Alternative Import Workflow for Inventory Data beyond CBS

Antje Niemann, GBV Göttingen

Felix Hemme, ZBW Kiel / Hamburg
CBS / CBS2FOLIO

Central Union Catalog (CBS) as common cataloguing tool
• ~ 500 German libraries
• ~ 80 million title records (~ 230 million ownerships)
• ~ 15 million authority records
• 2021: 1.3 million new records
• ~ 80 % takeover of third party bibliographic data
• In most cases just local information needs to be added (call number, barcode …)

CBS as data source for the local library systems
• Real-time update

CBS2FOLIO
• Set of components to populate FOLIO inventory storage (instances, holdings, items) with CBS metadata
Library Systems in GBV Consortium

- CBS
  - Shared cataloguing
  - Interlibrary loan
  - Metadata source for Discovery

- Local Library System (currently OCLC LBS4, future FOLIO)
  - ERM
  - Acquisition
  - Circulation

- Not all CBS data is relevant for FOLIO
  - Selection of fields and records -> no authority records, no subject headings
  - Just a minimum of relations between different titles
Import Workflow from CBS to FOLIO Inventory Storage

- CBS / K10plus
- Pica+
- OUF-PICA
- Pica XML
- CBS / K10plus Harvester
- OUF-PICA Harvester
- FOLIO JSON
- Logs / Errors
- Harvester Admin

Inventory Storage
- Mod-inventory-update
- Instance Update
- Holdings Update
- Item Update
Import Workflow from CBS to FOLIO Inventory Storage

• OUF-PICA
  • fetches records from CBS using the CBS OUF tools
  • calculates the record status (upsert or delete)
  • converts Pica+ to XML
  • controls the Index Data Harvester
Import Workflow from CBS to FOLIO Inventory Storage

- Harvester / localindices
  - Its primary use is harvesting of bibliographic records and its holdings
  - Can read data from a variety of data sources
  - Transforms the data through highly configurable XSLT based transformation steps and pipelines
  - Stores the transformed data to storage systems like Solr databases or FOLIO Inventory
  - Harvest job definitions, scheduling, and transformation pipelines are configured in a MySQL database
  - [https://github.com/indexdata/localindices](https://github.com/indexdata/localindices)
Harvester: Transformation via XSLT

Example for XSLT transformation steps
https://github.com/indexdata/cbs2folio-transformations

Excerpt from pica2instance.xsl (source and hrid)

```xml
<xsl:template match="metadata">
  <source>K10plus</source>
  <xsl:variable name="ppn" select="datafield[@tag='003@']/subfield[@code='0']"/>
  <hrid>
    <xsl:value-of select="$ppn"/>
  </hrid>
  <xsl:for-each select="datafield[@tag='001@']/subfield[@code='0'][not(contains(.,',99-99'))]">
    <statusUpdatedDate>
      <xsl:call-template name="pica-to-iso-date">
        <xsl:with-param name="input" select="."/>
      </xsl:call-template>
      <statusUpdatedDate>
    </xsl:for-each>
</xsl:template>
```
Import Workflow from CBS to FOLIO Inventory Storage

- **mod-harvester-admin**
  - Okapi service that can be put in front of Harvester
  - Provides FOLIO based access to control the Harvester
  - https://github.com/indexdata/mod-harvester-admin

- **ui-harvester-admin**
  - Provides an FOLIO/JSON based interface to the configuration database that FOLIO clients (like a Stripes UI) can use
  - https://github.com/indexdata/ui-harvester-admin
Harvester-admin provides an FOLIO/JSON based interface to the configuration for managing harvest jobs (Harvestables)
mod-harvester-admin / ui-harvester-admin

- ... for managing the
  - Storage Engines
  - Transformation Pipelines and
  - Transformation Steps
mod-harvester-admin / ui-harvester-admin

• ... for monitoring logs and error reporting
Import Workflow from CBS to FOLIO Inventory Storage

- **mod-inventory-update (MIU)**
  - Okapi service in front of mod-inventory-storage (Inventory Storage) for populating the storage with instances, holdings and items
  - [https://github.com/folio-org/mod-inventory-update](https://github.com/folio-org/mod-inventory-update)
mod-inventory-update (MIU)

- Accepts GET, PUT and DELETE requests
- 2 matchkeys (each also with batch process)
  - inventory-upsert-hrid / inventory-batch-upsert-hrid (GBV)
  - shared-inventory-upsert-matchkey / shared-inventory-batch-upsert-matchkey
- Provisional instance created when related instance does not exist yet
- Control record overlay on updates / Prevent MIU from overriding existing values

Performance and Scalability

• Stable and sufficiently fast processes for the initial loading of a tenant’s data and the real-time update

• Example: State and University Library Bremen
  
  • Initial loading: 6.75 hours for 18.8 million instances, holdings and items
    Average: 773 records / second
  
  • Real-time update: 2 hours for 1,5 months of changes in the CBS production system
CBS2FOLIO in a nutshell

- Functionality to import non-marc records into inventory
  - Record types such as json or xml are possible
  - 24/7 real-time update
  - Consistent results / controlled overriding of existing values
- Good Performance and Scalability
- Flexible Mapping by XSLT
- 2 matching processes (HRID or matchkey, each also with batch process)
- All CRUD processes are implemented (create, read, update, delete)
- Logging that allows the user to troubleshoot
  - Identifiers are given
  - Error messages are understandable
  - Clean up of the log file planned
  - Information about a hanging job and the last loaded record
- No connection to SRS yet, no authority records, minimum of relations between records
CBS2FOLIO -> thirdPartySystem2FOLIO?

• The software we developed provides the ability to connect a CBS based union catalog to FOLIO Inventory, but is not limited to CBS
• Let’s take a deeper look at some of the components and possible scenarios
Import scenario

- Functionality to import MARC and non-MARC records
- MARC should be provided in **XML serialization**
- No limit to file size. Harvester can be configured to split large files at defined number of records
- Can load in parallel using multiple *harvestables* at once
  - A *harvestable* is a job configuration that holds information about the transformation pipeline, storage, log level, URL to monitor
  - Can be used just once or multiple times, depending on the use case
- Create, update, and delete *Inventory record sets*
  - An *Inventory record set* is a set of records including an Inventory instance, and an array of holdings records with embedded arrays of items
Managing transformations

• Create XSLT mappings for MARC files
• XSLT can be shared and reused across libraries using external services like GitHub or GitLab, including version control
• This enables a collaborative workflow of managing mappings and the technical conversion
Harvester: Transformation via XSLT

Example for XSLT transformation steps
https://github.com/indexdata/cbs2folio-transformations

Excerpt from pica2instance-new.xsl (source and hrid)
A look at the Harvester admin FOLIO app
Improvements

Some areas of interest might be:

• SRS connection?
• Matchkeys?
• Testloads?
• Use cases?
SRS connection

• Reminder: No connection to SRS yet
• MIU populates into mod-inventory-storage directly
• The Harvester can store original MARC records in a given storage
• SRS records link to their Inventory equivalent by storing their UUIDs in 999’s – would need to look up the UUIDs after an import
• Unknowns:
  • Performance when populating SRS
  • Actions taken by SRS on Inventory records
Matchkey methods

• Two matchkeys implemented to match on HRID or an matchkey
• Would potentially need to enhance matchkeys to support matching on ID’s like OCLC ID in MARC 035$a/$z and on other fields in the Inventory records
• Investigate the need for multiple matchkeys with if/else conditions

• Example:

```java
if 035$z matches instanceIdentifierTypeId abc
    then update the instance
else if 001 matches hrid
    then update the instance
otherwise do nothing
```
Testloads

• Investigate a dry run functionality
• Perform a testload to see if record matching would work as expected
• Preview some statistical data, e.g.
  • Records matched
  • Records created
  • Records updated
  • Records deleted
  • Errors
Summary

• MIU and MHA can be used to connect a CBS based union catalog to Inventory, but they are not limited to CBS as a source.
• The tools are format agnostic; they rely on XSLT transformations and can convert data that is provided in an XML (MARC XML, DC XML, PICA+ XML etc.) or JSON format
• MIU has proven to be reliable when it comes to loading millions of instances, holdings, and items during migration. It is also performant when loading batches of records on a daily basis.
• There is no connection to SRS yet. MIU is connected to mod-inventory-storage directly. If there is interest in pushing (MARC) data into SRS, the workflow has to be enhanced.
Thank you!

antje.niemann@gbv.de
f.hemme@zbw-online.eu
The text of this presentation is published under the Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license: https://creativecommons.org/licenses/by-nc/4.0/

Excluded are graphics, screen shots and pictures from other authors. Their rights and licenses continue to be valid.

Decisive for this presentation is the spoken word.