Share VDE Virtual Discovery Environment

WOLFcon 2022
Aug. 31 – Sept. 2
Hamburg, Germany

Share-VDE and FOLIO cooperation

A challenge to introduce linked open data into the wider FOLIO perspectives

WOLFcon 2022, September 2

Tiziana Possemato

Share-VDE as a library-driven community

Share-VDE in a nutshell

Since 2016, R&D work to facilitate libraries in the transition from MARC-based cataloguing to linked data

this expanded over time from the pilot project to Share-VDE and the Share Family of initiatives

https://svde.org
https://wiki.svde.org/
Casalini Lab
Share - Linked Data Environment

What Share-VDE does:
MARC data (or other traditional formats) are converted to linked data

data describing library resources are connected in a union catalogue, and can be queried as authoritative source

exposition for end users and professionals on the web platform www.svde.org

a platform to manage data in a linked open data environment

A cooperative and library-driven initiative

Share-VDE is a collaborative initiative based on the needs of libraries, developed and supported by:

- the joint effort of the Share-VDE Advisory Council and of the Working Groups;
- Casalini Libri, provider of bibliographic and authority data as member of the Program for Cooperative Cataloguing;
- @Cult, provider of ILS, Discovery tools and Semantic web solutions for the cultural heritage sector;
- the vision of Linked Data for Production initiative with special endorsement of Stanford;
- with input and active participation from an international group of research libraries.

A collaborative community can produce a gravitational wave of energy that expands...

… and meets other communities, with other energy to Share

Active participation

Penn
UNIVERSITY *of* PENNSYLVANIA

PCC

BRITISH LIBRARY

NYU

University of Colorado Boulder

Duke
UNIVERSITY

THE UNIVERSITY OF CHICAGO

UNIVERSITY OF MICHIGAN

UNIVERSITY OF ALBERTA
LIBRARY

1640
THE NATIONAL LIBRARY OF FINLAND

Yale

Stanford University

nb.no

SHARE

LIBRARY
LIBRARY OF CONGRESS

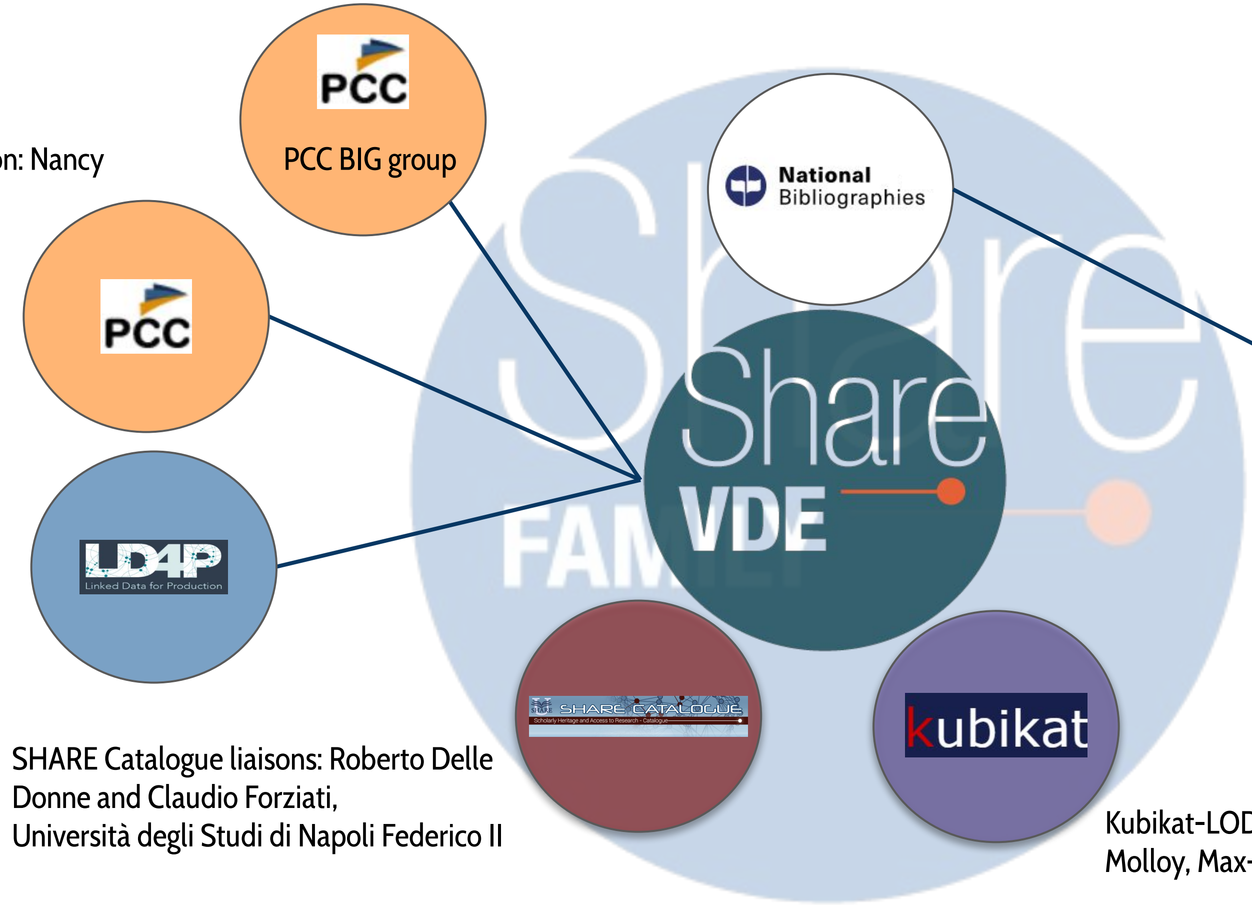
Smithsonian

Libraries members of SVDE working groups and parallel projects are constantly contributing with their Subject Matter Experts to requirements gathering, functional analysis and feedback to developments.

Share VDE

Share Family and Share-VDE liaisons

PCC liaison: Nancy Fallgren, NLM-NIH

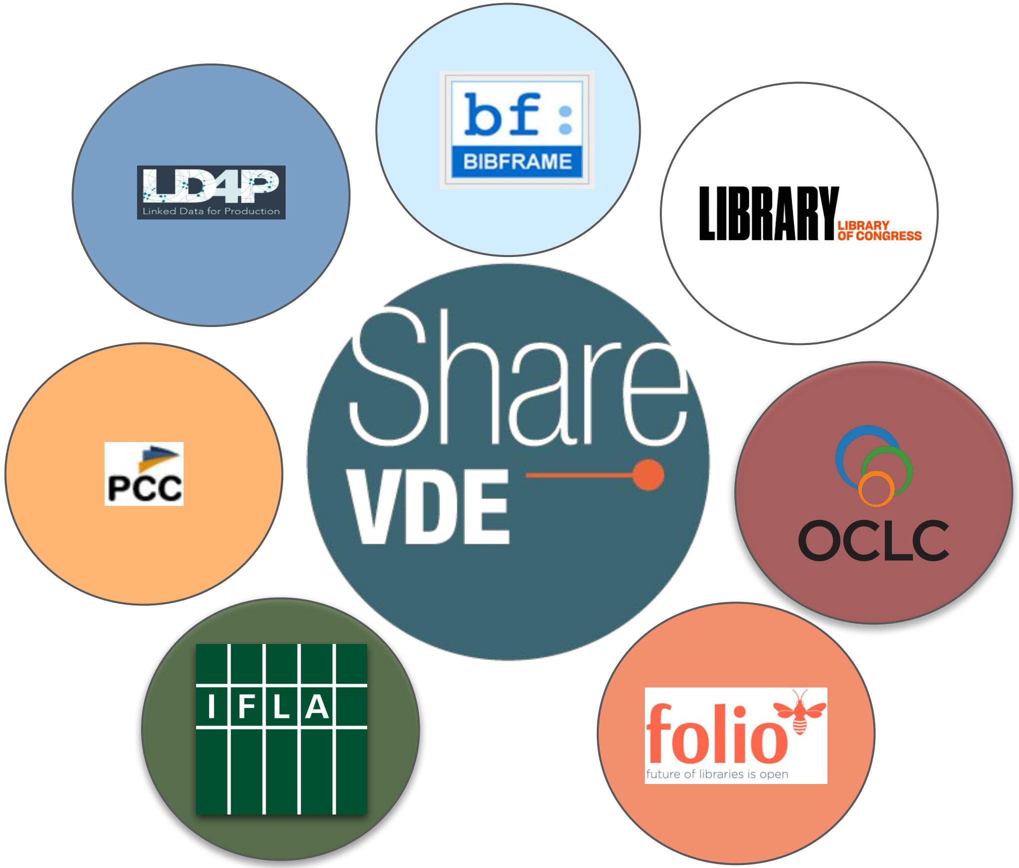


IFLA Bibliography Section liaison: Maud Henry, KBR - Royal Library of Belgium

SHARE Catalogue liaisons: Roberto Delle Donne and Claudio Forziati, Università degli Studi di Napoli Federico II

Kubikat-LOD liaison: Cora Molloy, Max-Planck-Institute

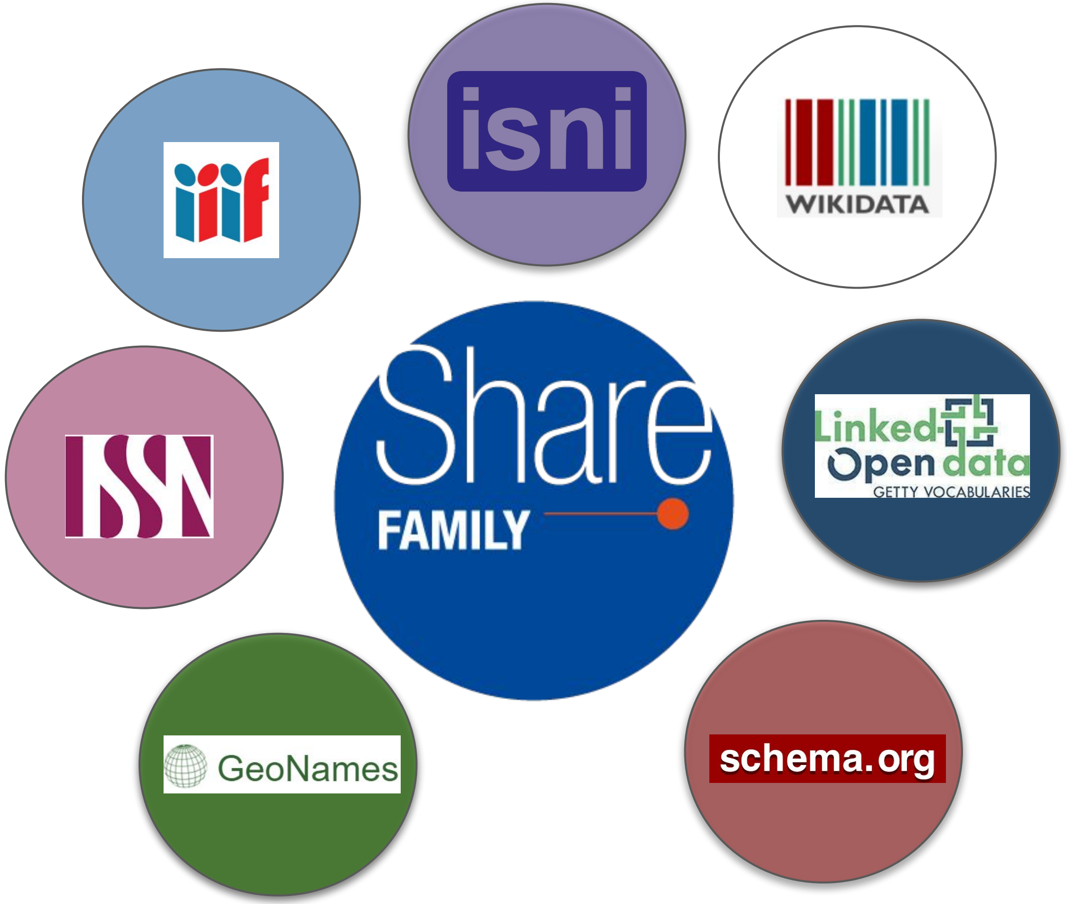
Community engagement: library community



Extended community: collaboration with heterogeneous initiatives and institutions in the library domain

Scientific value: sharing of data and services in different technological environments and diverse bibliographical and cultural context

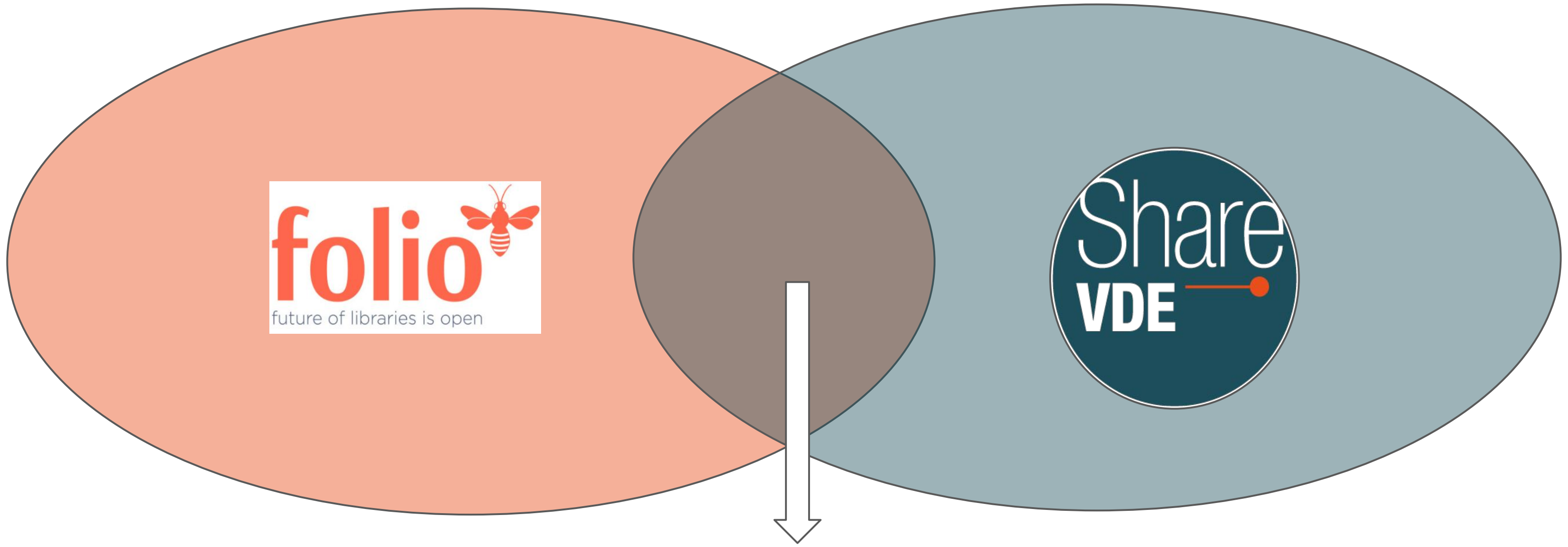
Community engagement: World Wide Web



Mixed community: cross-domain cooperation across the Web community

Scientific value: same solutions serve scopes of different communities, data reuse

FOLIO - Share-VDE intersection

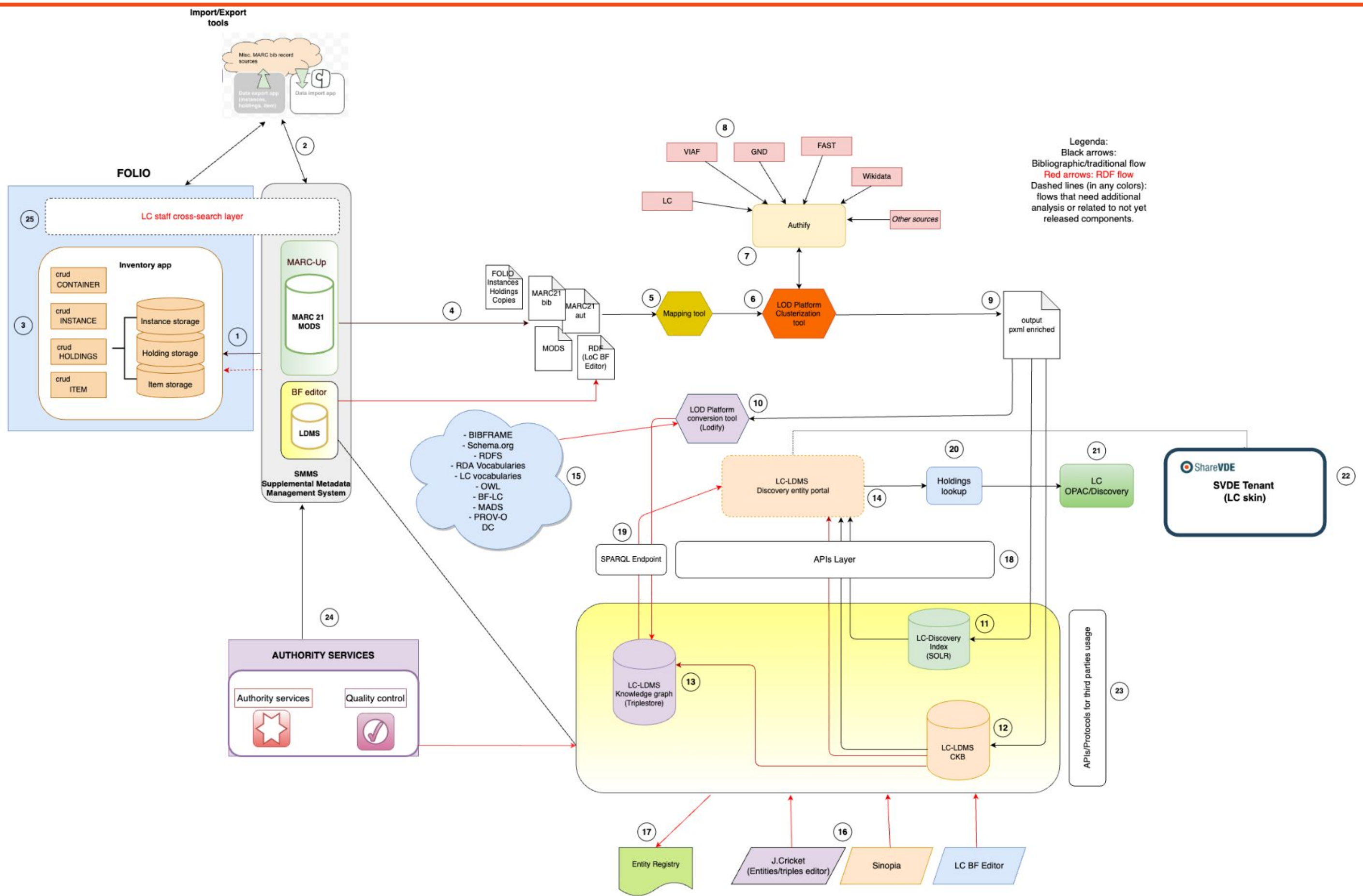


This intersection is what we want to discuss: **how to manage a data flows for libraries that are both part of FOLIO community and of Share-VDE community**

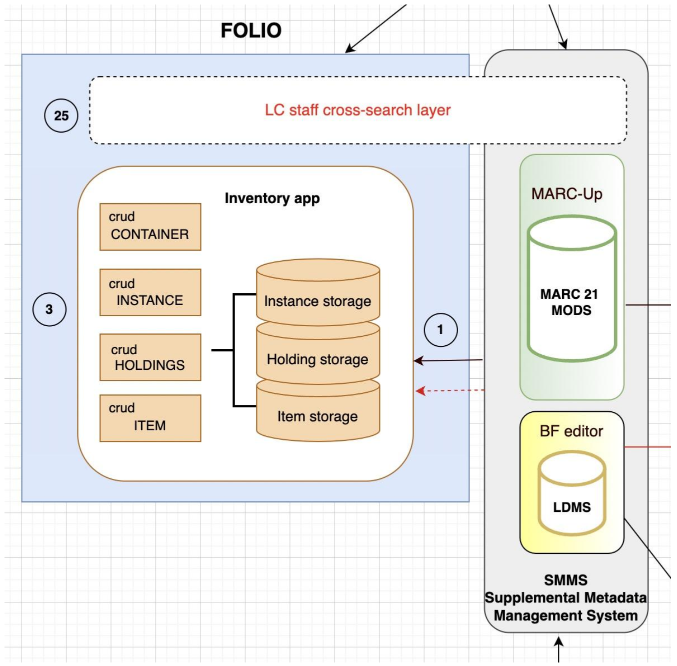
Orchestration of data flow

Share VDE

Share-VDE - FOLIO data flow



SVDE - FOLIO data flow - Step 1(1/2)



1) Main event: Records in MARC 21 are created/edited in MARC-Up (or in any local cataloguing environment) and updated in Inventory using the FOLIO APIs.

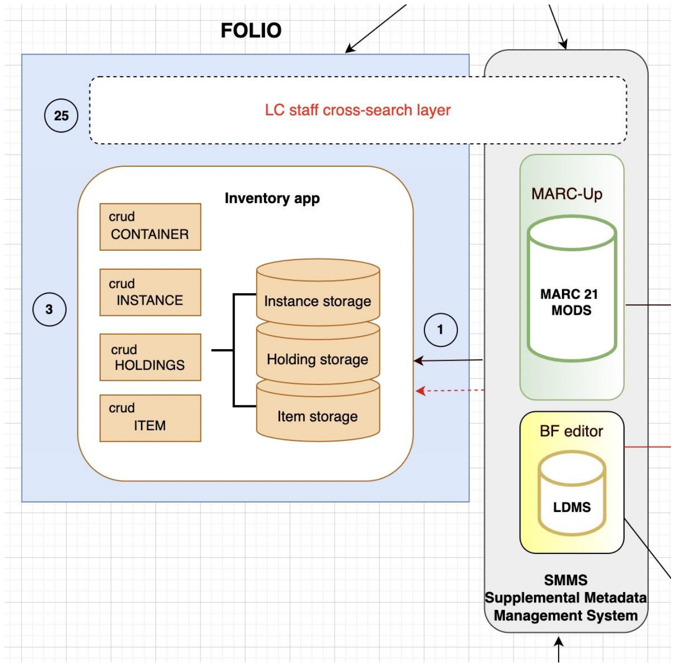
In progress: the same flow, from a BF editor (eg J.Cricket) to Inventory.

Extended description

Record-based metadata is created using MARC-Up, the cataloging module to edit MARC 21 (bibliographic and authority) records. The same process applies to LOD based workflow.

The data created with MARC-Up, as MARC 21, or with a BF editor, are then automatically updated in FOLIO Inventory, using the mod_inventory suite of APIs (mod_inventory, mod_inventory_storage, and mod_inventory_update). All create, update, and delete operations are performed on MARC-Up or BF editor and each modification is made in FOLIO Inventory in real-time. The data flow is always from the cataloging/editing tools to Inventory and never vice versa: every update/correction/cancellation of data are made in MARC-Up/BF editor and reflected on Inventory.

SVDE - FOLIO data flow - Step 1(2/2)

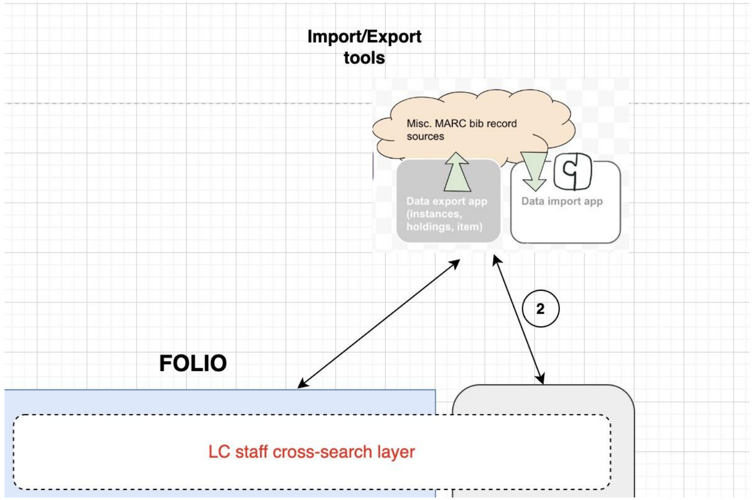


1) Main event: Data in RDF created using one of possible BF editor (eg J.Cricket) are updated in Inventory using the FOLIO APIs.

Extended description

RDF created in a potential BF editor are reflected in Inventory through FOLIo APIs. Consistency of identification/relationships of BF entity - Inventory/Instance is guaranteed through identifiers (every entity and every attribute in a LOD environment has its own identifier, that needs to be reflected in the Inventory/Instance).

SVDE - FOLIO data flow - Step 2


Import/Export tools
Misc. MARC bib record sources
Data export app (instances, holdings, item)
Data import app
2
FOLIO
LC staff cross-search layer

2) Main event: Groups of records are downloaded and imported into MARC-Up database from external sources

Extended description

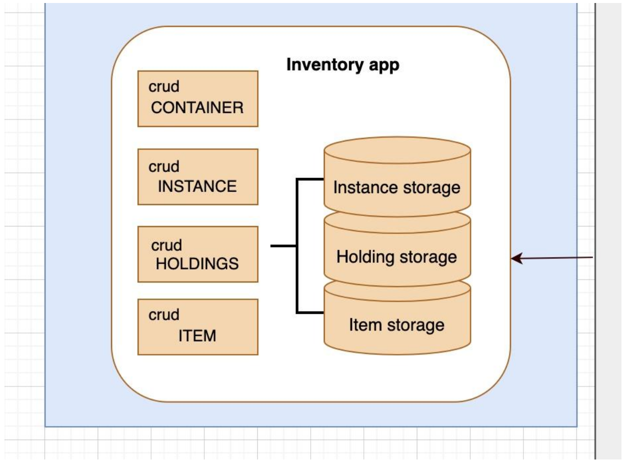
Data can come from external sources using the Import tool: data in multiple formats can be downloaded and imported into the MARC-Up database using the Import function of the Import tool. Data saved on MARC-Up database are also updated on Inventory, using the related Inventory Batch API.

SVDE - FOLIO data flow - Step 3

3) Main event: Instances are created in FOLIO-Inventory

Extended description

Following the creation of the instance record in MARC-Up/BF editor and the corresponding flow of records into FOLIO Inventory, holdings and items are added using the FOLIO Inventory app. The Inventory app records are a subset of richer descriptive formats (such as MARC2 1 and Linked Open Data), and contain enough information to be used in the other library workflows (acquisitions, circulation, etc.).



SVDE - FOLIO data flow - Step 4

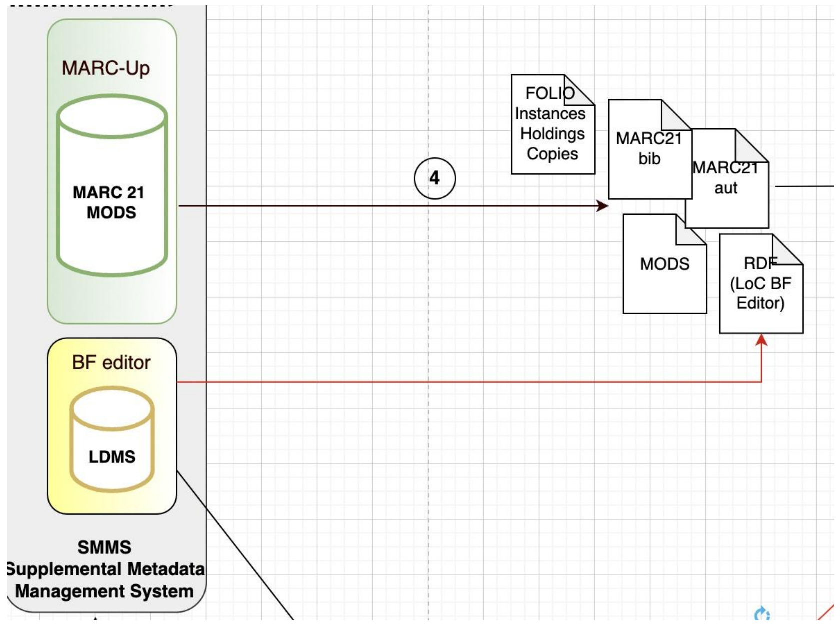
4) Main event: the Library exports the records to the LOD system in one of the supported formats.

Extended description

Original data are exported using the Import/Export tool in an overnight automated process. Original data can be exported in different formats, e.g.:

1. MARC 21 bibliographic records;
2. MARC 21 authority records;
3. RDF created using LC editor (Marva), Sinopia or J.Cricket: in case an RDF editor will be included, such as Sinopia - see item 15 - the RDF format will be managed as original format, for data created as linked data from scratch;
4. other possible formats

Data are sent in the LOD pipeline to be automatically converted in RDF following the SVDE/Lib conversion rules. Data export, in order to provide LOD conversion, must be performed by both SMMS (for bibliographic and authority records) and FOLIO (for Holdings / Copies data).

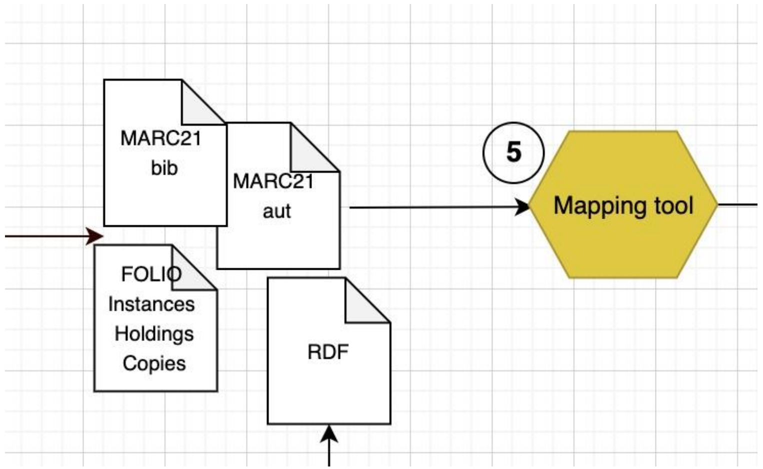


SVDE - FOLIO data flow - Step 5

5) Main event: Metadata produced during the cataloging processes are analysed to comply with LOD conversion processes (clustering of entities and conversion) using the mapping tool

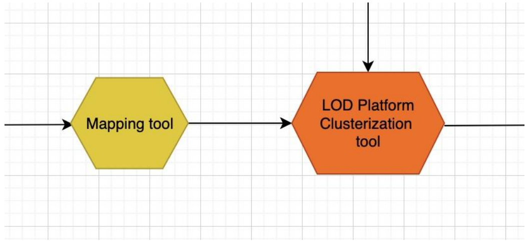
Extended description

Metadata in different formats is analyzed to identify the elements (content and format) useful for subsequent clustering and conversion processes into LOD. The system is sufficiently flexible to allow the extension of the source formats over time, allowing it to adapt the clustering and conversion processes in an agile way. This mapping phase—supported, in the case of new formats, by the analysis of domain experts—allows for the adjustment of the clustering and conversion logic in order to accept a wide and rich range of formats.



SVDE - FOLIO data flow - Step 6

6) Main event: The LOD Platform clustering tool creates clusters of entities (Agents, Work, Instance etc.)



Extended description

LOD Platform - Clusterization tool: the tool includes the clustering logic for the data coming from different, often non-homogeneous, sources in order to create the entity as a Real-World-Object (RWO) and assign a unique identifier. By clustering, we mean the mechanism of identification of the entities with Large Scale Fuzzy Name Matching Techniques, through different text analysis methods such as:

- Common key
- List
- Edit distance
- Statistical similarity
- Word embedding

These methods tackle issues about data identification, among them: similar names, split database fields, phonetic similarity, spelling differences, truncated components, titles and honorifics, initials and nicknames, etc. Other analysis logic supports the creation of a cluster / entity as well as the definition of the preferred form, among the many possible variants, to be assigned during the data presentation phase, including:

- access point size/weight
- usage count
- identifiers presence, etc.

(continues) SVDE - FOLIO data flow - Step 6

The process produces a cluster of data in which the many possible variant forms of a name (of whatever entity it is) are reconciled and collected in a cluster, which is assigned an identifier unique to the Library. The entities already managed with clustering processes are the following:

- Agent (Person, Organization, Family, Meeting)
- svde:Opus
- Work (with also HUB type)
- Instance
- Item
- Subject (Agents type, right now)
- Topics: we currently have algorithms and processes to enrich a topic with external URIs from external sources (such as FAST or LCSH) using mostly string matching. Intense work is being carried on to enrich terms from subject strings with Wikidata sources; this allows the Library to expand the number of identifiers, thus partially overcoming the issue of different alphabets and reducing the risk of creating clusters that seem to identify the same entity but actually don't (entity recognition process).

In addition we manage some "domain" entities, such as:

- places;
- occupations;
- roles;
- languages;
- other data coming from controlled vocabularies.

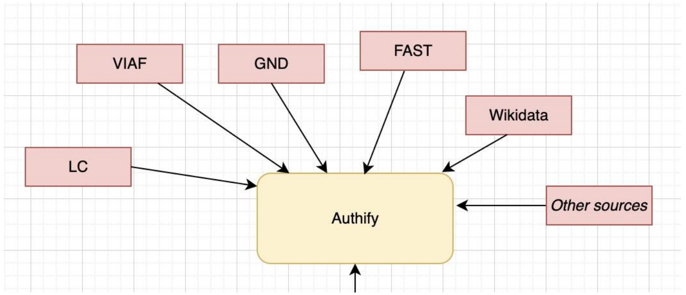
SVDE - FOLIO data flow - Step 7

7) Main event: entities are enriched with data from external sources (e.g. VIAF, Wikidata, FAST etc.)

Extended description

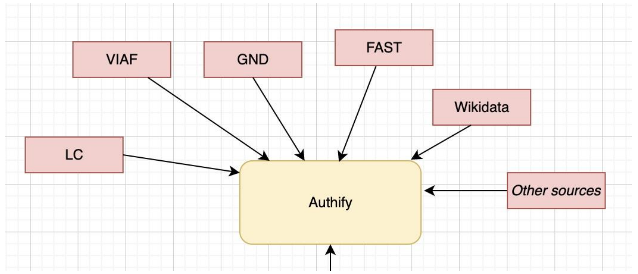
Authify is a RESTFul module that offers several search and detection services. At the very beginning, the LOD Platform project aimed at overcoming some limitations of the public VIAF Web API with the goal to obtain more comprehensive and precise URI retrieval results. VIAF, being a public project, doesn't allow a heavy invocation of its API: for those use cases where such a requirement is needed, the VIAF project provides a download of the whole dataset.
That was the main reason why Authify was implemented: indexing and storing the VIAF clusters dataset and providing, on top of that, powerful full-text and bibliographic search services. Other sources will be added progressively, to answer different libraries' needs.
The Authify Cluster Search Services provide, as the name suggests, a full-text search service among names and works clusters. The search Web API uses, behind the scenes, an "invisible queries" approach in order to try and find a match, as precise as possible, within the managed clusters.

In Authify, a complete hyperlinking process is created and the final result is a much richer and well-identified entity than the original one



SVDE - FOLIO data flow - Step 8

8) Main event: the data imported from external sources (e.g. VIAF, Wikidata, FAST etc.)

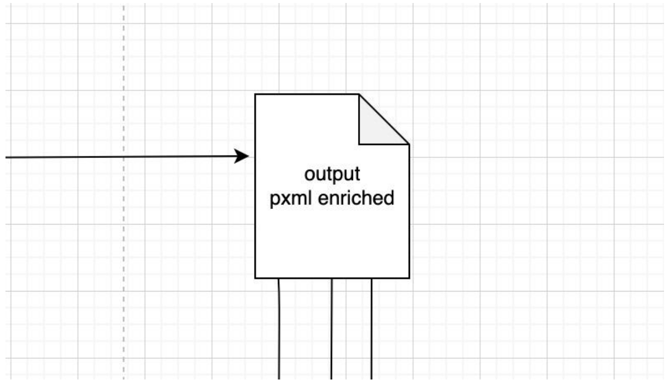


Extended description

The Authify tool also manages the external entity data sources through different processes depending on the query methods available from the external source (APIs/Web Services, bulk file loads, protocols etc.). At the beginning of each new project, a check list of the most relevant sources for the specific project is done with the target library in order to identify new available sources to be included in the process.

SVDE - FOLIO data flow - Step 9

9) Main event: The LOD platform creates an intermediate file in a proprietary format to feed different internal pipelines

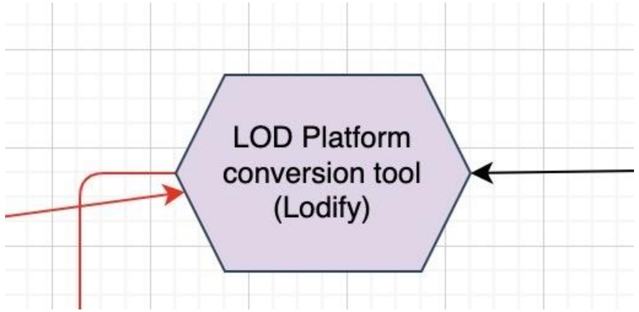


Extended description

The clusterization/reconciliation/enrichment processes have as output a *pxml file,* a proprietary file format designed to express the richness of data in a standard way. This file is used to feed two distinct processes, the LOD conversion and the text indexing into SOLR.

SVDE - FOLIO data flow - Step 10

10) Main event: The LOD Platform converts the data from the editors and processed through the enrichment in steps 3-8 to linked data



Extended description

RDFizer - The LOD Platform RDF conversion tool. RDFizer (an evolution of the previous Lodify module) is a RESTFul module that automates the entire process of converting and publishing data in RDF according to the BIBFRAME 2.0 ontology in a linear and scalable way. It is flexible and adaptable to multiple situations: it allows the Library to manage the classes and properties not only of BIBFRAME but also of other ontologies as needed. Lodify works strictly in conjunction with other LOD Platform tools and components such as Authify, the database of relationships and the Cluster Knowledge Base. The platform represents an enhanced and expanded version of the ALIADA framework within an infrastructure that is better adapted to handle large amounts of data. The enriched pxml file described in item 9 acts as the input for RDFizer, which translates it into triples and uploads them to the selected triplestore. Upon completion, the RDF data can be extracted as a Turtle file by using the APIs provided by the triplestore.

RDFizer manages two conversion procedures:

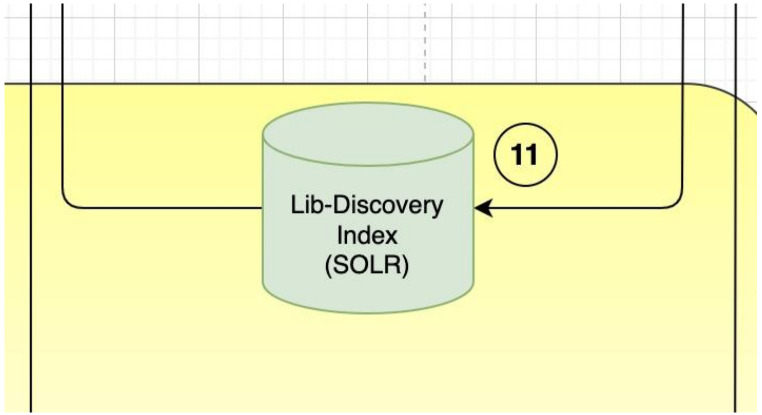
- Cluster Conversion: converts data obtained from the cluster enrichment process;

- Record Conversion: converts data obtained from an enriched MARC/MODS file into a triple.

SVDE - FOLIO data flow - Step 11

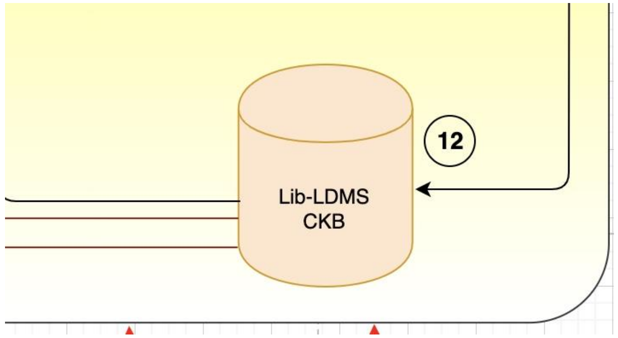
11) Main event: the data processed are indexed by the Solr search engine

Extended description
The Discovery Index (SOLR) - The same pxml file from item 9 is used for the inverted index in SOLR: this search engine, used in combination with the triplestore for the presentation of data in the search portal, allows to enormously extend the entities search and retrieval. Combined with what is made available by the triplestore, it allows the end user to access the data by having the entity as the subject of the research, and no longer the bibliographic or authority records. A complex and extended knowledge panel will be proposed for each entity addressed in the system, to show its attributes and the rich network of relationships with other entities, in a way that tries to combine the richness of data with the user-friendly and intuitive discovery. A long list of search and retrieve logic offered by the SOLR system can be applied to extend the search capabilities of the system.



SVDE - FOLIO data flow - Step 12

12) Main event: the data converted feed the entities Cluster Knowledge Base, ie the database of Library entities



Extended description

The *Cluster Knowledge Base* (CKB) is the result of the data processing and enrichment procedures with external data sources for each entity. The CKB is populated with clusters of all the linked data entities that are created within the specific project that uses the LOD Platform. Such clusters derive from the reconciliation and clustering of the bibliographic and authority records and of RDF data created by the SMMS (Supplementary Metadata Management System), to form groups of resources that are reconciled and converted to linked data to represent a real world object.

The CKB is the *authoritative source* of the system and it's available both on the relational database Postgres (mostly for internal maintenance purposes, reports etc.), as well as in RDF in order to be used for the Entity Discovery Interface and public exposure. The CKB is updated both through automated procedures as well as through manual actions via the entity editing module CKB editor.

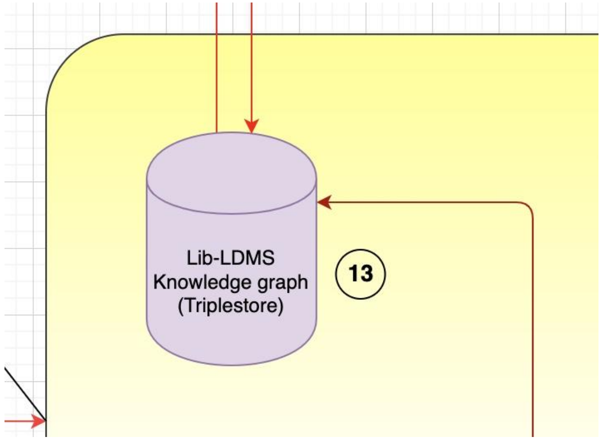
SVDE - FOLIO data flow - Step 13

13) Main event: the data converted to RDF are sent to the triple store

Extended description

The Knowledge Graph (triplestore) - The data converted to RDF according to the agreed entity model (BIBFRAME 2.0 as core ontology, and other classes and properties derived from the ontologies indicated in point 15) are indexed in the triplestore. Also the data stored in the triplestore can vary (according to the update cycles defined by the target library), both through manual and automatic procedures, via the CKB editor module.

The Knowledge Graph contains the CKB in RDF, and relations and connections among resources can be inferred and queried via SPARQL endpoint. Advanced API layers are currently under development.



SVDE - FOLIO data flow - Step 14

14) Main event: The data are published to the Entity Discovery Interface on the web portal.

Extended description

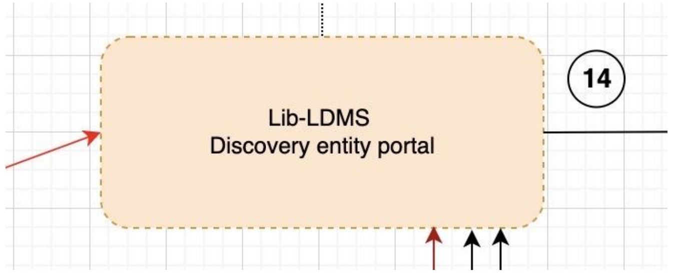
The *Discovery Interface* will harness the potential of linked data to offer an easy and intuitive user experience and deliver ever more wide-ranging and detailed search results to library staff, basing on the BIBFRAME data model.

A library that is already a member of the Share-VDE community has at least two opportunities to manage its LD discovery portal, to achieve the result to present its data in a linked data environment:

- set up a dedicated skin portal within the Share-VDE tenant;
- set up an autonomous tenant with independent skin portal as part of the broader Share Family initiative;

in the flow chart above, the box for the LD portal is drawn with a dashed line precisely to indicate this double opportunity for the Library to have or not have its own tenant or just a skin on Share-VDE.

The design focus of the portal, in both the cited approaches, is to provide intuitive access to complex data and make BIBFRAME easy to understand and benefit from.



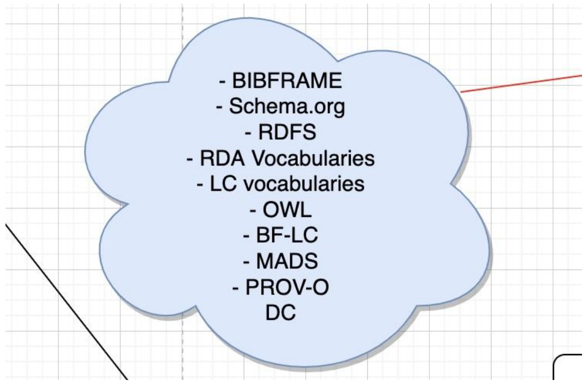
SVDE - FOLIO data flow - Step 15

15) Main event: Entity modeling is a key process in a LD process. The system creates BIBFRAME data but can integrate different ontologies

Extended description

The data modeling in a LOD project is one of the most delicate and crucial aspects of the whole data management process. The conversion tool, Lodify (*RDFizer*, in the new version), is built in a way that allows to extend it, following an approach as open as possible to receive and manage changes/extensions in existing ontologies and the inclusion of new ontologies and controlled vocabularies. Currently, the conversion tool uses the following ontologies:

- Bibframe, in its version 2.0
- BF-LC
- RDFS
- OWL
- MADS
- PROV-O
- RDA Vocabularies
- LC Vocabularies



SVDE - FOLIO data flow - Step 16

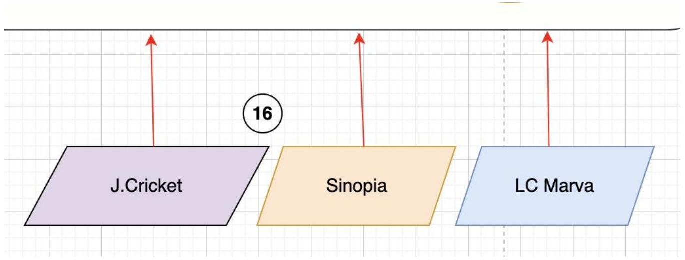
16) Main event: The data produced by SVDE can be edited by J.Cricket linked data entity editor or by other BF editors (such as Sinopia and/or Marva, the LC BF editor)

Extended description

All data produced by conversion processes and stored in the different databases can be modified manually, to better address the issue of data quality.

Within the Share-VDE initiative, a cluster/entity editor (J.Cricket) was designed and it's currently under development. As already mentioned, all changes in the clusters, also through manual actions, are reported in the Entity Registry (see item 17). The J.Cricket development plan is far-reaching and spans over the current (2022) and the next year (2023).

In addition to J.Cricket, APIs are being developed for a direct connection to the open-source, RDF editor Sinopia (part of the LD4P initiative). The Library could also decide to use the LC BF editor (Marva).



SVDE - FOLIO data flow - Step 17

17) Main event: Changes done to entities are tracked in the Entity Registry

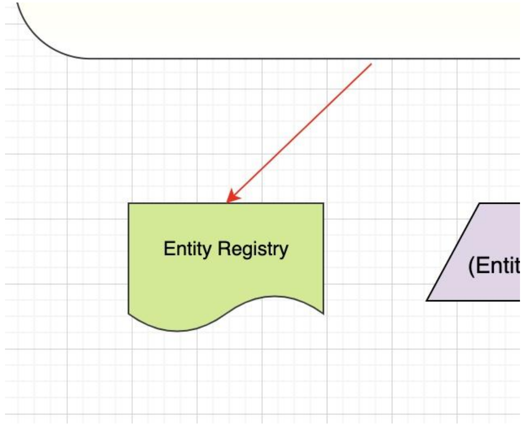
Extended description

The management and tracking of changes to the clusters in the CKB is entrusted to the *Entity Registry*. As suggested by the name itself, the Entity Registry is a special tool in which the association between clusters and the URIs that identify such clusters is registered, and where all the changes affecting this association are reported. An interesting example is the Redirect, that is the registration of the redirect from a cluster no longer valid to a valid one: this guarantees the recovery of the entities and their persistent identification even in the presence of heavy cluster modifications, such as the merge/matching process.

There are two types of processes that can change clusters and their URIs:

- automatic: these are periodical processes starting from the SMMS, and are activated to manage the "delta" of the data;
- manual: through the use of the CKB Editor.

The Entity Registry is a tool to guarantee constant control over the entities' CKB and to share clear and complete information on the data with third parties using the LC data in RDF format.



Each change performed on the entities of the CKB (both manual and automatic) is reported in the Entity Registry, which has the key role of keeping track of every variation of the resource URI, in order to guarantee the effective and broad sharing of resources.

SVDE - FOLIO data flow - Step 18

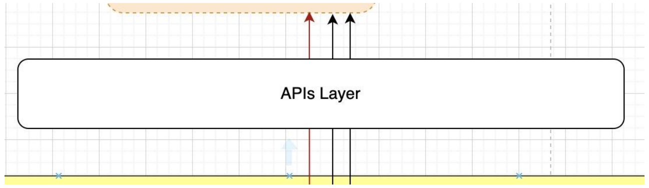
18) Main event: The API layer provides REST APIs layer and a GraphQL layer to query the data produced by the Library

Extended description

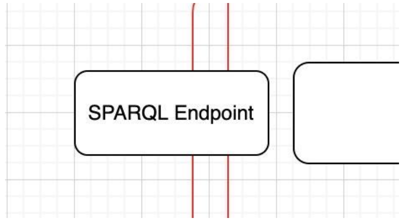
The data produced in the conversion or creation workflow are accessible through an *API layer* that will offer two interfaces to query the entities:

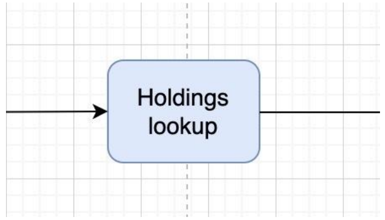
- a REST APIs layer
- a GraphQL layer

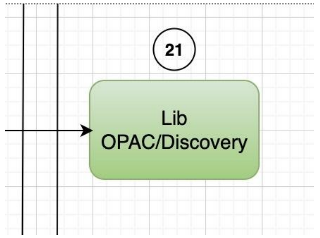
In the REST API's several endpoints serve different request types. In the GraphQL layer, the same endpoint will perform the queries, using a language similar to SQL. GraphQL is a query language which allows to expose the underlying dataset as a graph. That allows a powerful mechanism for introspecting the entities and the relationships that form the domain model. The same entities are also exposed using a RESTFul paradigm, where a persistent URI is assigned to each entity and HTTP verbs are used for querying / manipulating the dataset.



SVDE - FOLIO data flow - Steps 19, 20, 21







19) Main event: SVDE makes available a SPARQL end point via the triple store

Extended description - SPARQL endpoint - Users can query triple data via SPARQL. A list of properties of datasets will make the query formulation easier for the user.

20) Main event: The Entity Discovery portal will also display holding data

Extended description - Holdings Lookup - Entity Discovery Interface will also display holding data, integrating APIs to retrieve holding information.

21) Main event: The Entity Discovery portal is connected with local library OPAC

Extended description - Library OPAC discovery: the SVDE entity discovery portal will be connected with the local OPAC of individual libraries. The Library is free to select the OPAC (VuFind, Blacklight, or any other it decides to adopt): any OPAC can be integrated so that from the entity portal it is possible to call up local services provided by the selected OPAC.

SVDE - FOLIO data flow - Steps 22, 23

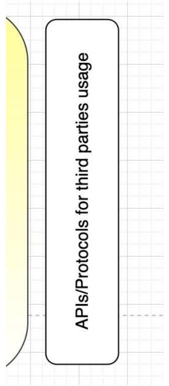
22) Main event: The Library is part of Share-VDE initiative and publishes its catalogue on the portal, with its own skin

Extended description
SVDE tenant includes data coming from the Library of Congress, with data of many other libraries, in an integrated linked data common portal. While the main entity discovery portal of a tenant shows the data of all the institutions feeding the tenant's Cluster Knowledge Base, the skin portal shows only the data of the institution or group of institutions that the skin has been designed for.

ShareVDE
SVDE Tenant
(Lib skin)
22

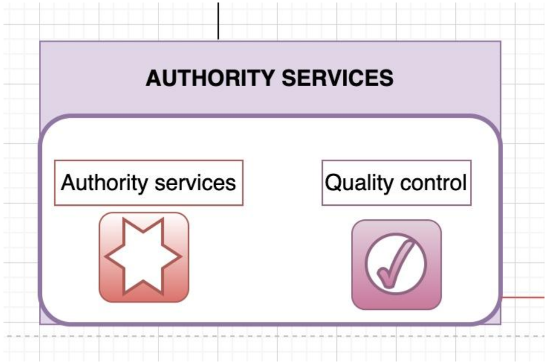
23) Main event: The LOD Platform will make available further tools for data harvesting, including APIs, OAI-PMH, Atom feeds and Activity stream

Extended description
APIs/Protocols for third parties usage - to close the loop, the data elaborated in the LOD workflow will be made available to the Library for its use, through protocols including APIs, OAI-PMH, Atom feeds and Activity stream.



SVDE - FOLIO data flow - Step 24

24) Main event: SVDE is connected with services for the authority control



Extended description

Authority services - the services for the authority control are connected to the MARC and BIBFRAME data workflow. The LOD Platform is integrated with MARC-based specific service workflows for automated enrichment, reconciliation, and validation of library data. Submitted data is checked against a number of leading authoritative sources and updated accordingly. a set of APIs allows for the interrogation of external sources and authority systems. Options are also available for the automatic updating of authority records of the library. The authority services provide for the verification and correction of MARC records and procedures for the automatic enrichment of MARC fields with standard identifiers in URI format from different sources (e.g. URIs from VIAF, ISNI, LCNAF, LCSH, LCGFT, FAST etc.). URIs are added to MARC fields to identify the data unambiguously. In addition to linking the data in the bibliographic records to additional information found in other sources, adding URIs to MARC records is a prerequisite for converting the records into linked data, which is increasingly important for the transition of the library community to the new generation cataloging.
This authority service will also be available for bibliographic descriptions in linked data: in addition to verifying and correcting MARC records, authority control features are designed for linked data-based workflows (e.g. BIBFRAME format). This process will be supported by the LOD Platform technology.

SVDE - FOLIO data flow - Step 25

25) Main event: A unique search system to cross-repositories is available for staff searching

Extended description

The purpose of this search layer is to centralize the responsibility of providing cross-cutting staff search services across three different subsystems:

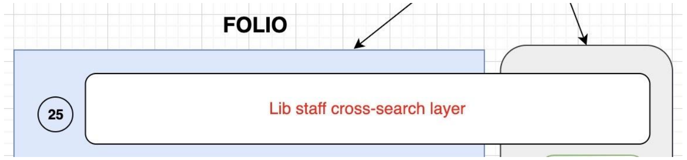
- FOLIO (Inventory)
- Marc-UP
- Linked Data Management System (LDMS)

The module acts as a separate and independent layer that can manage entities composed of a shared superset of all properties belonging to those three different datasets.

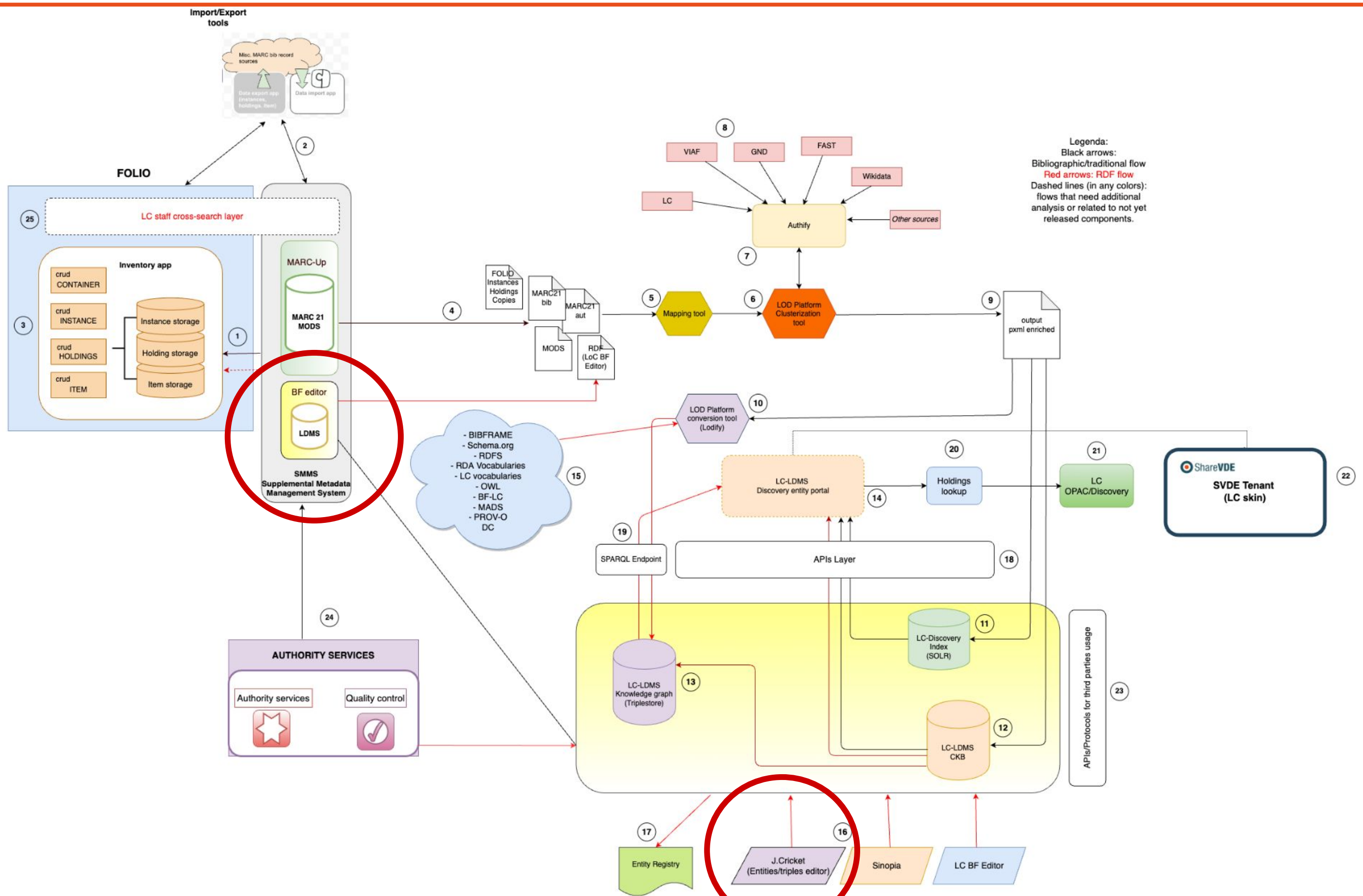
The definition of such an entity is configured within the staff search module and it is published to the other modules, so they are aware of the shape they need to send for synchronizing their data.

As a consequence of that, the responsibility of sending data in the correct format relies on a transformation pipeline implemented on the caller side.

The following diagram illustrates the main components involved in the metasearch subsystem.

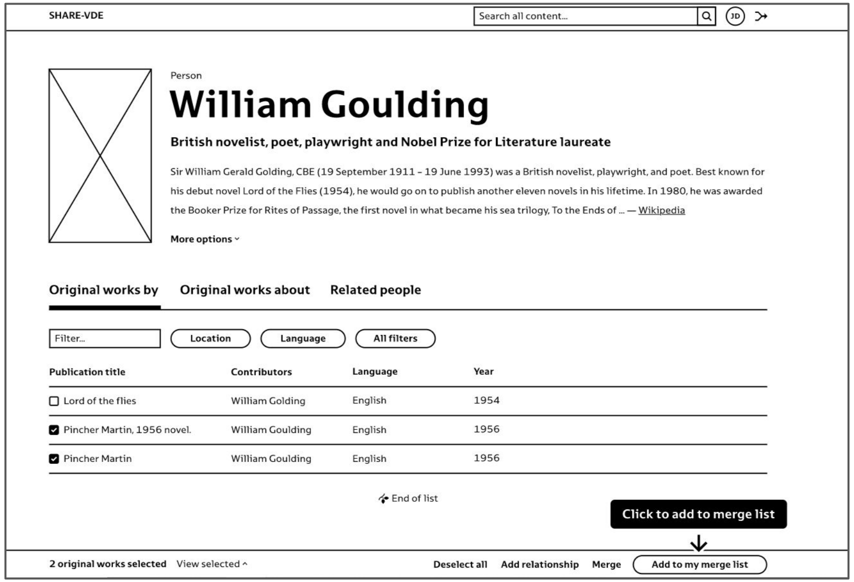


Share-VDE - FOLIO data flow



J.Cricket entity editor

From linked data publication to linked data editing



SHARE-VDE

Search all content...

Person

William Goulding

British novelist, poet, playwright and Nobel Prize for Literature laureate

Sir William Gerald Golding, CBE (19 September 1911 – 19 June 1993) was a British novelist, playwright, and poet. Best known for his debut novel Lord of the Flies (1954), he would go on to publish another eleven novels in his lifetime. In 1980, he was awarded the Booker Prize for Rites of Passage, the first novel in what became his sea trilogy, To the Ends of ... — Wikipedia

More options

Original works by | Original works about | Related people

Filter... | Location | Language | All filters

Publication title	Contributors	Language	Year
☐ Lord of the flies	William Golding	English	1954
☑ Pincher Martin, 1956 novel.	William Goulding	English	1956
☑ Pincher Martin	William Goulding	English	1956

End of list

Click to add to merge list

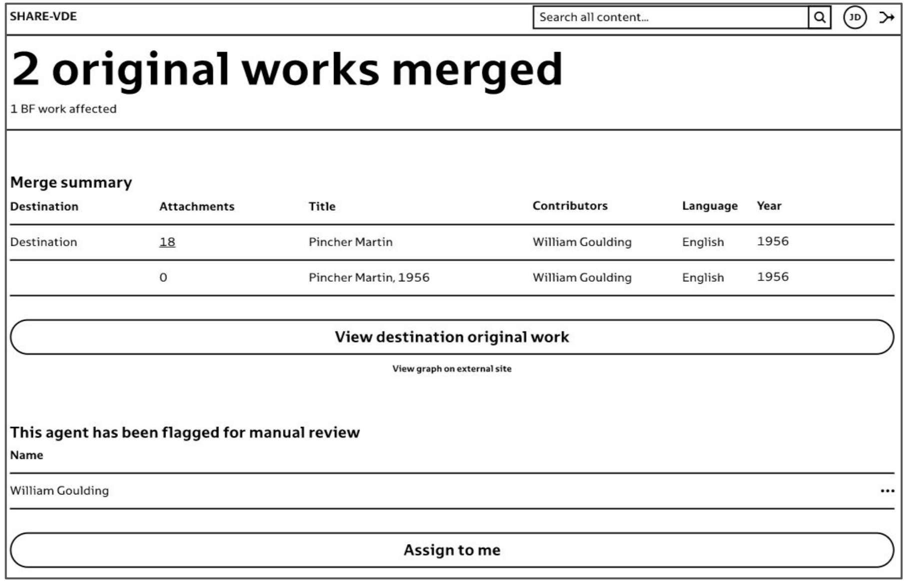
2 original works selected   View selected   Deselect all   Add relationship   Merge   Add to my merge list

The Share family platform is evolving from a discovery environment that converts traditional MARC data of libraries in Linked Open Data to an interactive authoritative source providing real services for libraries. This transition is happening through the editor named J.Cricket, that is the new application dedicated to the editing of the clusters of data in a collaborative and integrated environment.

From linked data publication to linked data editing

The editing tool J.Cricket will allow for editing the SVDE Cluster Knowledge Base, Sapientia, enabling several actions on the clusters (entities) saved in the SVDE database, including creation, modification, merge of clusters of works, of agents etc.

J.Cricket will extend authority capabilities through the integration with external data sources such as Wikidata and ISNI.



2 original works merged

1 BF work affected

Merge summary

Destination	Attachments	Title	Contributors	Language	Year
Destination	18	Pincher Martin	William Goulding	English	1956
	0	Pincher Martin, 1956	William Goulding	English	1956

This agent has been flagged for manual review

Name
William Goulding

Active participation and concrete output

Libraries members of Share-VDE and Share Family Working Groups and parallel projects are constantly contributing with their Subject Matter Experts to requirements gathering, functional analysis and feedback to developments.

Share-VDE Advisory Council and Working Groups:

- Share-VDE Advisory Council
- Sapientia Entity Identification WG
- Authority/Identifier Management Services WG
- Cluster Knowledge Base Editor WG
- User experience/User Interface WG

Share Family Working Groups:

- National bibliographies Working Group involving SVDE members and external institutions
- Italian group for the conversion UNIMARC - BIBFRAME
- discussions in the field of photo libraries and audio-visual collections

Authority/Identifier Management Services WG

The AIMS WG defines guidelines and best practices for Authority/Identifier management; defines scope and data-flow for the creation and implementation of automated services based on preliminary documentation; proposes additional use cases identified as essential for effective knowledge base management.
Group materials (meetings are on hold)

Latest outcomes: new generation of services for the authority control

- definition of use cases;
- functional analysis;
- analysis of interaction with Wikidata and ISNI (joint work with CKBE WG to design J.Cricket functionalities);
- pilot of MARC-based authority services with Stanford University Libraries;
- initial analysis of services for authority control in linked data workflows.

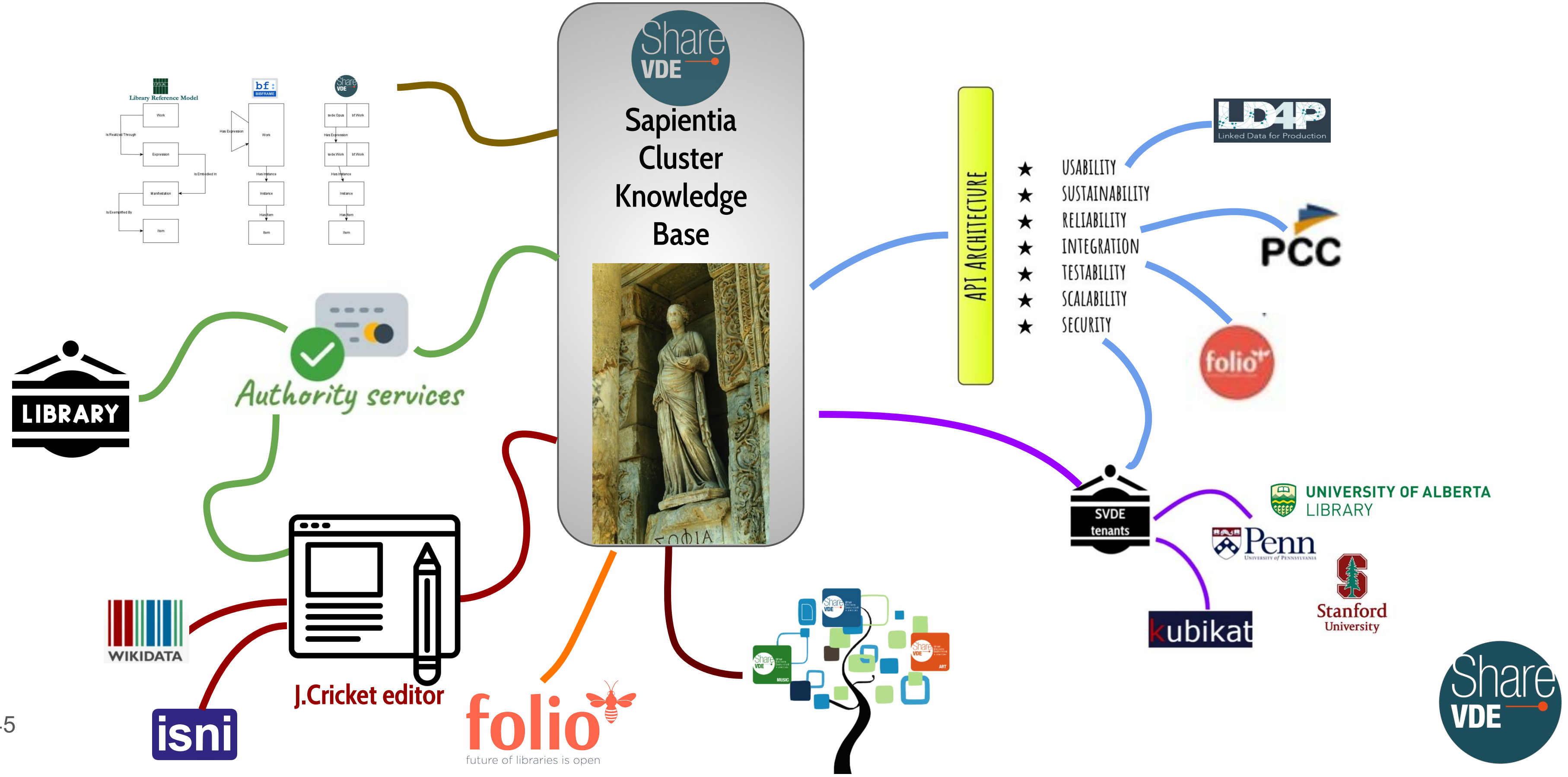
Cluster Knowledge Base Editor WG

The CKBE WG analyses how libraries interact with the *Sapientia* Cluster Knowledge Base (CKB) and their use of the J.Cricket Editor for modifying (correcting / enriching), deleting, merging and separating clusters.
Group materials (meetings are on hold)

Latest outcomes: back-end developments for J.Cricket entity editor started

- definition of use cases;
- design of manual editing features;
- analysis of interaction with Wikidata and ISNI to be incorporated into J.Cricket and authority dataflows that feed the Cluster Knowledge Base (joint work with AIMS WG to design J.Cricket functionalities);
- back-end developments started; respective front-end features will follow throughout 2022.

Towards the Share-VDE Sapientia CKB ecosystem



Next generation cataloguing

The J.Cricket editor is an example of how the Share family of initiatives is pursuing a new way of managing library cataloguing:

- aggregation of data from multiple sources
- managed through standard protocols (linked data)
- in a collaborative and integrated environment
- that makes available open data and resources
- to end users and professionals (researchers, scholars etc.)
- for reuse in the library community and beyond

J.Cricket - How does it work?

J.Cricket 1.1.0: Features Recap

- **AAA:** Authentication + Authorization + Auditing
- **Cluster Status API**
- **Edit Cluster**
  - real time notifications (through GraphQL subscriptions) about cluster property changes
- **Merge:** C1, C2, C3 => ~~C1, C2,~~ C3
  - Multiple phases: create the merge list, edit the merge list, edit clusters, request for review, approve (or deny the merge)
- **Split (Cluster):** C1 => C1, C2
  - C2 could even be a new cluster
  - Multiple phases: create the split-set, edit the split-set, edit clusters, request for review, approve (or deny the merge)
- **Dictionary API**: What are the available cluster types? Which attributes belong to a cluster type? Which relationships? Given an attribute, which is its cardinality? Is it mandatory or not?
- **Data changes synchronization across Share-VDE storages (e.g. RDF Store, Search Engine, RDBMS)**
- **Entity Event Log (aka cluster changes)**: give me the history of changes of a given cluster
- **User notifications**: for managing the merge/split review lifecycle

Edit

Share VDE

Edit - Overview

The **Edit Cluster** operation is available for **J.Cricket editors** to add, remove and amend **attributes**, **relationships** and **links** belonging to a single Entity.

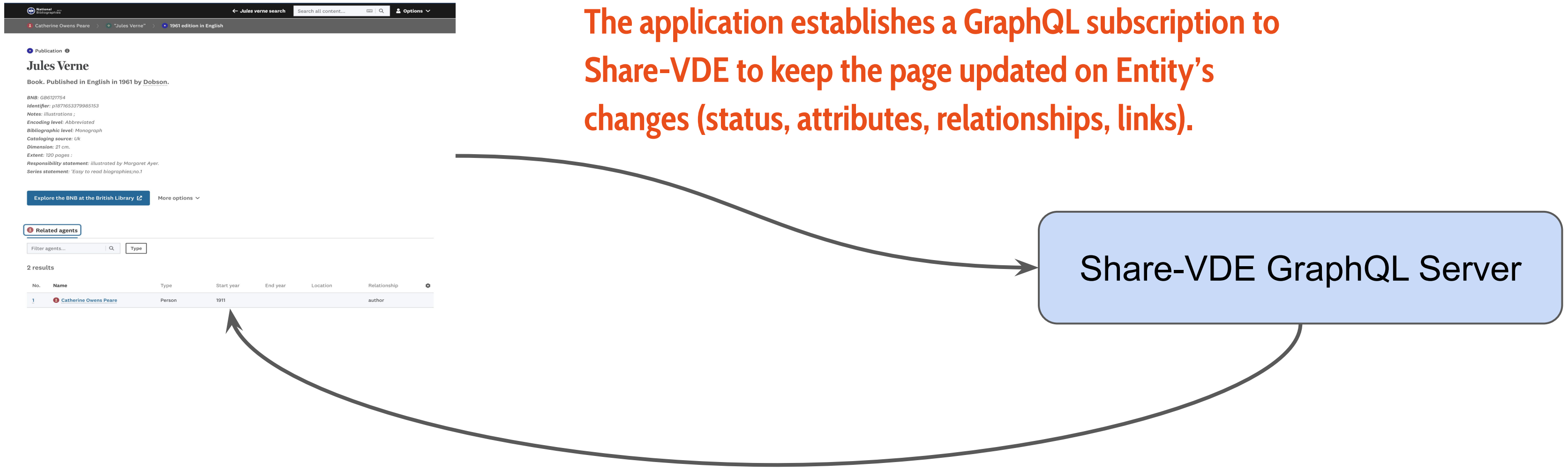
If the user is a **basic editor**, only the properties coming from the **user's provenance** will be **editable**. If the user is an **advanced editor**, the **whole Prism** (meaning all properties) will be **editable**.

We will show the use cases related to this scenario, observing the client interaction with the Share-VDE server in relation to editing **one property at a time** (<u>the recommended way</u> to implement the edit feature).

However, the server supports submitting changes for multiple properties at the same time as well, with some limitations for transient values broadcast.

Edit - An editor enters an Entity’s page

A user with the “editor” role enters an entity’s page.



Catherine Owens Peare > “Jules Verne” > 1961 edition in English

Publication

Jules Verne

Book. Published in English in 1961 by Dobson.

BNB: *GB6121754*
Identifier: *p1871653379985153*
Notes: *illustrations ;*
Encoding level: *Abbreviated*
Bibliographic level: *Monograph*
Cataloging source: *Uk*
Dimension: *21 cm.*
Extent: *120 pages :*
Responsibility statement: *illustrated by Margaret Ayer.*
Series statement: *'Easy to read biographies;no.1*

Explore the BNB at the British Library | More options

Related agents

Filter agents... | Type

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owens Peare	Person	1911			author

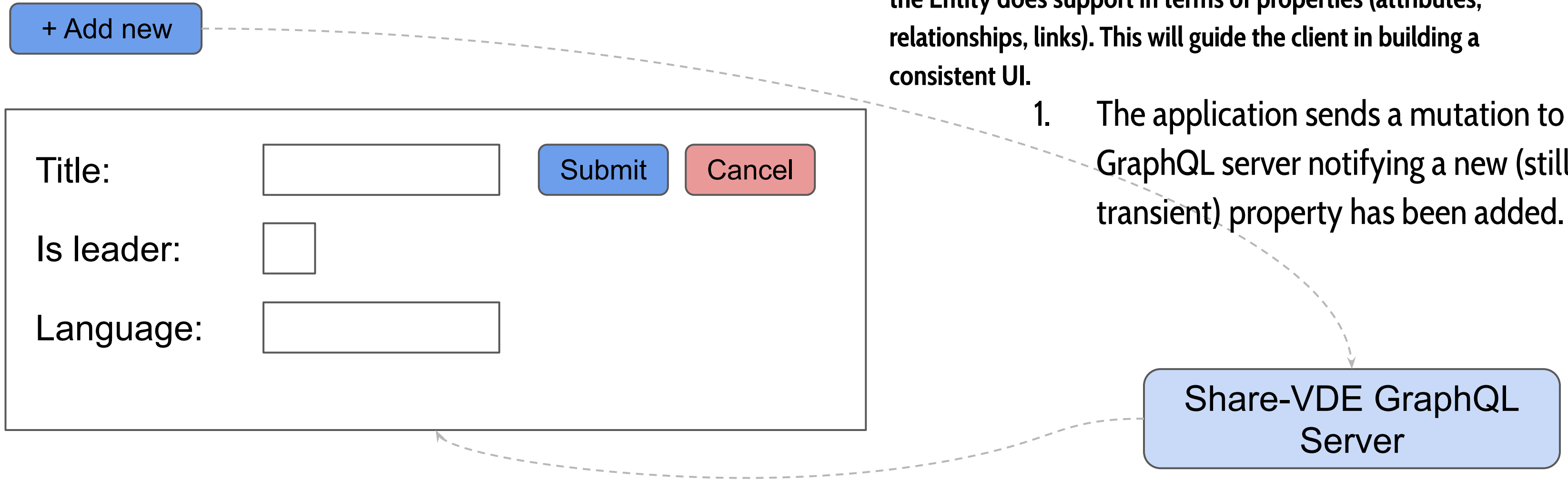
The application establishes a GraphQL subscription to Share-VDE to keep the page updated on Entity’s changes (status, attributes, relationships, links).

When the Entity changes due to someone else modifying it, the change is notified to our user’s browser as well, and the application updates the related field.

Edit - An editor adds a new property (1 / 4)

The user adds a new attribute (attribute, relationship or link)

Initially, the client application uses the Dictionary API to request the Entity's Dictionary description, so to know what the Entity does support in terms of properties (attributes, relationships, links). This will guide the client in building a consistent UI.



1. The application sends a mutation to the GraphQL server notifying a new (still transient) property has been added.

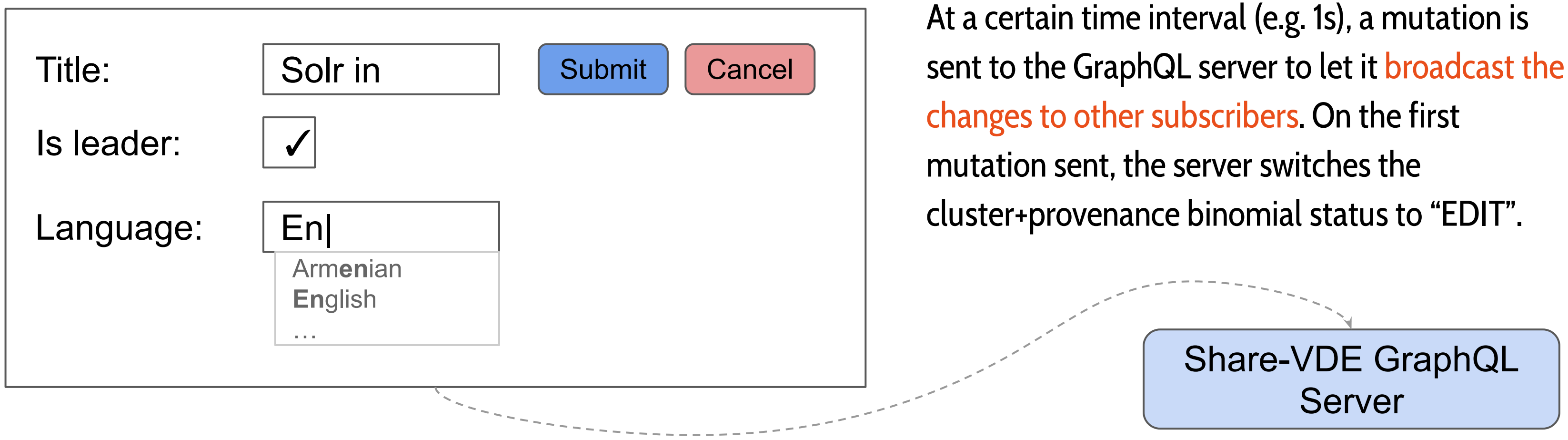
2. No persistence is made by the server, as the change is still transient, but it responds with the empty attribute containing the server-assigned identifier.

3. The client must not let the user enter values in the new field before getting that response from the server.

Edit - An editor adds a new property (2 / 4)

The user starts making changes to the new attribute.

The mechanics, despite some intrinsic differences between the property types, are the same for relationships and links as well.

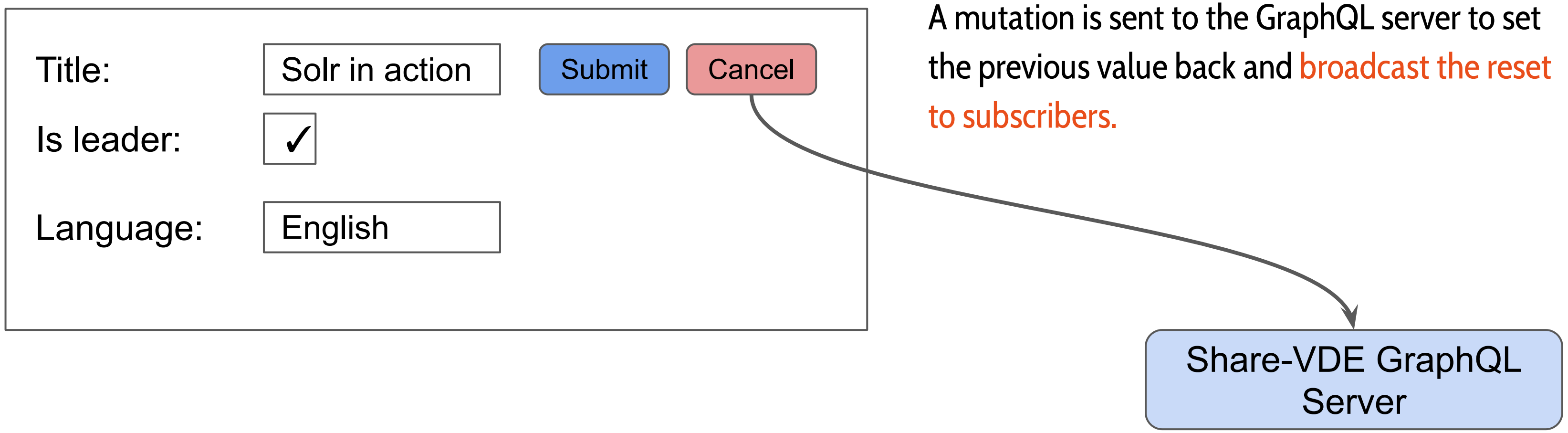


At a certain time interval (e.g. 1s), a mutation is sent to the GraphQL server to let it broadcast the changes to other subscribers. On the first mutation sent, the server switches the cluster+provenance binomial status to “EDIT”.

No persistence is made by the server, as the change is still transient.

Edit - An editor adds a new property (3 / 4)

The user may now hit the field's **Cancel** button

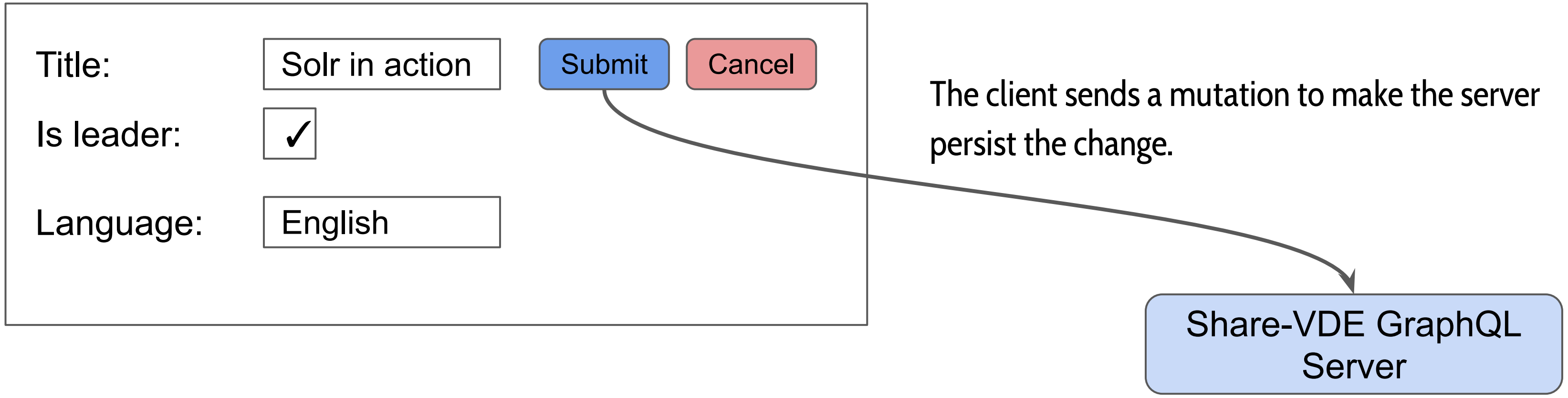


A mutation is sent to the GraphQL server to set the previous value back and broadcast the reset to subscribers.

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status shifts back to "SAVED"

Edit - An editor adds a new property (4 / 4)

The user can now hit the field's **Submit** button

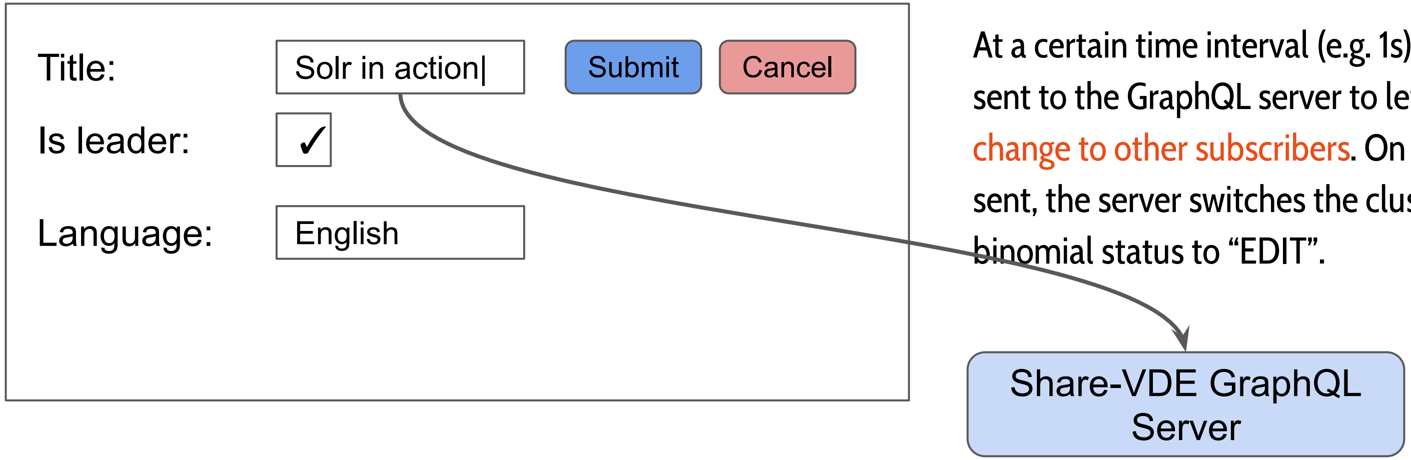


The client sends a mutation to make the server persist the change.

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status is updated to "SAVED"

Edit - A user edits an existing property (1 / 3)

The user edits a property.

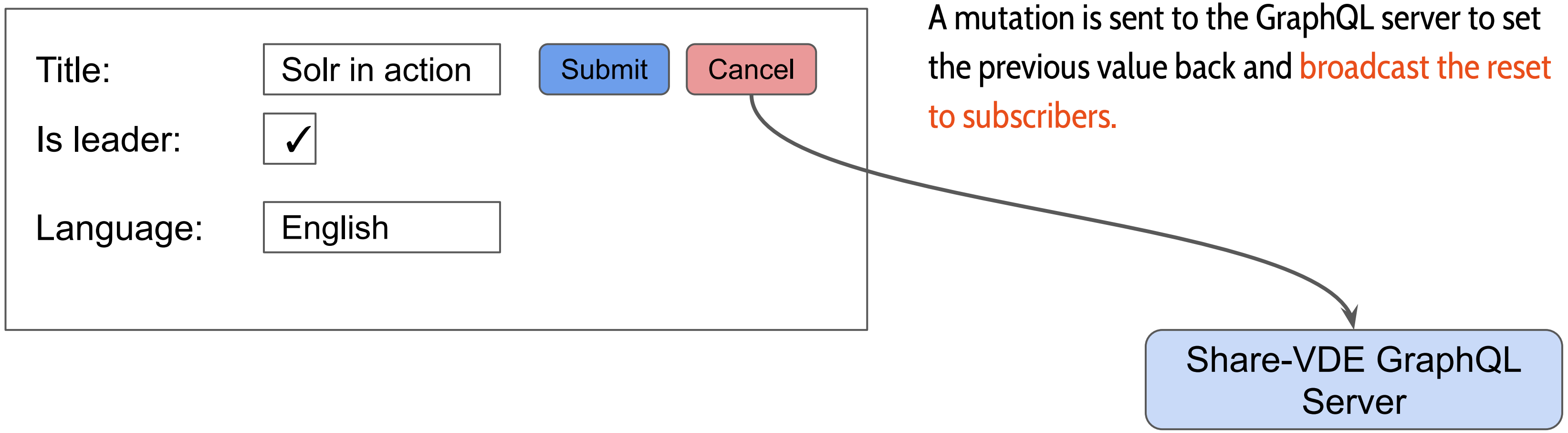


At a certain time interval (e.g. 1s), a mutation is sent to the GraphQL server to let it broadcast the change to other subscribers. On the first mutation sent, the server switches the cluster+provenance binomial status to "EDIT".

No persistence is made by the server, as the change is still transient.

Edit - A user edits an existing property (2 / 3)

The user may now hit the field's **Cancel** button

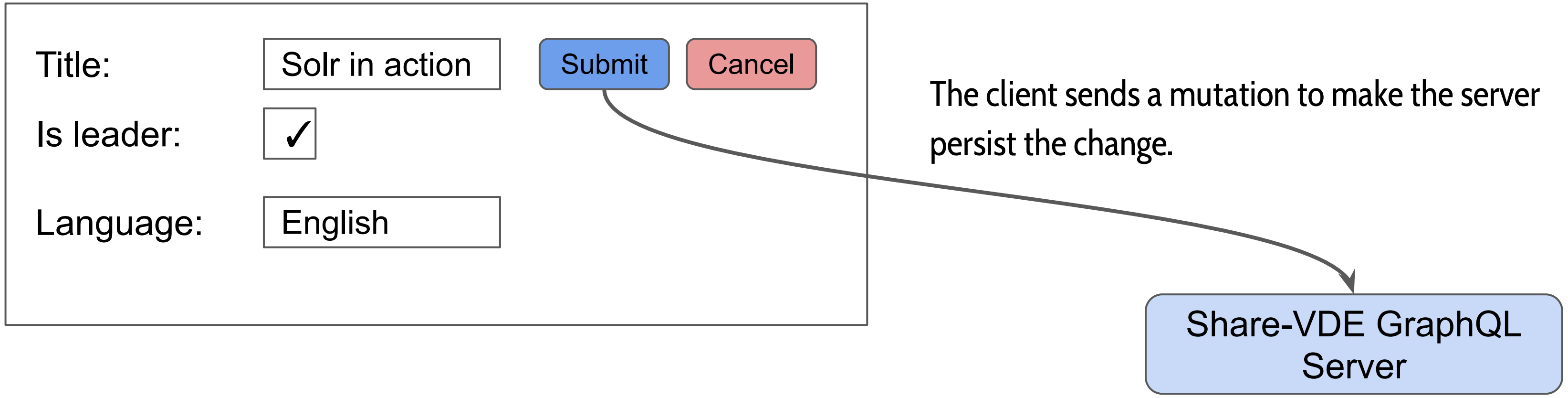


A mutation is sent to the GraphQL server to set the previous value back and broadcast the reset to subscribers.

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status shifts back to "SAVED"

Edit - A user edits an existing property (3 / 3)

The user can now hit the field's **Submit** button

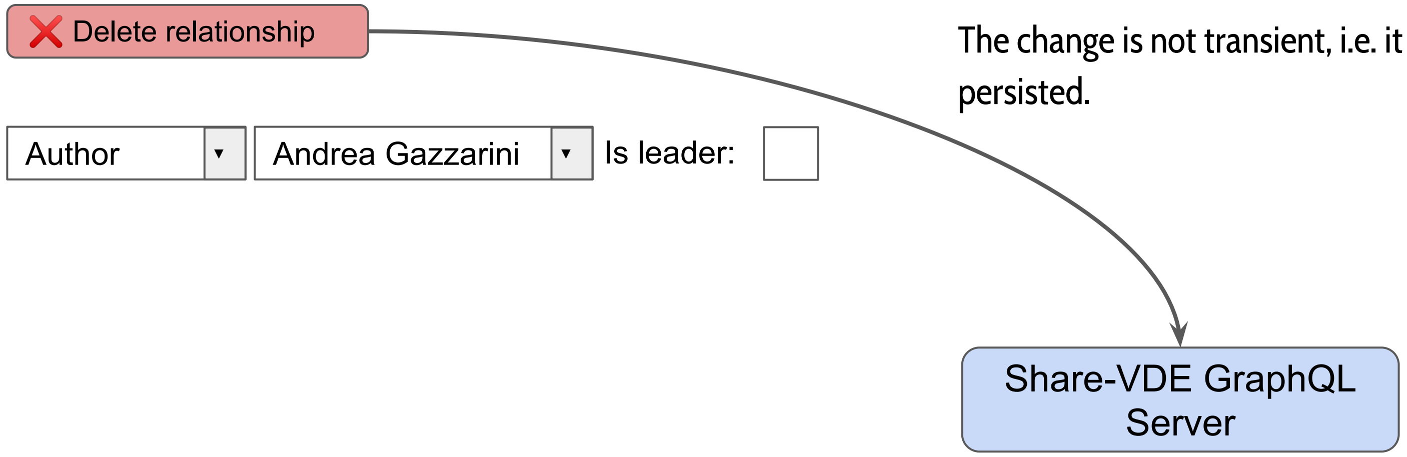


The client sends a mutation to make the server persist the change.

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status is updated to "SAVED"

Edit - An editor deletes a property

The user deletes a property.



The change is not transient, i.e. it is immediately persisted.

The GraphQL server persists the change, the mutation triggers the change broadcast to other subscribers.

The Prism status is brought back to "SAVED".

Merge

Merge - Overview (1 / 2)

The **Merge** Clusters operation is available for users with the “**advanced editor**” role to convey **one or more** source **Entities into one**, picking the source Cluster(s) properties that must be ported.

The user picks **two or more Clusters** to merge, then designates the **destination** one. The remaining Clusters are automatically marked as “**source**”. Such **destination** or **source** traits are sealed by dedicated statuses “MD” (Merge Destination) and “MS” (Merge Source) .

After that, the user can choose which properties to copy to the destination Entity.

Merge - Overview (2 / 2)

After picking all the properties to put in the destination Cluster, the user **confirms the merge** and contextually **requests a review action** by **designating** a **reviewer**. The destination Cluster shifts its status to “RN” (Review Needed).

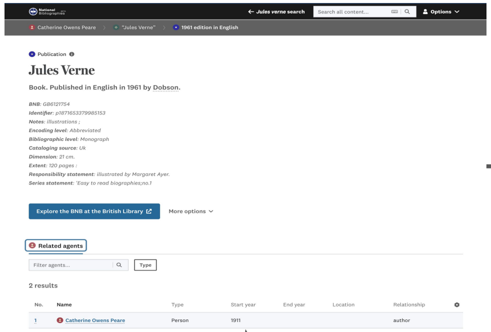
The reviewer may **approve** the merge; at that point the destination Cluster shifts its status to “SV” (**Saved**), while source Clusters acquire the status “IN” (**Invalidated**).

Invalidated Clusters will remain in the system, but they won’t be indexed anymore, i.e. they will not appear in search results.

The reviewer may even **reject** the merge; in that case the destination cluster shifts back to the “MD” status; the reviewer provides some **rejection notes** to guide the editor.

Merge - An editor enters an Entity’s page

A user with the “advanced editor” role enters an entity’s page.



Jules Verne

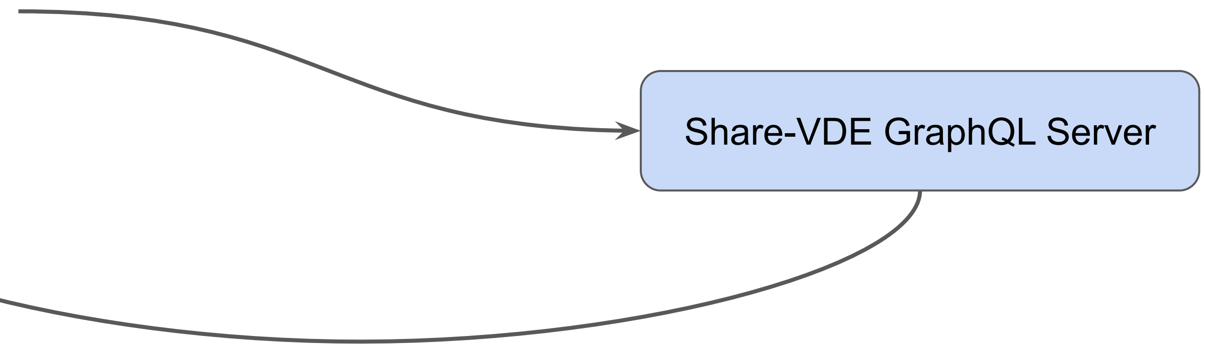
Book. Published in English in 1961 by Dobson.

BNB: *GB6121754*
Identifier: *p1871653379985153*
Notes: *illustrations ;*
Encoding level: *Abbreviated*
Bibliographic level: *Monograph*
Cataloging source: *Uk*
Dimension: *21 cm.*
Extent: *120 pages :*
Responsibility statement: *illustrated by Margaret Ayer.*
Series statement: *'Easy to read biographies;no.1*

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owens Peare	Person	1911			author

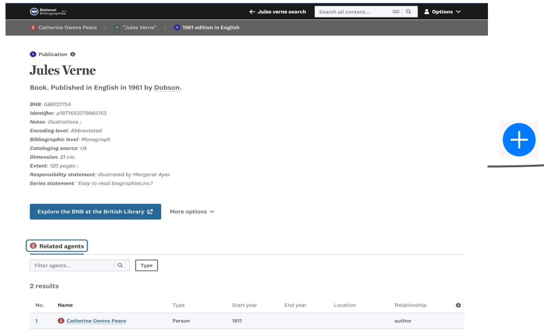
The application establishes a GraphQL subscription to Share-VDE to keep the page updated on Entity’s changes (status, attributes, relationships, links).



When the Entity changes due to someone else modifying the same Entity, the change is notified to our user’s browser as well, and the application updates the related field.

Merge - An Entity is added to the Merge List

While **browsing** entities, the user **adds them** to the **Merge List**



Publication

Jules Verne

Book. Published in English in 1961 by Dobson.

BNB: *GB6121754*
Identifier: *p1871653379985153*
Notes: *illustrations ;*
Encoding level: *Abbreviated*
Bibliographic level: *Monograph*
Cataloging source: *Uk*
Dimension: *21 cm.*
Extent: *120 pages :*
Responsibility statement: *illustrated by Margaret Ayer.*
Series statement: *'Easy to read biographies;no.1*

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owens Peare	Person	1911			author

People
- Jules Verne (16455887799)
- Julius Verneau (1234567383322)
- …

Opuses
…

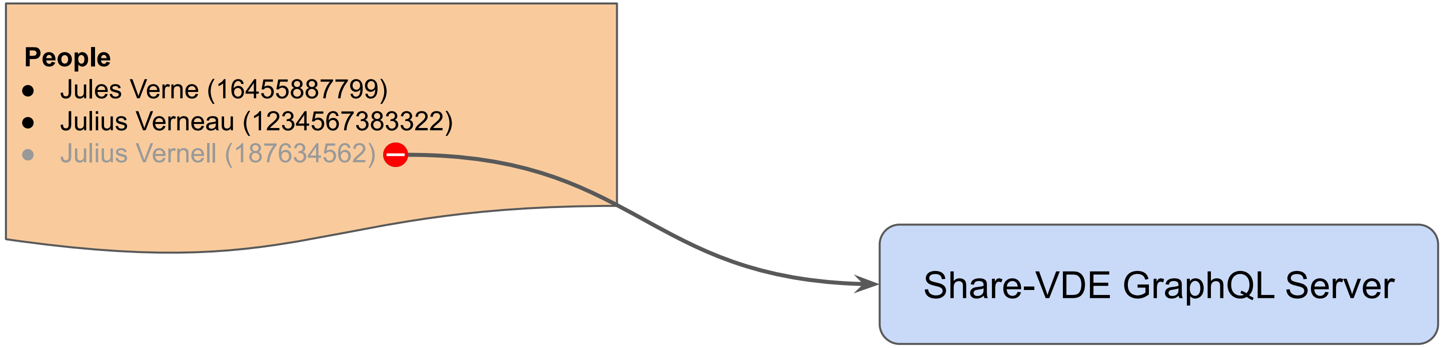
The Merge List is **subdivided** by **Entity** type.
It works just like a shopping cart.

Share-VDE GraphQL Server

A **mutation** is sent to inform the server about the action, so to let it set the entities status to "ML" (**Merge List**).

Merge - Entity is removed from the Merge List

While looking at the Merge List, the user decides to **remove** an Entity from it.

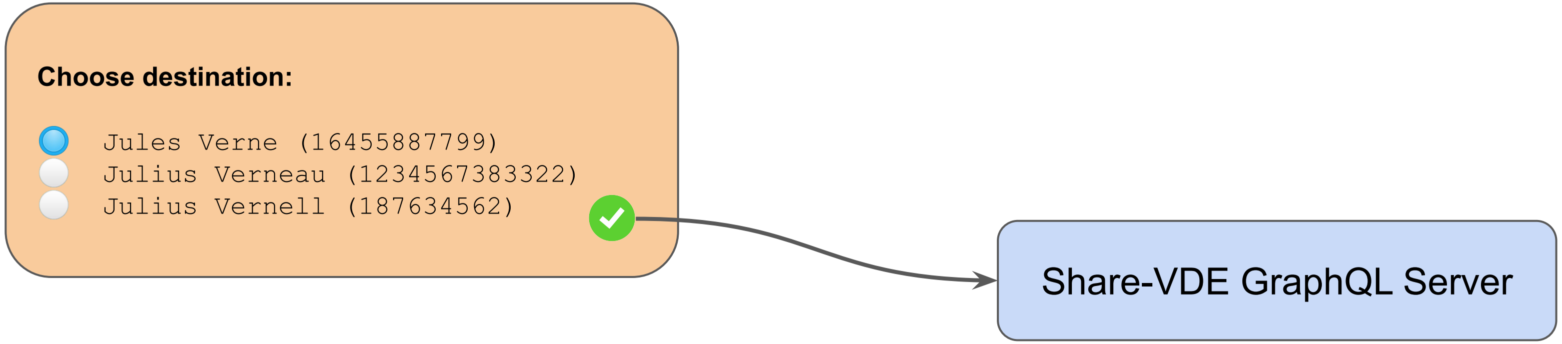


People
Jules Verne (16455887799)
Julius Verneau (1234567383322)
Julius Vernell (187634562)
Share-VDE GraphQL Server

A **mutation** is sent to inform the server about the action, so to bring back the Entity's status to "SV" (Saved).

Merge - The user confirms the Merge List

When all of the Entities of interest are in the Merge List, the user **designates the destination** Entity and **confirms** the Merge List

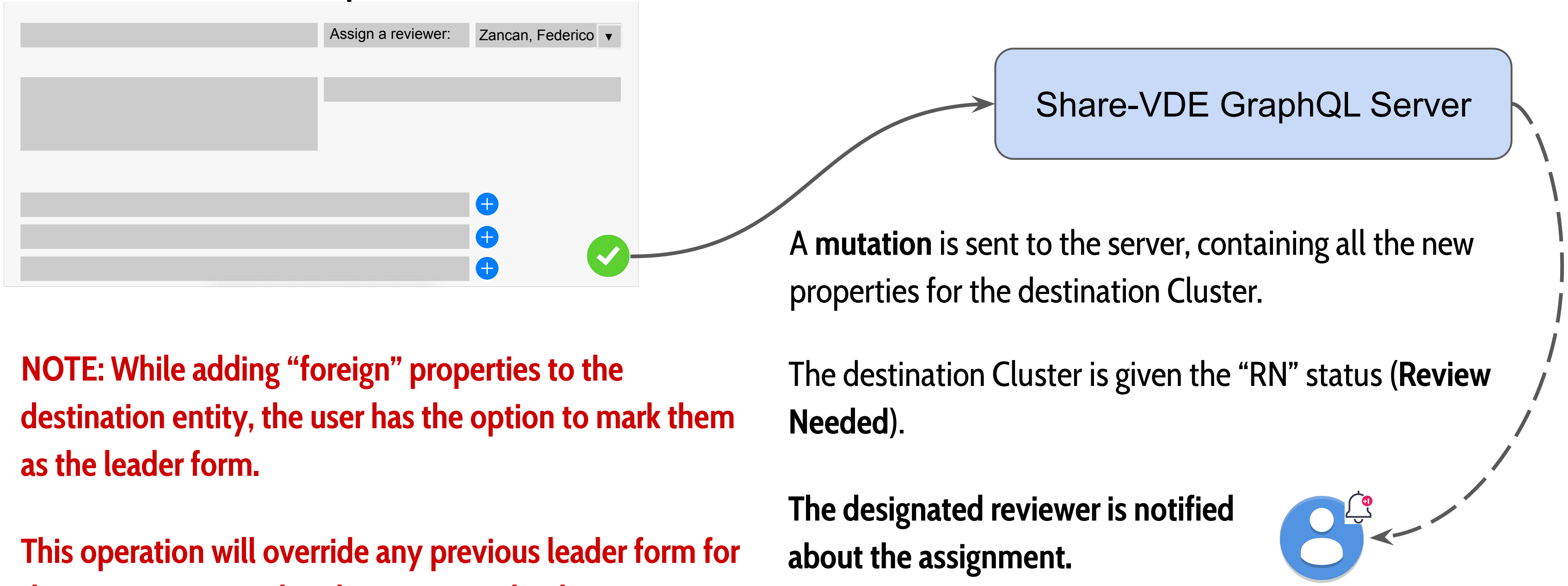

Choose destination:
● Jules Verne (16455887799)
○ Julius Verneau (1234567383322)
○ Julius Vernell (187634562)
✓
Share-VDE GraphQL Server

A **mutation** is sent to inform the server about the action.

All the **non-destination** Entities are given the “MS” (**Merge Source**) status.
The **destination** Entity acquires the “MD” (**Merge Destination**) status.

Merge - Properties are added to the destination

The user chooses what properties coming from source Entities must be **added** to the **destination**, **designates** a **reviewer**, and then **confirms** the **operation**



A **mutation** is sent to the server, containing all the new properties for the destination Cluster.

The destination Cluster is given the "RN" status (**Review Needed**).

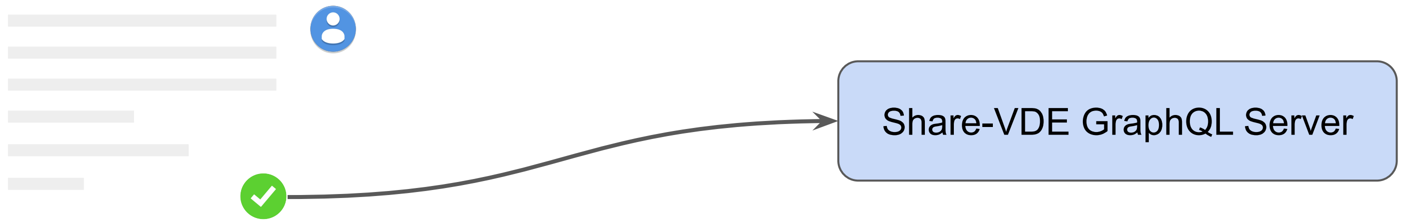
The designated reviewer is notified about the assignment.

NOTE: While adding "foreign" properties to the destination entity, the user has the option to mark them as the leader form.

This operation will override any previous leader form for the same property already present in the destination.

Merge - The Reviewer approves the Merge

The designated **Reviewer** analyzes the merge result and decides to **approve** it



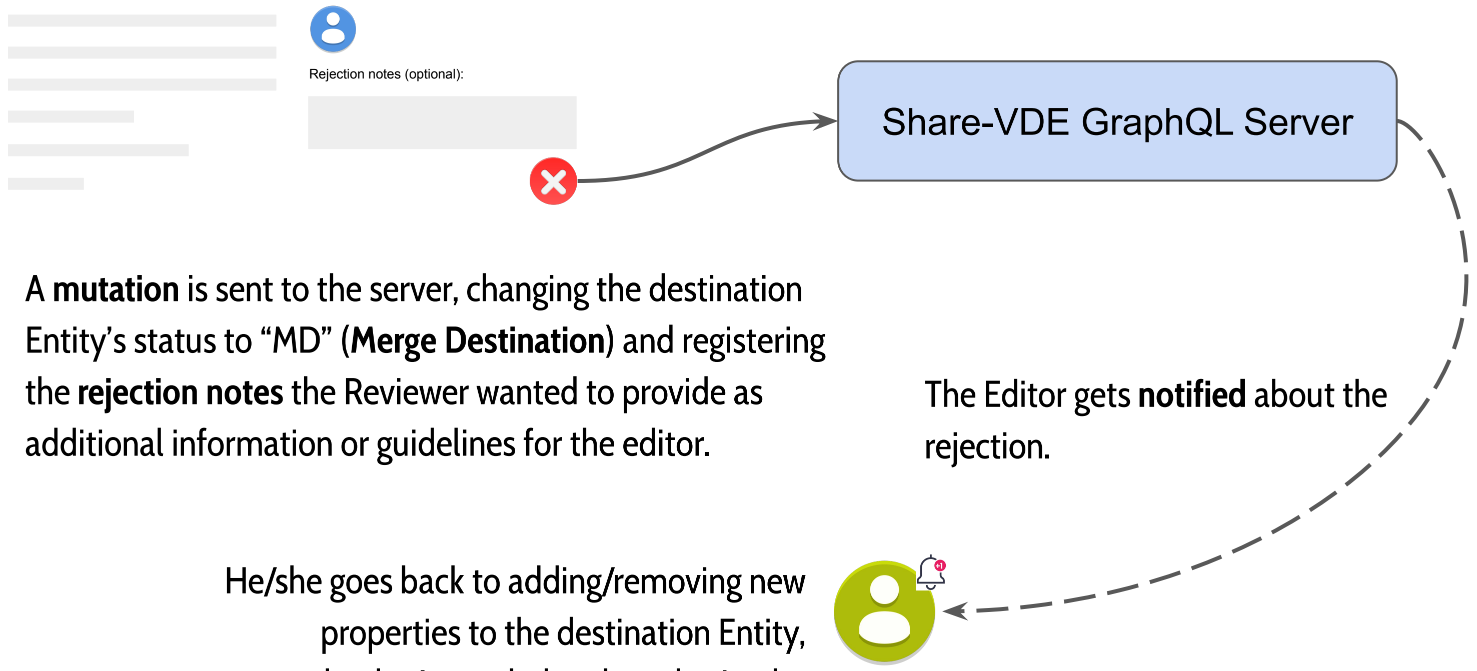
A mutation is sent to the server, changing the destination Entity's status to "SV" **(Saved)**

Source Entities pass from the "MS" (**Merge Source**) to the "IN" (**Invalidated**) status. Although they **remain** in the system, and their **URIs** are still **valid**, they are **not** part of **search results** anymore.

If visited, their pages show off in a greyed-out fashion.

Merge - The Reviewer rejects the Merge

The designated **Reviewer** analyzes the merge result and decides to reject it



A **mutation** is sent to the server, changing the destination Entity's status to "MD" (**Merge Destination**) and registering the **rejection notes** the Reviewer wanted to provide as additional information or guidelines for the editor.

The Editor gets **notified** about the rejection.

He/she goes back to adding/removing new properties to the destination Entity, amends what's needed and resubmits the review request.

Split

Split - Overview (1 / 2)

The **Split** Cluster operation is available for users with the “**advanced editor**” role to let them **move** one or more **properties** between two clusters.

The user picks the “**Giver**” and “**Receiver**” Clusters. The giver Cluster takes the “SG” (**Split Giver**) status, while the receiver takes the “SR” (**Split Receiver**) status.

The user can then **choose** the **giver’s properties** to be moved to the **receiver**.

When satisfied with the choice, the user **confirms** the **split** and contextually **requests** a **review** action by **designating** a **reviewer**. The receiver Cluster shifts its status to “RN” (**Review Needed**).

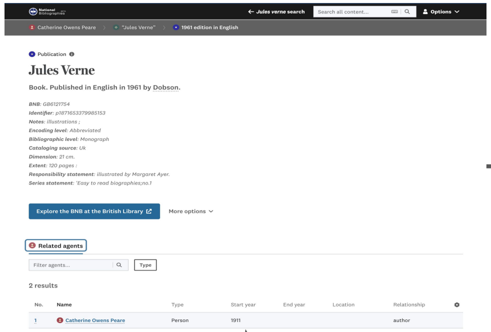
Split - Overview (2 / 2)

The reviewer may **approve** the **split**; at that point the giver and receiver Clusters shift their status to “SV” (**Saved**).

The reviewer may even **reject** the **split**; in that case the receiver cluster shifts back to the “SR” (**Split Receiver**) status; the reviewer provides some **rejection notes** to guide the editor.

Split - An editor enters an Entity’s page

A user with the “advanced editor” role enters an entity’s page.



National Bibliographies BETA | ← *Jules verne search* | Search all content... | Options

Catherine Owens Peare › “Jules Verne” › 1961 edition in English

Publication

Jules Verne

Book. Published in English in 1961 by Dobson.

BNB: *GB6121754*
Identifier: *p1871653379985153*
Notes: *illustrations ;*
Encoding level: *Abbreviated*
Bibliographic level: *Monograph*
Cataloging source: *Uk*
Dimension: *21 cm.*
Extent: *120 pages :*
Responsibility statement: *illustrated by Margaret Ayer.*
Series statement: *'Easy to read biographies;no.1*

Explore the BNB at the British Library | More options

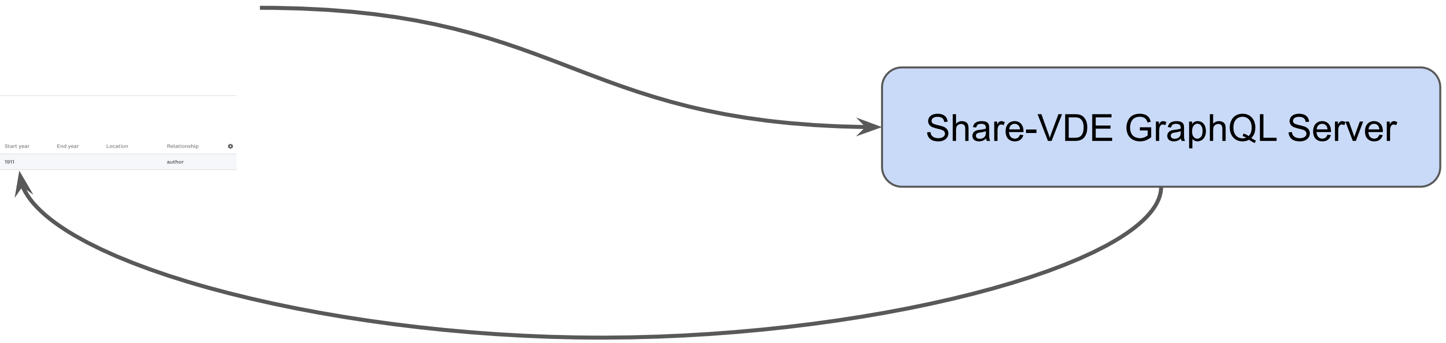
Related agents

Filter agents... | Type

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owens Peare	Person	1911			author

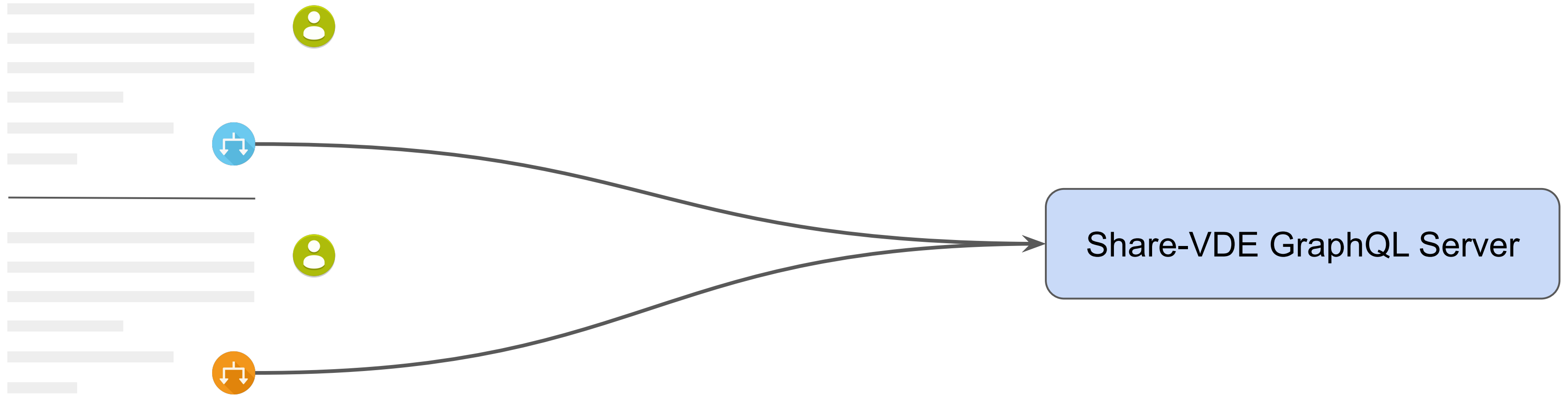
The application establishes a GraphQL subscription to Share-VDE to keep the page updated on Entity’s changes (status, attributes, relationships, links).



When the Entity changes due to someone else modifying the same Entity, the change is notified to our user’s browser as well, and the application updates the related field.

Split - The user activates the Split

The user activates the split context, designating the **giver** and the **receiver**.

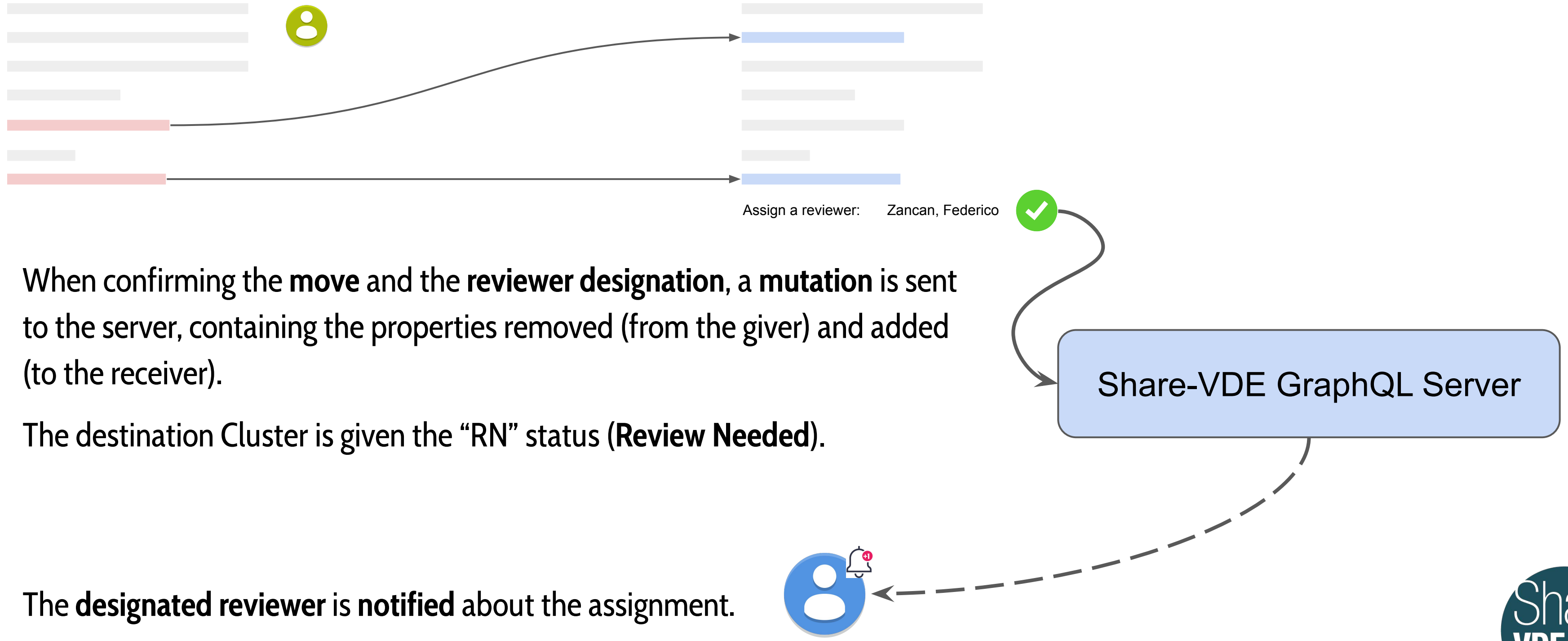


A **mutation** is sent to the server, changing the giver Entity's status to "SG" (**Split Giver**). A second mutation is sent to let the server mark the receiver Entity with the status SR (**Split Receiver**).

<u>It is important to state that both the operations can even be contextual other than separate. The important thing is: the split operation starts when the split context is confirmed.</u>

Split - The user moves the properties

The user moves the giver's properties to the receiver.



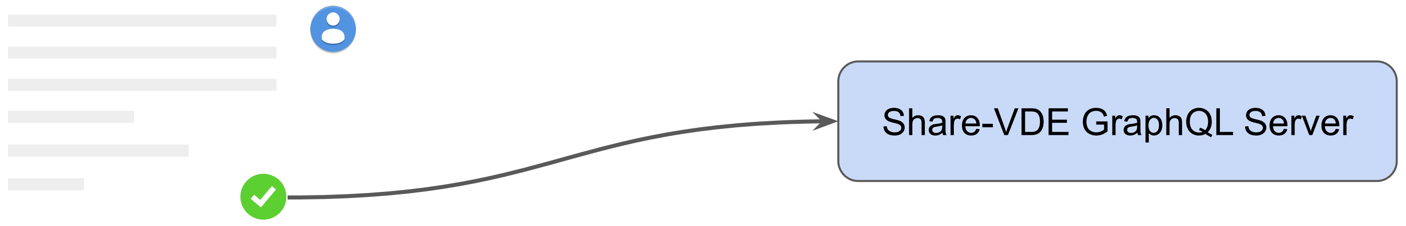
When confirming the **move** and the **reviewer designation**, a **mutation** is sent to the server, containing the properties removed (from the giver) and added (to the receiver).

The destination Cluster is given the "RN" status (**Review Needed**).

The **designated reviewer** is **notified** about the assignment.

Split - The Reviewer approves the Split

The designated Reviewer analyzes the properties that have been moved and decides to give the **approval**



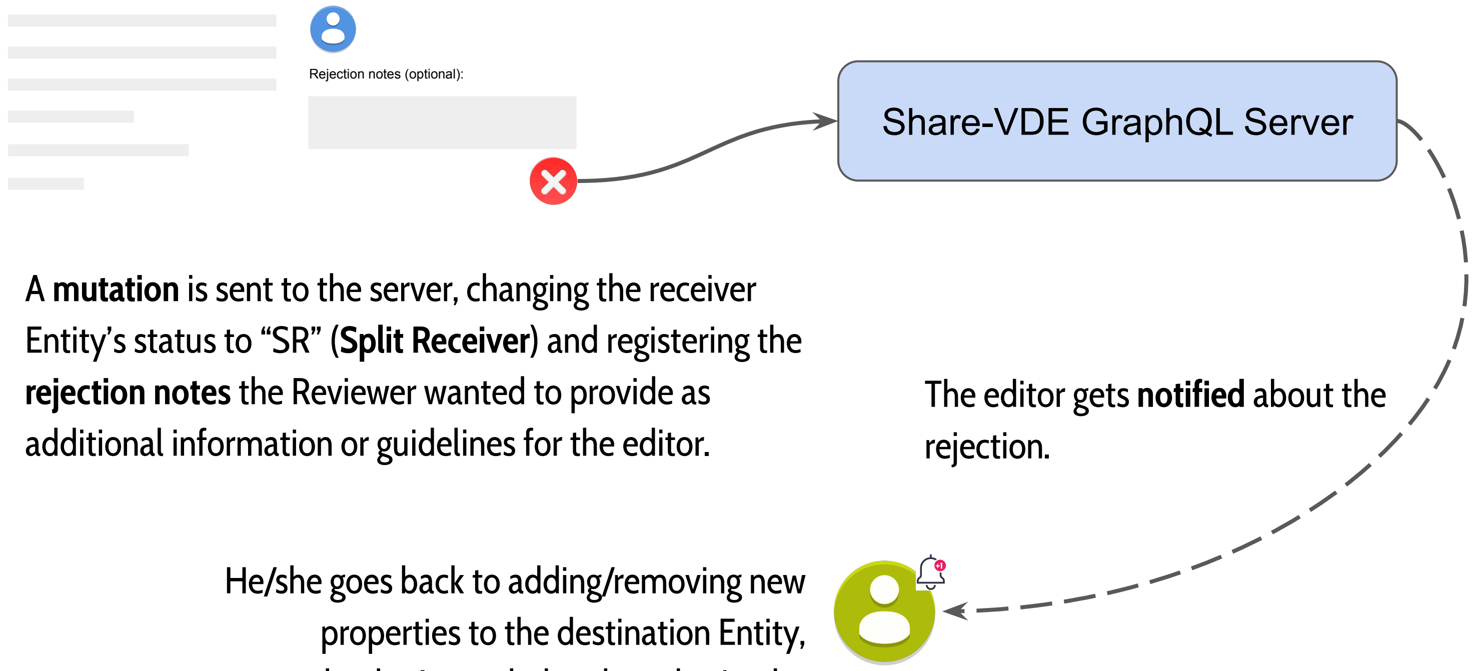
A **mutation** is sent to the server, changing the giver’s and the receiver’s statuses to “SV” (**Saved**).

The **giver** is now available to the world, **deprived** of the yielded properties.

The **receiver** is now available to the world, **enriched** of the given properties.

Split - The Reviewer rejects the Split

The designated Reviewer analyzes the split result and decides to **reject** it



A **mutation** is sent to the server, changing the receiver Entity's status to "SR" (**Split Receiver**) and registering the **rejection notes** the Reviewer wanted to provide as additional information or guidelines for the editor.

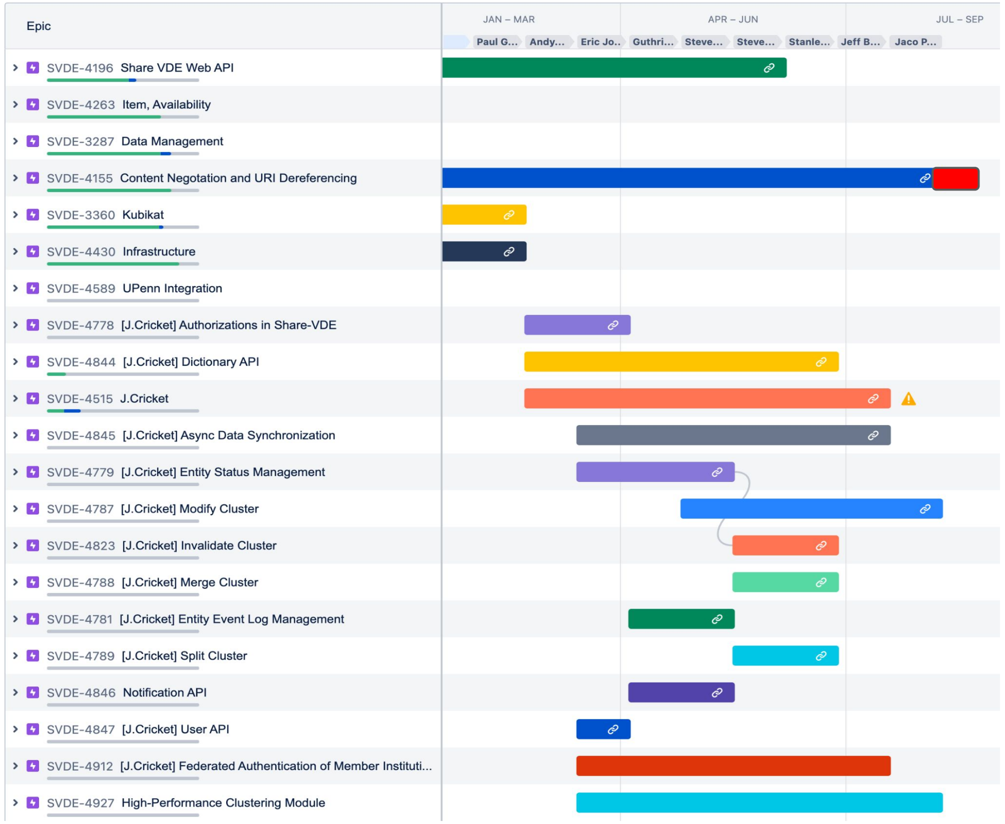
The editor gets **notified** about the rejection.

He/she goes back to adding/removing new properties to the destination Entity, amends what's needed and resubmits the review request.

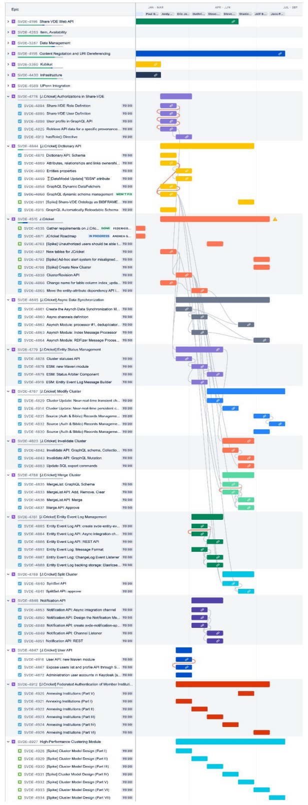
J.Cricket - Postponed Features

- Create new Cluster
- Split cluster outputs n clusters (n >= 2)
- Unauthorized users should be able to request changes to entities
- Ad-hoc alert system for misaligned clusters with respect to bibliographic records

J.Cricket roadmap



Epic
SVDE-4196 Share VDE Web API
SVDE-4263 Item, Availability
SVDE-3287 Data Management
SVDE-4155 Content Negotation and URI Dereferencing
SVDE-3360 Kubikat
SVDE-4430 Infrastructure
SVDE-4589 UPenn Integration
SVDE-4778 [J.Cricket] Authorizations in Share-VDE
SVDE-4844 [J.Cricket] Dictionary API
SVDE-4515 J.Cricket
SVDE-4845 [J.Cricket] Async Data Synchronization
SVDE-4779 [J.Cricket] Entity Status Management
SVDE-4787 [J.Cricket] Modify Cluster
SVDE-4823 [J.Cricket] Invalidate Cluster
SVDE-4788 [J.Cricket] Merge Cluster
SVDE-4781 [J.Cricket] Entity Event Log Management
SVDE-4789 [J.Cricket] Split Cluster
SVDE-4846 Notification API
SVDE-4847 [J.Cricket] User API
SVDE-4912 [J.Cricket] Federated Authentication of Member Instituti...
SVDE-4927 High-Performance Clustering Module



Share VDE Virtual Discovery Environment

WOLFcon 2022
Aug. 31 – Sept. 2
Hamburg, Germany

Thank you

tiziana.possemato@atcult.it
tiziana.possemato@casalini.it

https://wiki.svde.org/
https://svde.org
info@svde.org