SIP Expert GUI Generic Use Cases and Requirements

This document described the generalised uses cases and related requirements on components of the Spatial Information Platform (SIP) for the SWITCH-ON project that have been derived from D2.1 “Spatial Information Platform and Application Programming Interface”, D2.6 “Standard Information Model: Definition and Description” and user requirements expressed by WP3 “New forms of Research” and WP4 “Products and services”.

- **SIM and Meta-Data Requirements**
  Requirements on the **Standard Information Model** (cids Integration Base) and available meta-data: How does the SIM support this use case? By which classes and attributes? What are the mandatory attributes? What are expected and default values? Which dynamic catalogue statements have to be implemented?
- **SIP Back-End Requirements**
  Requirements on **Meta-Data Repositories** and **SIP Expert Catalogue Management APIs** (cids Domain Server, Broker, etc): Which custom searches and actions have to be implemented? Which additional functionalities of the (REST) API are needed?
- **SIP Expert GUI Requirements**
  Requirements on **SIP Expert Catalogue and Data Management GUIs** (cids Navigator): Requirements on Renderer, Editors, Wizards, etc.
- **Requirements on other Components**
  Requirements on other SIP Components, e.g. Processing and (Meta-)Data Import Tools.

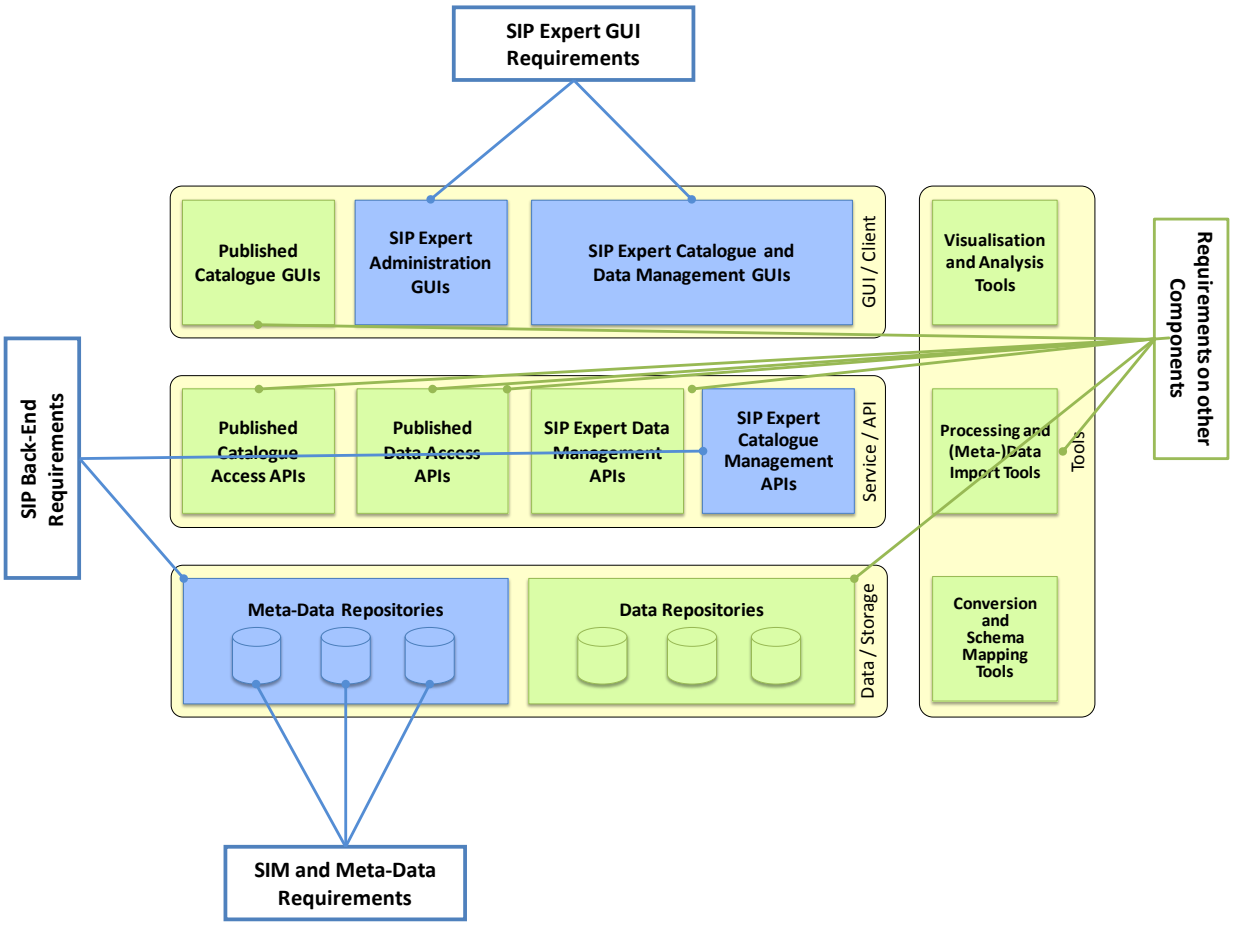

Figure 1: Relation of Requirements to the SIP Architecture

Figure 1 shows the relations of requirements described in this annex to the architecture of the SIP introduced in D2.1 (D2.1, 2014). Obviously, the focus is on components related to the SIP Expert GUI.

Find Data by queries on Resource Meta-Data

The user wants to find data by performing a basic query on meta-data available in the SIP. The user wants to search for data by spatial and temporal extent, by keywords, by topic category, by description and title or by a combination of different search parameters.

SIM and Meta-Data Requirements

Mandatory attributes required for this search type are:

- **Geospatial Search**: *Resource/geom*.
  The *geom* attribute leverages well know PostGIS spatial search and is available for all resources. There is also an attribute *location* which can be used for search by location name (Country, Region, ...).
- **Temporal Search**: *Resource/toDate*, *Resource/fromDate*
  Temporal search shall be limited on *fromDate* and *toDate* of the resource data. If *toDate* is empty, this means that the collection of data is ongoing. However if *fromData* and *toData* are both empty, it means that no information about temporal coverage is available. This has to be considered when implementing the temporal extent queries.
- **Keyword Search**: Resource/tags
  Keywords are all tags belonging to the tag groups “INSPIRE themes 1.0” and “keywords – open”. There is no dedicated attribute for Keywords in the Resource Class. Keywords Tags are stored in the *tags []* attribute.
- **Topic Category:** *Resource/topicCategory*
  Similar to keywords, with the exception that there is only one topic category assigned with a resource. Available for most imported resources.
- **Text Search**: *Resource/name*, *Resource/description*
  Full text search on resource names and description shall be supported.

The *tags* attribute of the Resources class can contain multiple tags from different tags groups. Usually, tags of the same tag group (like Keyword Tags) which can occur more than once are stored in the *tags* array rather than in a dedicated attribute.

SIP Back-End Requirements

Implementation of **custom search(es)** in cids Domain Server is necessary (cids-custom-switchon-server on GitHub[1]):

- Combination of search parameters (AND) must be possible.
- Extension for additional search parameters should be foreseen right from the beginning.
- The results of these basic queries are always complete Resource objects (including all child objects like Representation and Metadata).

Since the possible query values for Keywords are known (all tags belonging to the “INSPIRE themes 1.0” and “keywords – open” tag groups), auto complete functionality should be provided.

[1] https://github.com/switchonproject/cids-custom-switchon-server

SIP Expert GUI Requirements

For the basic user search a simple dialog or form (**Basic Search Form**) has to be implemented. A combination of search parameters must be supported by the GUI. All input fields should reside in one panel. The selection of a specific class is not necessary; the search is always performed for Resource objects.

- **Geospatial search:**
  Geometry-based Geospatial search is based on entering coordinates. The coordinates of the search area (e.g. Bounding Box) can be selected with cismap. The coordinates can be shown in the search form. Search by location name depends on the availability of location *attributes*. Geospatial search by location name should present the user with either a pre-filled combo box (all tags of the "location" tag group that are actually used in resources) with auto complete decorator[2].
- **Temporal search:**
  From-date and *toData* input fields with calendar widget support. It should be possible to set *toData* to "now" (search for empty *toDate* attributes)
- **Keyword search:**
  Although there are two types of keywords (closed list of INSPIRE keywords and open list of free keywords) only o*ne combo b*ox wit*h auto* complete functionality should be provided.
  Furthermore, the keywords list box should only show *keywords* that are actually used by resources.
- **Topic Categories**
  There is a small and fixed list of INSPIRE topic categories which should be selectable from a combo box.
- **Full te*xt* search**
  Simple text field and a checkbox to select whether to search in names and descriptions of resources (default text search is only in names).

Requirements on other Components

If geographical search by location name (*location* attribute) is needed, a **reverse geocoding component** which uses the spatial extent of the resource (*geom* attribute) to populate the *location* attribute is needed. It could be implemented as cids trigger or cids server action that is either executed whenever a new Resource Object is added (automatically) or when Meta-Data Import Wizard. Resources imported by batch SQL statements have to be processed separately only once.

Another possibility is to import a GeoNames database[3] into the Integration Base (PostgreSQL with PostGIS) and perform the reverse geocoding locally. However, it has to be noted that all GeoNames Features (including countries) have only point geometries.

Find Data by Browsing Meta-Data

The user wants to find data by browsing a catalogue of Resources. Data should be categorised according to location, type, data collection, etc.

[2] https://code.google.com/p/java-autocomplete-decorators/

[3] http://download.geonames.org/export/dump/

SIM and Meta-Data Requirements

The data repository[4] of the Virtual Water-Science Laboratory is structured according to *geography*, e.g. "Level A Pan Europe" and Level "B Selected Subbasins". For each geographic area the repository is furthermore structured based on *hydrological concepts*, e.g. discharge, soils.

To be able to replicate this structure in the SIP Catalogue, Resources are equipped with a *geography* attribute and *"*hydrological concept" and "catchments" *tags*. Dynamic catalogue statements taking those tags into account have been written.

This information is only available for repurposed data or data resulting from experiments, thus data generated within SWITCH-ON. Imported resources haven't been categorised according to this schema, thus the respective tags are not available. For categorising the imported resources, the following attributes are available:

- By collection of datasets: *Resource/collection*
  This tag is available for all imported data sources.
- By geographical location: *Resource/location*
  The location attribute must not be confused with the geography attributes which is only used to distinguish between "Level A" and "Level B" datasets. Fine grained location information is not available for all datasets but "world" or "Europe". Reverse geocoding could be applied as described in section 3.1.4 to populate the location attribute.
  Moreover, dynamic catalogue statements which consider also the hierarchy of the location (Continent, Country, Region) based on a (local) GeoNames Database and the spatial extent of the resource could be generated.

Figure 2 shows the current catalogue structure in the SIP Expert GUI which is constructed with help of the attributes and tags *collection, geography* and *hydrological concept*. The distinction between "external" and "internal" resources can be made with help of the *type* attribute of the resource (e.g. "external data", "repurposed data", ...).

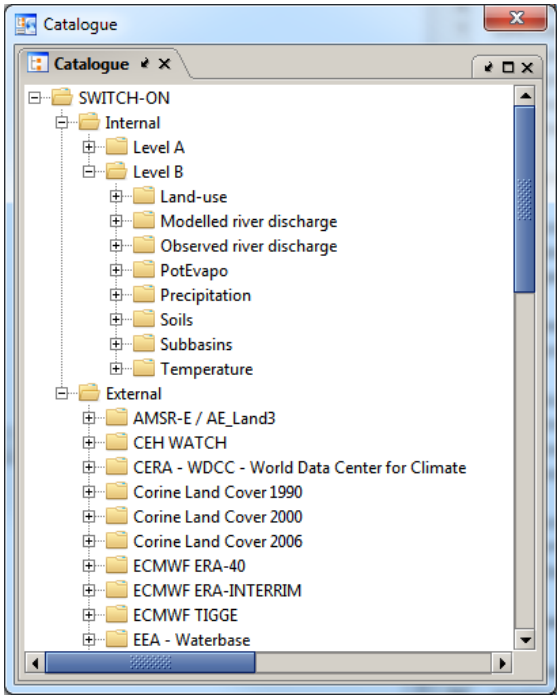

Figure 2: Catalogue Structure

[4] http://www.water-switch-on.eu/dataset.html

SIP Back-End Requirements

There are no specific SIP Back-End Requirements as this functionality is already supported by dynamic catalogue statements. Of course, the respective SQL queries for dynamic catalogue statements have to be developed.

SIP Expert GUI Requirements

There are no specific SIP Expert GUI Requirements as this functionality is already supported by the cids catalogue.

Requirements on other Components

Meta-Data stored in the Meta-Data Repository is made available via OGC CSW (pyCSW[5]). The catalogue structure used in the SIP should also be exposed by CSW. Thus, attributes and tags for categorising resources have to be considered in the SQL Views for pyCSW.

Assess usefulness of resources by viewing basic Meta-Data

Once found (either by search or browsing), the user wants to assess the usefulness of resources by viewing basic meta-data.

"Basic-Meta data" refers to the relational part of the SIM and currently available dynamic tag extensions and tag groups, respectively. Visualising the relationship between resources or accessing resource data (Representation) is addressed in another use case.

SIM and Meta-Data Requirements

The SIM defines several objects with many different attributes and relationships to other objects. Not all of them are actually relevant for the end user. The visualisation should therefore not try to replicate the complete structure of a Resource Object but rather focus on important and available attributes.

SIP Back-End Requirements

Currently, there are no specific requirements on searches or the entities API. Complete Resource Objects including complete nested sub objects (especially tags) have to be returned by the API.

SIP Expert GUI Requirements

Object Renderers for each object type (cids class) have to be implemented. Some Renderers for nested objects are only used to within other Renderers. The contact object for example is only rendered within the respective Resource Renderer and Metadata Renderer.

Some attributes are used for internal processing only (e.g. *type*, *collection*, ...) and are not of general interest for the user. They should not be shown. The Resource Renderer should furthermore present a "flat" view of the resource although a resource may have multiple meta-data and representation objects. This means, that objects which can occur more than once (Representation and Metadata) should be shown on one page or panel instead of different tabs.

The Basic Resource Renderer could be inspired by the INSPIRE Meta-Data Editor[6] (Figure 3).

[5] http://pycsw.org/

[6] http://inspire-geoportal.ec.europa.eu/EUOSMEGEOPORTAL/

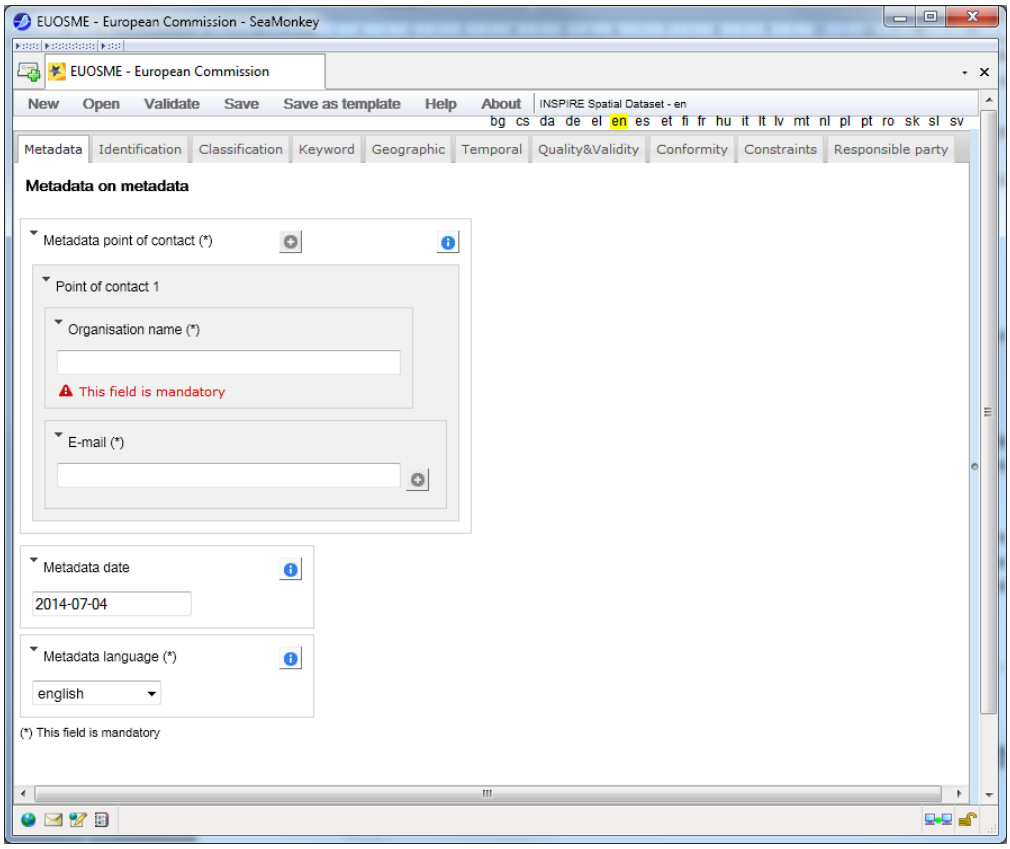

Figure 3: INSPIRE Meta-Data Editor

Different **categories** of SIP meta-data should be organised in tabs as shown in Figure 3. The following categories, that are inspired by but not identical to ISO 19115 categories, can be realised as distinct pages or tabs in cids SIP Expert GUI. Related attributes of the Resource Object are also mentioned.

- **Description**: *Resource/description*, *tags* (keywords), topicCategories
- **Contact Information**: *Resource/contact* (0...n), Metadata/Contact
- **Representation (Data)**: *Resource/representation* of type "**original data**" (0...n): *Type, name, description, link, ...*
- **Geographic**: *geom, location, (spatialScale, spatialResolution) +* cismap preview
- **Temporal**: *creation- ,publication- ,lastModoficationDate, fromDate, toDate*
- **License**: *conformity*, *accessConstraints*, *accessLimitation*, *licenseStatement* (as description)
- **Additional Metadata**: *Resource/Metadata* (0...n) of type "origin meta-data".

Note: *spatialScale*, *spatialResolution* and *temporalResolution* are currently assigned with Representation object not with Resource objects. There is good reason for this: Resource data may be available in different resolutions (Example: Corine Land Cover 1990 raster data[7]) and/or preview or aggregated representations of a resource have a different spatial and temporal resolution as the original representation. The "**Geographic**" and "**Temporal**" pages should therefore show the attributes (e.g. comma separated) of all representation of type "original data". In most cases there is only one "original data" representation available. Even if multiple "original data" representations are available (e.g. WFS and WMS), per definition the spatial and temporal should not be different.

Requirements on other Components

Currently, there are no Requirements on other Components.

[7] http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-1990-raster-3/

Assess usefulness of resources by viewing Quality and basic Lineage Meta-Data

Once found (either by search or browsing), the user wants to assess the usefulness of resources by viewing meta-data on data quality and general lineage meta-data.

Quality and Lineage meta-data is stored in the SIM as dynamic content extension.

SIM and Meta-Data Requirements

Meta-Data on data quality is represented by a Metadata Object of the type "quality meta-data". The relevant attributes are:

- *Metadata/description*: free text describing the data quality
- *Metadata/content:* structured meta data on data quality document, e.g. JSON or XML according to a commonly agreed schema (currently not available)
- *Metadata/contentLocation:* Link to an arbitrary structured or unstructured document, website, etc. that provides information ion data quality

Lineage Meta-Data is either represented by a Metadata Object of type "lineage meta-data" or by a Relationship Object. A dedicated Relationship Object is provided only if a relationship between resources registered in the SIP can be established, e.g. when resources have been repurposed or used in an experiment. The relevant attributes are:

- *Metadata/description*: free text describing the lineage information
- *Metadata/content:* structured meta data on the lineage , e.g. JSON or XML according to a commonly agreed schema (currently not available)
- *Metadata/contentLocation:* Link to an arbitrary structured or unstructured document, website, etc. that provides lineage information, e.g. link to a protocol of an experiment

SIP Back-End Requirements

Quality and simple lineage information is available as part of a resource object (*metadata[]* attribute) and thus can be accessed via the default entities API.

SIP Expert GUI Requirements

The Resource Renderer has to be extended. Preferably, all types of additional meta-data except for "basic meta-data" are shown on the same page (Additional Metadata Tab) in an accordion GUI widget[8]. Each item of the accordion should be realised as new instance of a reusable **Metadata Renderer.**

While the complete structure and all attributes of a meta-data object including the nested contact information object are available in the Attribute View (Figure 4) in cids Navigator, the Metadata Renderer should only show the most important information. Depending on the availability of attributes, it should show:

- Always the *name* as title of the accordion,
- the type of the Metadata Object as Icon in the title bar of the accordion,
- the *description* as label or text area and
- An Icon indicating the *contentType* and an additional link as clickable label or button, if *contentLocation* is available or
- the (formatted) *content* as label or JEditorPane with HTML-rendering capabilities.

[8] http://en.wikipedia.org/wiki/Accordion%28GUI%29

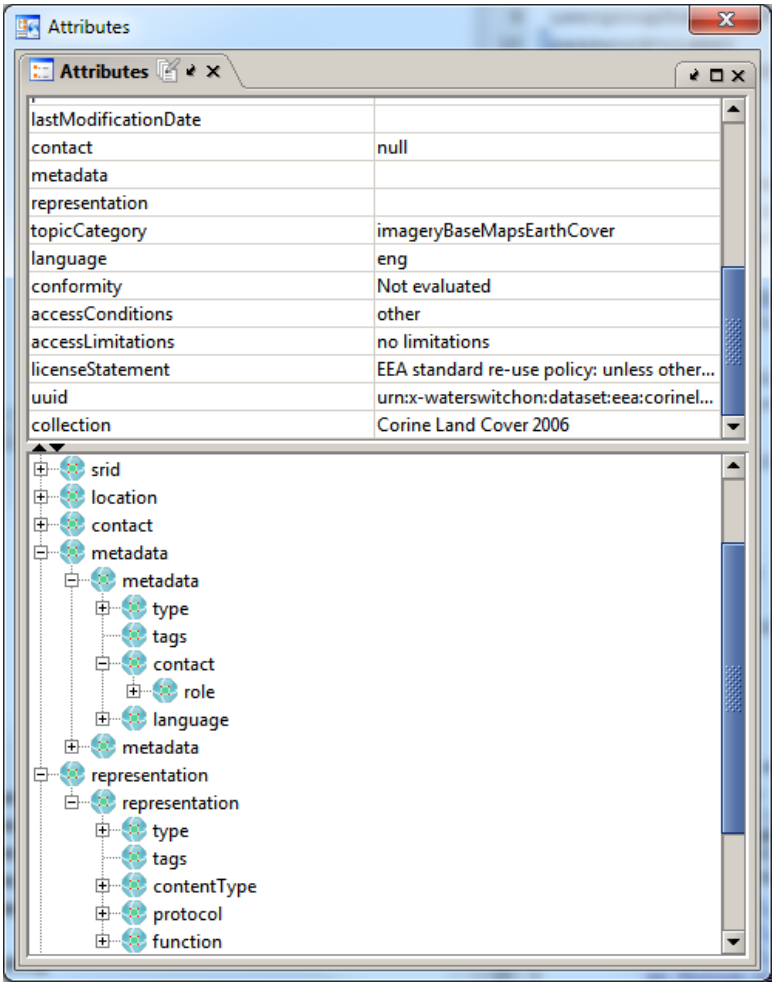

Figure 4: Object Structure in Attribute View

The "raw" value of the content attribute (e.g. XML or JSON documents) should not be shown in the GUI. Instead, it should be formatted as HTML so that it is easily readable by the user.

Requirements on other Components

Currently, there are no Requirements on other Components.

View Experiments and Protocols related to Resources

The user wants to inspect the relationships between resources including information on Experiments and Protocols related to Resources.

SIM and Meta-Data Requirements

Relationship Objects are needed to support this use case. In general an experiment is a relationship that relates input data (source resources, *fromResources[]* attribute) to an experiment result (target resources, *Relationship/toResource* attribute) accompanied by a protocol (*Relationship/metadata* attribute) documenting the experiment.

SIP Back-End Requirements

In contrast to Metadata and Representation objects, Relationship objects are not directly associated with Resource objects but via their *fromResources* and *toResource* attributes. The reason is that

possibly large and complex objects structures would have to be constructed and transmitted, if a resource has many relationships, i.e. have has used in many different experiments. Therefore, explicit "findSourceRealtionship" and "findTargetRealtionship" searches are required if the relationships of a resource is requested by a client.

SIP Expert GUI Requirements

The Resource Renderer should be furthermore extended by a **Relationship Renderer** which adds the following new tabs for displaying the resource relationships:

- **Usage**:
  Populated by a "findTargetRealtionship" search. Shows an Accordion of relationships where the resource is used as input. Each accordion item is a separate Relationship Renderer instance which shows only on link to the respective target resource.
- **Provenance**
  Populated by a "findSourceRealtionship" search. In general, shows only one Relationship Renderer instance (no accordion needed) but multiple links to the source resources.

The Resource Renderer should not implicitly construct the resource relationship when it is activated since for retrieving resource relationships a dedicated search is required. The relationships should be retrieved and constructed asynchronously once the respective tab is activated.

A new **Relationship Renderer** has to be implemented. The Render should show the following information:

- *Relationship/name* as title of the accordion (Usage) or title label (Usage),
- *Relationship/type* as Icon in the title bar or the label,
- *Relationship/description* as label or text area
- If the Relationship Renderer is shown on the **Derivation Tab:** List of *Relationship/fromResources* clickable labels that active the respective Resource Renderer
- If the Relationship Renderer is shown on the **Usage Tab:** *Relationship/toResource* as clickable label that actives the respective Resource Renderer
- Relationship/metadata as Metadata Renderer

Requirements on other Components

Currently, there are no Requirements on other Components.

Show a Preview of the Resource

Once found (either by search or browsing), the user wants to assess the usefulness of resources by looking at a preview of the resource data.

SIM and Meta-Data Requirements

To be able to show a preview of the resource, Representations of type "aggregated data" and / or "preview data" are required. Aggregated data as well as preview data has to be added manually or created automatically.

Aggregated data could be presented in tabular format, so a simple schema for aggregated data that can easily transferred into table model (including column headings and data types) should be provided. Aggregated data should be stored in JSON Format in the content attribute or the "aggregated data" representation.

Preview data could be represented by images, e.g. screenshots of the data (table headings, pre-rendered diagrams, map images, etc.) Preferably, the *contentLocation* of the "preview data" representation points to an image.

To give the Preview Renderer a hint on how to treat and show preview and aggregated data, the attributes *contentType*, *function*, *protocol* and *applicationProfile* of the Representation object are required. The *function* attribute of preview and aggregated representations is either set to "inline" if the data is provided within the *content* attribute or to "download" if it is referenced in the *contentLocation* attribute.

Besides letting the Preview Renderer decide on how to render the preview/aggregated data by evaluating the *contentType* (e.g. image/png or text/javascript) and *protocol* attributes, the *applicationProfile* attribute can be used to explicitly specify a rendering component to be used in the SIP Expert GUI (e.g. de.cismet.cids.navigator.custom.switchon.renderer.preview).

SIP Back-End Requirements

Apart from the (automatic) creation or uploading of preview and aggregated data, possibly with help of custom server actions, there are no requirements on the default entities API.

SIP Expert GUI Requirements

Preview and aggregated data is treated differently than "original data". Representations of type "original data" are shown on the Representation Renderer which gives the user the possibility to access the resource data. The **Preview Renderer** on the other hand enables the user to assess the resource data.

A new **Preview Tab** has to be added to Basic Resource Renderer. The preview tab contains an accordion with Preview Renderer instances for all available preview (and possibly also aggregated data) representations. The Preview Renderer has furthermore to support different types of preview data (images, tabular JSON data, ...) and implement a strategy for deciding how to select the appropriate rendering component. Besides the actual data that is provided either via *content* or via *contentLocation* attributes, the Preview Render should also show the *name* in the title bar of the accordion item. The *description* is automatically generated and hasn't to be shown.

Requirements on other Components

Apart from the need to derive preview and aggregated data from original data, there is possibly a need to store preview data external to the Meta-Data Repository, e.g. on a dedicated WebDAV or FTP Server (Data Repository). This is especially helpful for preview images, which can but shouldn't be stored ASCII-encoded in the TEXT *content* attribute. Instead, *contentLocation* should point to the preview image located on the separate WebDAV / FTP server.

Access Resources Data

Once the user has found (either by search or browsing) and assessed (e.g. by inspecting quality meta-data, preview data, etc.) a resource, he want to access the actual representation(s) of that resource.

SIM and Meta-Data Requirements

At least one Representation object of type "original data" must be associated with the Resource object. This is a fundamental requirement that all Resources stored in the Meta-Data Repository have to fulfil. For the ability to perform custom actions on resources within or from within the SIP Expert GUI, the attributes *contentType*, *function*, *protocol* and *applicationProfile* are needed.

SIP Back-End Requirements

There are no additional SIP Back-End Requirements.

SIP Expert GUI Requirements

A dedicated **Representation Renderer** is needed. An accordion of Representation Renderer instances should be shown in the Representation tab of the Resource Renderer.

In contrast to the Preview Renderer, the Representation Render does not visualise data but facilitates access and usage of data. The information and the respective attributes to be shown by the Representation Renderer are:

- *Name* (title of accordion) and *description*
- If available *spatialResolution, spatialScale* and *temporalResolution*
  Note: Although those attributes are also shown on the Spatial and Temporal Tabs of the Resource Renderer they are needed here to highlight the difference between different resource representations
- Icon indicating the *function* of the link (*contentLocation*) to the representation
- Icon indicating the *contentType* of the representation
- Download Button for direct access to the representation (*contentLocation*)
- If possible, action button that performs an action on the resource representation, e.g. “open in application”, “add to cismap”, ...

The last point in the above list requires the implementation of appropriate **Representation Action Providers** for certain representations. Similar to the Preview Renderer, the appropriate action type can be indentified attributes *contentType*, *function*, *protocol* and *applicationProfile.* Actions should only be activated if the function is either *“download”, “service”* or “inline”.

Several default actions can be implemented for popular combinations of *contentTypes* and *protocols*. For example, an action to add the resource as new data source in cismap should be implemented for the protocols “OGC:WMS” and “OGC:WFS”. The attribute *applicationProfile* can be used to directly specify a custom Action Handler.

Requirements on other Components

Many resources are available in proprietary formats and/or need downloading and unzipping before they can be used. Some, especially the most important ones that are directly used for experiments or for repurposing, could be imported into SWITCH-ON Data Repositories, e.g. an OPeNDAP Server like THREDDS [9]or a WMS/WFS/... like GeoServer[10].

This is most likely a manual process that involves downloading the original data and importing into the respective services. However, some data is available as ESRI SHP or NetCDF files and can be easily imported into those services. Of course, license restrictions and use policies have to be considered.

An example for open data that has been made available in different representations at different locations is HydroSHEDS data: This data is both available in different files formats (ESRI Grid, ESRI BIL, ESRI Shapefile, ...) at the HydroSHEDS website[11] and via WMS and WFS at the U.S. Geological Survey data catalogue[12].

[9] http://www.unidata.ucar.edu/software/thredds/current/tds/

[10] http://geoserver.org/

[11] http://www.hydrosheds.org/

[12] https://www.sciencebase.gov/catalog/item/537f6c31e4b021317a8720e4#

Register new Resource Meta-Data

The user wants to register the meta-data of a new resource in the SIP. This can either be an interesting open data resource available on the internet, repurposed data used as experiment input data or the result data of an experiment.

This use case considers only requirements for putting meta-data in the Meta-Data Repository of the SIP. The upload of open data to a Data Repository is addressed in another use case.

SIM and Meta-Data Requirements

The meta-data of resources that have to be registered in the SIP must be as complete as possible. Besides many mandatory attributes some attributes can be derived automatically or set to appropriate default values.

SIP Back-End Requirements

Validation of meta-data should be performed in the SIP Expert GUI. If needed, validation functionality can be implemented also as part of the new Restful Entities API (SIP Expert Catalogue Management API). Validation and possibly also creation of default attributes can then be performed by custom **Constraints** on database level.

SIP Expert GUI Requirements

A GUI for adding new resource meta-data is needed. This **Meta-Data Wizard** must be able to collect a complete set of meta-data without asking too much from the user. Therefore, some of the attributes should be set to default values or be derived automatically, if possible. For example, the geographic location name could be derived from the spatial extent; *name* and *description* of Representation objects could be set to default values.

Since some parts of the SIM are optional (e.g. meta-data on data quality), the wizard should provide the possibility to skip or hide some of the panels. Furthermore, the possibility to repeat some input panels is needed, e.g. when multiple representations of one resource have to be added. Thereby, the Wizard should allow the selection of different input profiles (e.g. Basic, Advanced and Expert). Thus, the user can decide in advance how much effort he wants to spent to describe open data sets in the SIP.

The Wizard must be implemented in a modular fashion so that wizard panels are also usable stand-alone or can be combined in different order. This is especially needed if only parts of an existing meta-data record have to be edited or complemented by additional meta-data.

The functionality of the wizard can be split into a basic and an advanced part and the development activities can be prioritised accordingly. The basic functionality includes the collection and creation of the following information (basic input profiles):

- One Resource object with mandatory and optional attributes
- One Resource/Contact information object
- Auto-generated Resource/Metadata object of type “basic meta-data”
- Auto-generated Resource/Metadata object of type “original metadata” if meta-data is imported from a file
- 1...n Representation objects of type “original data”

Advanced functionality that can be added later and that is only relevant for advanced and expert input profiles includes:

- Adding additional meta-data like quality and lineage meta-data
- Adding additional preview and aggregated representations
- Creating Relationships between Resources and adding respective meta-data (e.g. links to protocols or uploading protocols to Data Repositories)

The Wizard should furthermore enforce completeness and correctness of the provided input values (e.g. date, ...) by implementing appropriate validation mechanisms.

Requirements on other Components

External services (Data Repositories) have to be provided if aggregated or preview data or protocols should be uploaded by the user but should not be stored directly in the Meta-Data Repository. For specifying the spatial extent of the resource, a WMS server and an appropriate layer (e.g. DEMIS World Map[13]) must be available in cismap.

Edit Resource Meta-Data

The user wants to correct or complete existing resources in the SIP, e.g. by adding a representation of the data or by adding meta-data on data quality.

SIM and Meta-Data Requirements

There are no further requirements.

SIP Back-End Requirements

There are no additional requirements. Updating of objects in the Meta-Data Repository is supported by RMI Entities API. The RESTful Entities API is has to be developed.

SIP Expert GUI Requirements

The Object Renderers introduced in the previous sections should facilitate editing functionality, too. Thus, Resource-, Meta-Data-, Relationship-, Preview- and Representation Renderer should be derived from **Resource-, Meta-Data-, Relationship-, Preview-** and **Representation Editors**.

Furthermore, parts of the Meta-Data Wizard could be re-used when a major editing process has to be performed by the user, e.g. when adding an additional representation of the resource.

Requirements on other Components

There are no additional requirements.

View Resource Data in SIP Expert GUI

Once the user has found (either by search or browsing) and assessed (e.g. by inspecting quality meta-data, preview data, etc.) a resource, he wants to view the actual data of that resource in the SIP Expert GUI.

SIM and Meta-Data Requirements

Data (Representation) of the resource must be in a format that is supported by the SIP Expert GUI. Attributes *contentType*, *protocol*, *function* and *applicationProfile* are required to select an appropriate Data Visualisation Component.

[13] http://www.demis.nl/home/pages/wms/demiswms.htm

SIP Back-End Requirements

No additional requirements are currently known.

SIP Expert GUI Requirements

Data Visualisation Components for different types of data have to be implemented. Geospatial data available on WMS and WMS servers can for example be visualised directly in cismap.

Requirements on other Components

Visualisation Tools for supported data formats and data sources (services) that can be integrated / embedded into the SIP Expert GUI as well as in the Published Catalogue Access GUIs (HTML5 Components) have to be provided.

Publish Resource Data (Simple File Upload)

The user wants to publish new open data which can be the result of (local) repurposing or an experiment by uploading a file to a File Repository.

SIM and Meta-Data Requirements

Provision of a complete and valid set of meta-data for the data to be published has to be ensured. If possible, preview representations as well as relationships to other open data in the SIP have to be created (during the registration of the new data set).

To enable the **Data-Import Wizard** to choose a **File Repository** as well as a location for the uploaded files, appropriate tags have to be assigned to the Resource object to determine the storage location. Thereby, host and some default paths can be chosen from a local configuration file.

Figure 5 shows an example of relevant tag groups that can be used for the configuration of the upload path.

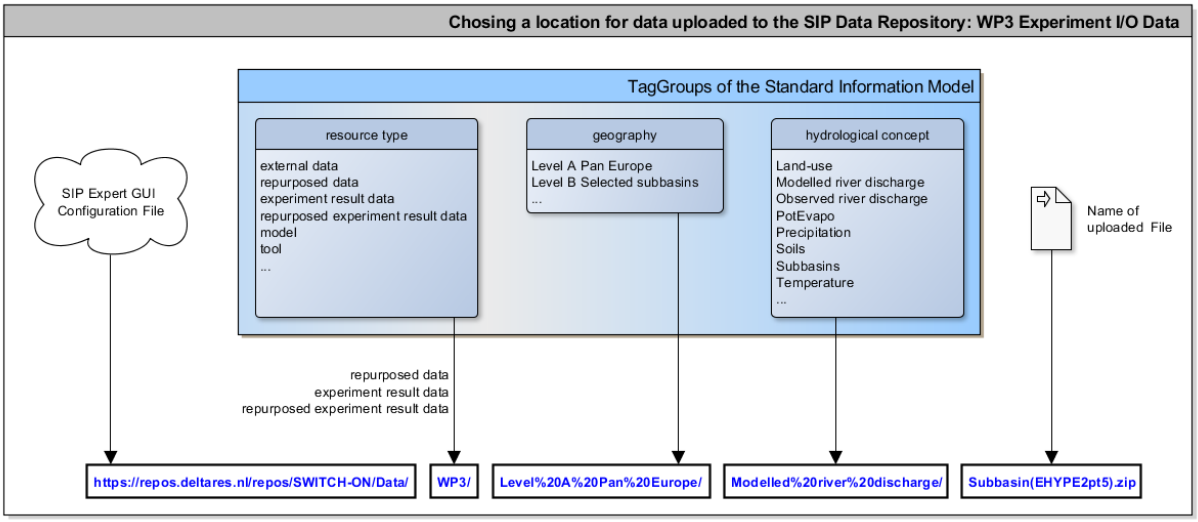

Figure 5: Configuration of File Upload Location

SIP Back-End Requirements

The Meta-Data Repository supports the storage of meta-data but is not concerned with the storage of actual data. However, upload of data and import of data into Data Repositories should be supported.

The default functionality of the new **Data-Import Wizard** includes uploading of data to a WebDAV or FTP Server. The import of the data into a GeoServer or THREDDS Data Server is addressed in another use case.

SIP Expert GUI Requirements

A generic **Data-Import Wizard** has to be developed. This Data Import Wizard can be largely based on the Meta-Data Import Wizard and supports uploading local files to a file-based Data Repositories like WebDAV and FTP. The Data-Import should extend the Meta-Data Import Wizard by new panels for selecting and uploading data.

Requirements on other Components

File based Data Repositories (FTP, WebDAV or SVN Services) for storage of open data and components for importing data into data repositories have to be provided. Furthermore, if open data is stored in SVN, a server-side process that commits to uploaded data into the SVN Repository has to be provided also.

Publish Resource Data (Data Repository Import)

The user wants to publish new open data by importing a file to a dedicated Data Repository (e.g. OPeNDAP or OCC Service). Thus, the data is not only downloadable a single file but can also be inspected and accessed via either the SIP GUI or external data clients.

This use case is considered as an extension of the previous use case: First the file is uploaded to a simple file based Data Repository, the file is imported into another Data Repository like GeoServer or THREDDS.

SIM and Meta-Data Requirements

Since this is the second step after a simple file upload, it is assumed that a Resource Representation with valid *contentLocation* exist. This, and the presence of a respective Tag of the Tag Group "*publish type*" is required to let the back-end determine how to post process the imported file. Further properties of the Representation Object have for example to be used to configure name of layers, etc.

The server-side **Trigger** that publishes the original Resource Representation to an Advanced Data Repository has to ensure that an appropriate meta-data record for the alternate Resource Representation is created. Thereby, it has to be ensured that relevant properties like *function*, *contentType*, etc. are created. For example, if the Trigger publishes a SHP File to a GeoServer as OGC WFS Feature Collection, the *function* property has to be set to "OGC:WFS-1.0.0-http-get-feature", in case of a WMS Layer to "OGC:WMS-1.1.1-http-get-map", etc.

SIP Back-End Requirements

The actual import of the data into a Data Repository can be triggered in the SIP Back-End. The **Trigger** uses the GeoServer or THREDDS API to import the data into the respective service, e.g. creating a new layer from an uploaded SHP file.

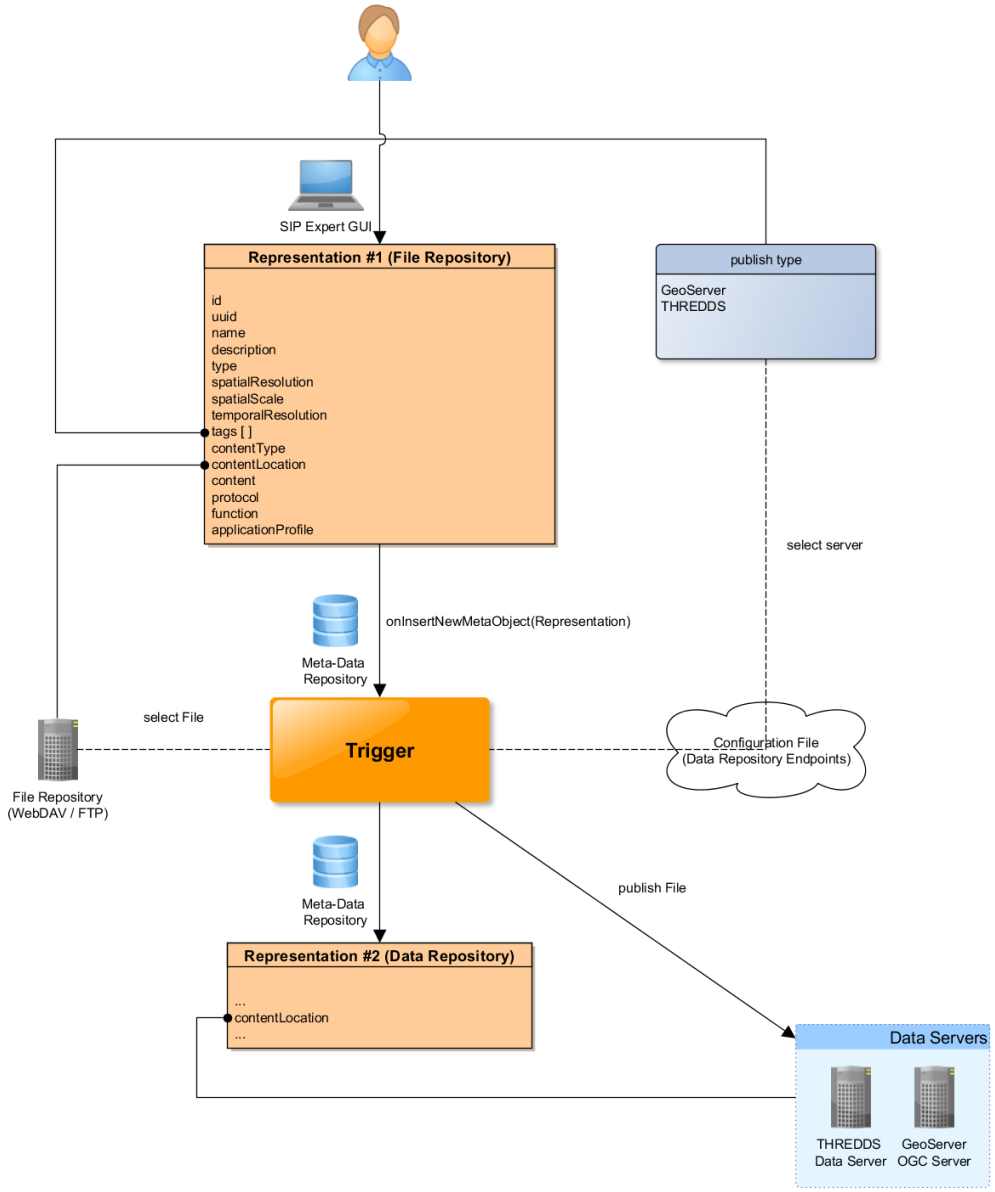

Figure 6: 2-Step Approach for publishing Data to Advanced Data Repositories

As illustrated in Figure 6, the **Trigger** is invoked when a new Representation of Resource that contains a specific Tag of the Tag Group "*publish type*" is stored in the **Meta-Data Repository**. At this point in time, the file referenced by the *contentLocation* of the initial Resource Representation Object has already been uploaded to a simple File Repository by the **Data Import Wizard**. According to the "*publish type*" tag and a local configuration file that contains e.g. endpoints and credentials for the respective **Data Repositories**, the Trigger can perform the advanced import of the file.

Since the Trigger created another alternate representation of the Resource, a second Representation Object has to be stored in the Meta-Data Repository. This second Representation Object has to contain appropriate properties. After successful publication of the alternate representation, the "*publish type*" tag has to be removed from the initial Representation Object to avoid a second publication in case the Representation Object is updated.

SIP Expert GUI Requirements

The Data-Import Wizard has to add a respective "*publish type*" tag to the initial Representation Object. First, the wizard has to determine whether a file uploaded to the simple File Repository is also supported by the advanced Data Repositories. This can be determined by the *contentType*. GeoServer supports for example SHP and GeoTIFF while THREDDS supports at least NetCDF. A mapping of supported *contentTypes* to "publish type" has to be performed. Then, the user has to be asked (e.g. a check box on the next wizard panel) whether he wants to import the uploaded file also to an advanced Data Repository.