Inventory reference data: Considerations for customizations, translations, system upgrades

Julian Ladisch, VZG
Felix Hemme, ZBW
What is Inventory reference data?

- Reference values are (mainly) a pair of a unique ID (UUID) and a name (display label) that is displayed in the UI (plus some more administrative metadata)
- They can be managed in Settings or directly via APIs using the defined endpoints
- A list of reference values is tied to an Inventory field
- When populating those fields, only the UUID is stored in the record and not the actual name
- The UI is looking up the names by their UUIDs in order to display them
- Exports etc. only contain the UUID, not the name of a reference value

Before performing imports into mod-inventory-storage, the reference data has to be prepared and made available
What is Inventory reference data?

Example from /alternative-title-types

```json
{
    "id": "2ca8538d-a2fd-4e60-b967-1cb220101e22",
    "name": "Added title page title",
    "source": "folio",
    "metadata": {
        "createdDate": "2023-08-15T01:48:43.986+00:00",
        "updatedDate": "2023-08-15T01:48:43.986+00:00"
    }
}
```
What is Inventory reference data?

Example instance record containing an alternative title

```
"alternativeTitles" : [ {
  "authorityId" : null,
  "alternativeTitleTypeId" : "2ca8538d-a2fd-4e60-b967-1cb220101e22",
  "alternativeTitle" : "A journey through Western Europe"
} ],
```

<table>
<thead>
<tr>
<th>Alternative title type</th>
<th>Alternative title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added title page title</td>
<td>A journey through Western Europe</td>
</tr>
</tbody>
</table>
Definitions in the GBV library network

• At GBV, we distinguish between
  o **FOLIO reference values** that are provided by the system itself (if loadReference = true)
  o **Locally added reference values** added by a library or a library network
• In GBV, all tenants use reference values with the same name and UUID. Exception: Locations, since library-specific
• Reference values can only be stored in one language; the translation process cannot be managed via [https://lokalise.com/](https://lokalise.com/)
• The translation and use of **FOLIO reference values** takes place across all library networks in Germany, so that a reference value is represented by the same UUID. Exception: **Locally added reference values** when there is no coordination between parties
Multiple sources

- The **FOLIO reference values**, which FOLIO includes by default, are located at [https://github.com/folio-org/mod-inventory-storage/tree/master/reference-data](https://github.com/folio-org/mod-inventory-storage/tree/master/reference-data)
- Their translations are managed at [https://github.com/zbw/folio-inventory-translations/tree/main/translated-files](https://github.com/zbw/folio-inventory-translations/tree/main/translated-files) and can be imported with a shell script
- GBV’s **locally added reference values** are located at [https://github.com/gbv/folio-referencedata](https://github.com/gbv/folio-referencedata) and can be imported by a shell script as well. This repository also contains a list of **FOLIO reference values** that are not needed in GBV and can be deleted.
- Our CBS mapping relies on the reference value’s UUIDs as well, therefore we manage a codes2uuid XSL file at [https://github.com/indexdata/cbs2folio-transformations/blob/master/codes2uuid.xsl](https://github.com/indexdata/cbs2folio-transformations/blob/master/codes2uuid.xsl)
Process to date

- One of our tenants served as reference system for all Inventory reference values
- When setting up a new tenant, all reference values were copied from the reference system into the new tenant
- This step has proven to be error-prone in the past
- We were also missing a way for version control
- Managing reference values was tied to our SysOps – but system librarians wanted to be more independent and able to relieve them
Goals

• When a new tenant is initially set up, all reference values (both FOLIO and GBV-specific) must be imported before loading Inventory data. Only then may the initial loading of instances, holdings and items take place.
  ○ At best, this step also includes the import of the German translations
  ○ The locations must also be determined before the initial loading
• If, for whatever reason, all or individual reference values are "lost" in an existing tenant, they must be easy to reload
• If a new version of mod-inventory-storage contains new FOLIO reference values, their German translations has to be importable as soon as they have been translated
• There should be as little work as possible for SysOps and system librarians
Inventory Translations Repository

- [https://github.com/zbw/folio-inventory-translations](https://github.com/zbw/folio-inventory-translations)
- Contains German translations
- For other languages: Simply create a new directory in the repository
- Git allows for versioning and maintains the commit history
- Repository contains the update-translations shell script
- Prerequisite: FOLIO user with write permissions for the inventory reference data
- Note: This repository is a temporary solution until mod-inventory-storage implements internationalisation as required by [DR-000009](https://github.com/zbw/folio-inventory-translations)
GBV Union Catalog Reference Data Repository

- [https://github.com/gbv/folio-referencedata](https://github.com/gbv/folio-referencedata)
- Contains local reference data
  - that is specific to GBV Union Catalog (K10Plus)
  - that is required for data import from union catalog
- Contains unused FOLIO default reference data
  - that is not needed by GBV and
  - should be deleted
- Repository contains shell scripts that insert local reference data and delete unused default reference data
Deploy issues

- Currently GBV runs a FOLIO installation with 20 tenants
- Whenever installing or upgrading a tenant the sysops need to run the reference data scripts
  - Must provide tenant specific credentials
- How to automate?
Reference Data Module

• Solution: Combine data and shell scripts into a backend module
• Sysop only needs to add that new module to install.json
• Okapi takes the install.json file
• Okapi will do all the work
• Tiny module with about 5 MB container size, based on Alpine
• Development of the module will start after WOLFcon
Module Descriptor

- The ModuleDescriptor (MD) is metadata about a module
- What does the MD of the reference data module contain?
Module Descriptor „requires“ Section

• The „requires“ section contains all needed mod-inventory-storage
  ○ APIs and
  ○ Permissions
• Okapi sorts modules using the „requires“ information
  ○ Dependency resolution
  ○ Only install/upgrade a module after all dependencies are ready
• Sorts mod-inventory-storage before reference data module
• Okapi may install/upgrade multiple modules in parallel
  ○ If dependency resolution allows it
Module Descriptor „provides“ Section

• The „provides“ section contains
  ○ the „POST /_/tenant“ API
  ○ needed permissions

• Okapi automatically calls the API on tenant install and tenant upgrade

• Okapi provides a token with all needed permissions
  ○ No user with those permissions is needed

• The reference data module executes the scripts from the two repositories mentioned before
Summary

• Reference data repositories
  o Commit history
  o Versioning
  o Collaboration

• Using Okapi has multiple advantages
  o Correct install order
  o Parallel execution
  o Okapi creates token, no user credentials needed
  o Automatic execution
Your questions

julian.ladisch@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.