appropriate to unambiguously identify a graph by naming it, i.e. add a fourth element to the triple to tie a bunch of related triples together. The advantage of that is you can swap out the old version and replace it with the new without having to know what has changed.
- In RDF, you don't necessarily cache the whole thing locally as you do a MARC authority record. If you only cache the label and the URI, does it mean that you don't care if a variant label is added to the description? Perhaps this last point could be addressed by setting up a profile with the vendor, so that you only get what is relevant to you, leaving the rest at the source. This would require the more granular monitoring of the "change stream" that we have discussed in our meetings

---

Change document specification approaches

rdflib

Reference: https://rdrr.io/cran/rdflib/man/rdflib-package.html

This is an R package used to manipulate triples. Worth looking at how it specifies changes.

- add - rdf_add - could be used to express adding a triple
- delete - do not see a spec for removing a triple

rdf_add

Reference: https://rdrr.io/cran/rdflib/man/rdf_add.html

rdf_add could be used to specify adding a triple.

Example:

```
rdf <- rdf()
rdf_add(rdf,
    subject="http://www.dajobe.org/",
    predicate="http://purl.org/dc/elements/1.1/language",
    object="en")

## non-URI string in subject indicates a blank subject
## (prefixes to "_:b0")
rdf_add(rdf, "b0", "http://schema.org/jobTitle", "Professor")

## identically a blank subject.
## Note rdf is unchanged when we add the same triple twice.
rdf_add(rdf, "b0", "http://schema.org/jobTitle", "Professor",
        subjectType = "blank")

## blank node with empty string creates a default blank node id
rdf_add(rdf, "", "http://schema.org/jobTitle", "Professor")


## Subject and Object both recognized as URI resources:
rdf_add(rdf,
        "https://orcid.org/0000-0002-1642-628X",
        "http://schema.org/homepage",
        "http://carlboettiger.info")

 ## Force object to be literal, not URI resource
rdf_add(rdf,
        "https://orcid.org/0000-0002-1642-628X",
        "http://schema.org/homepage",
        "http://carlboettiger.info",
        objectType = "literal")
```

SPARQL

This is how you add/delete triples with SPARQL.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT { <http://example/egbook> dc:title  "This is an example title" } WHERE {}
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
DELETE DATA { <http://example/egbook> dc:title  "This is an example title" }
```