FUSION REGISTRY

Web Services User Guide

FUSION REGISTRY
Web Services

This document provides information on the Web Service API for submission and query of Registry Content.

Fusion Registry: 8.4.x
Document Last Updated: May 27th 2015

Contents

1 Overview

The Fusion Registry supports RESTful GET and POST via the URL:

http://[server]:[port]/ws/rest

There is also a RESTful Web Service that supplies auto-increment functionality. This is available via the URL:

http:// [server]:[port]/ws/rest/auto

The auto-increment Web Service will, if necessary, update the version numbers of POSTed structures if there is a collision with existing structures within the Fusion Registry.

1.1.1 RESTful POST

The POST protocol can be used to submit SDMX-ML documents to the Fusion Registry. An SDMX *StructureDocument* can be POSTed in any version of SDMX. Additionally the web service supports SDMX *RegistryInterfaceDocument* submissions, and SDMX-EDI Structure submissions. Documents submitted via POST must have the content type set to either *application/text* or *application/xml*.

1.1.2 Auto Increment API

If posting to the auto-increment web service, the Registry will process the submission to determine if any of the submitted structures already exist in the Registry. If the structures do exist, then the service will determine if any modifications have been made to the submitted structure and, if so, whether the modification is minor or major. In the instance where a modification has been made, the auto increment service will increment the version number of the submitted structure and in turn any referencing structures. The auto increment rules are as follows:

- If a structure is submitted that does not yet exist in the Registry no action is taken.
- If the structure already exists then the version number of the submitted structure is incremented, meaning the existing structure remains unchanged.
- A minor version increment is performed if the structure does differ in the identifiable composite structures. For example, a codelist with exactly the same codes (a code is identifiable as it has a URN).
- A major increment is performed if the submitted structure has an additional identifiable or has had an identifiable removed. For example a codelist may have had a code removed.
- A minor version increment is a .1 increase. For example 1.2 becomes 1.3 (or 1.3.2 becomes 1.4).
- A major version increment is a full integer increase. For example version 1.0 becomes 2.0 (or 1.4 becomes 2.0).
- Any structures that cross reference the existing structure will also have a minor version increment and the references will be updated to reference the latest structure. This rule is recursive, so any structures that reference the references are also updated. This rule will also apply if the referencing document is the same structure submission, as long as the referencing structure in the submission has had the reference modified.

1.1.3 Authentication

If security is enabled, then a POST action must include security credentials in order to authenticate the user. User credentials must be provided using Basic Authentication, as described in the HTTP/1.0 specification.

Note: As basic authentication includes the username and password in the header of the HTTP message it is strongly recommended to POST documents using the HTTPS protocol which will encrypt the entire message.

1.1.4 POST Application Code (Java Sample)

To POST a document from machine code, ensure the content-type HTTP header is set to "application/text" set "Accept-Encoding" to "gzip" if your application is able to handle compressed responses.

The following code is a Java snippet for POSTing a document to the Fusion Registry

```
String xml = "";//some XML to POST
String webService = "http://registry.sdmxcloud.org/ws/rest";
URL url = new URL(webService);
URLConnection urlc = url.openConnection();
urlc.setDoOutput(true);
urlc.setAllowUserInteraction(false);
urlc.addRequestProperty("Content-Type", "application/text");

//Optionally support gzip
urlc.addRequestProperty("Accept-Encoding", "gzip");

//Optionally provide authentication credentials
urlc.addRequestProperty("Authorization", authDetails);

//Optionally provide accept response type
urlc.addRequestProperty("Accept", responseType);

PrintStream ps = new PrintStream(urlc.getOutputStream());
ps.print(xml);
ps.close();
InputStream response = urlc.getInputStream();
```

1.2 RESTful GET

The GET protocol is used to query the Registry contents and conforms to the SDMX Web Services Guidelines specification. The Fusion Registry supports some additional parameters over and above what is defined in the SDMX guidelines, and additionally support is given to query for Registrations via REST, both are documented in sections 1.2.1 and 1.2.2.

The Fusion Registry provides an interactive User Interface to simplify the creation of a REST query, this is available in the HTML UI. As Fusion Registry structures are public there are no restrictions on what can be queried, and there is no requirement to provide security information.

1.2.1 GET Structures

The Fusion Registry supports all parameters as defined in the SDMX Web Services Guidelines. Additional support is given for the following parameters.

Parameter	Allowable Content	Description
partial	true / false	If set to true creates partial Codelists in the response based on constraints information in the Fusion Registry. The pre-requisite is that the query must be for a single constrainable structure (Provision Agreements, Dataflow, or Data Structure) and include references.
version	1.0, 2.0, 2.1, edi	Specifies the response version of the SDMX standard. **Note:** Also supported is the use of content negotiation to define the response version, this is described in the SDMX Web Services Guidelines.

1.2.2 GET Registrations

Registration queries via REST are not specified in the SDMX Web Services Guidelines. The Fusion Registry supports a query via a POST document (RegistryInterface QueryRegistrationRequest). The Fusion Registry also specifies a REST query in order to offer a simpler API.

The entry point is:

http://[server]:[port]/ws/rest/registration

The following parameters are used to identify a registration by specifying what structure(s) the registration references (directly or indirectly).

Parameter	Allowable Content	Description
Referenced structure	dataprovider, datastructure, dataflow, provisionagreement, registration	Defines the structure type that is referenced by the following parameters
agencyID	A string compliant with the SDMX common:NCNameIDType	The agency maintaining the referenced structure.
Id	A string compliant with the SDMX common: IDType	The id of the referenced structure. **NOTE:** For a data provider this is the id of the provider, not the provider scheme. For a single registration this is the id of the registration
Version	A string compliant with the SDMX common:VersionType	The version of the referenced structure. **Note:** This is not relevant if the referenced structure is dataprovider or registration.

Examples:

Query String	Description
ws/rest/registration	All Registrations
ws/rest/registration/dataflow/ESTAT/	All Registrations for ESTAT
ws/rest/registration/dataprovider/WB/DE/	All Registrations for the data provider DE (Germany) for the Agency WB (world Bank)
ws/rest/registration/registration/uudi1236	The Registration with the id uudi1236

1.2.2.1 Parameters used to further describe the desired results

The following query parameters are used to further describe the desired results, once the resource has been identified.

Parameter	Allowable Content	Description
updatedBeforeDate	Common: TimePeriodType, as defined in the SDMXCommon.xsd schema. Can be expressed using xs:gYear (ex: 2008), xs:gYearMonth (ex: 2008-01), xs:date (ex: 2008-01-01), xs:dateTime (ex:2008-01-01T01:18:59+01:00) and the SDMX period type (ex: 2008-Q1). In case the : or + characters are used, the parameter must be percent-encoded by the client11.	Returns any registrations which have been updated before the given date.
updatedAfterDate	Same as above	Returns any registrations which have been updated after the given date.

Examples:

Query String	Description
ws/rest/registration?updatedAfterDate=2008	Returns all registrations updated since 2008
ws/rest/registration/dataflow/ESTAT/?updatedBeforeDate=2008-01-01	All Registrations for ESTAT updated after 2008-01-01

1.2.3 GET Application Code (Java Sample)

The following code is a Java snippet for GETing a document from the Fusion Registry

```
String restQuery =
"http://registry.sdmxcloud.org/ws/rest/datastructure/all/all?references=descendant
s";
URL url = new URL(restQuery);
URLConnection urlc = url.openConnection();
urlc.setDoOutput(true);
urlc.setAllowUserInteraction(false);

//Optionally support gzip
urlc.addRequestProperty("Accept-Encoding", "gzip");

//Optionally provide accept response type
urlc.addRequestProperty("Accept", responseType);

InputStream response = urlc.getInputStream();
```

1.3 Reference Resolution

The REST query supports a query for one or more structures by defining the structure type, agency id, id, and version of the structure(s) to be returned. Once the Fusion Registry has determined which structure(s) match this criteria it will then, if requested, resolve any referenced structures. There are six types of reference resolution requests which are described below:

1. None (do not resolve any references)
2. Children
3. Descendants
4. Parents
5. Parents and Siblings
6. All

1.3.1 References Children

"Children" refers to any structures which are directly referenced from the target structure. For example a Dataflow must reference a Data Structure. The Data Structure is considered a child of the dataflow, in this context. A structure may have more than one child; for example a Data Structure may reference may Codelists and many Concepts[1]. The diagram below shows a tree like structure which represents a hierarchy of structures. This diagram illustrates what would be returned if a dataflow was queried, with references=children.

[1] Note that with a reference such as a concept, the entire concept scheme will be returned

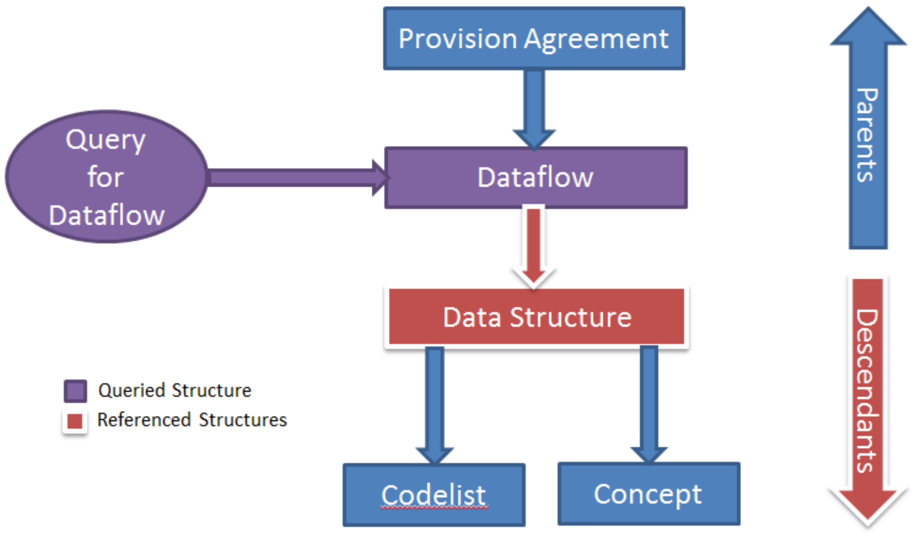

Figure 1 Showing a REST query where references=children

1.3.2 References Descendants

"Descendants" refers to the structures which are referenced from the target structure, including all the children of those structures, and their children etc. Setting the references parameter to descendants will bring back the 'whole tree' in the hierarchy starting from the target structure.

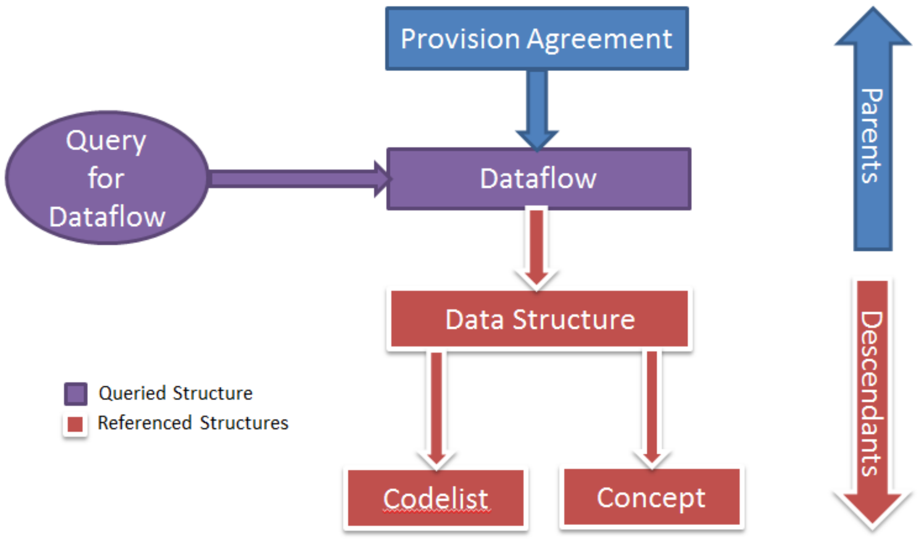

Figure 2 Showing a REST query where references=descendants

1.3.3 References Parents

"Parents" refers to the structures that directly reference the target structure. This does not include the parents of those parents. Unlike traversing down the hierarchy which can traverse down to all the descendants, the references parameter only provides support for traversing 1 level up the structure hierarchy.

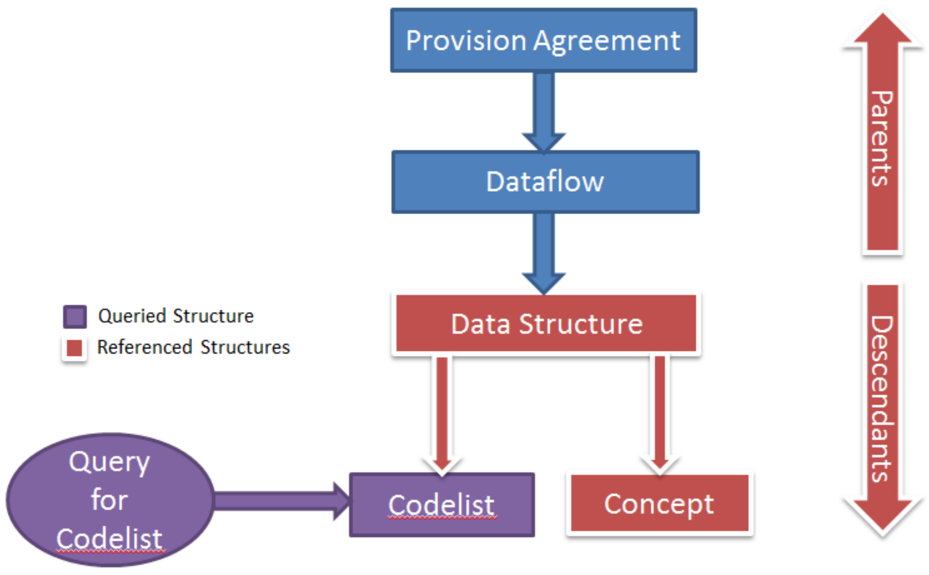

Figure 3 Showing a REST query where references is set to parents

1.3.4 References Parents and Siblings

"Parents and Siblings" is quite a complex type of reference resolution. It brings back the immediate parents of the referenced structure, and for each parents, their immediate children are returned. These child structure of the parents, are deemed as the siblings of the target structure. This can be seen in the diagram below, where the queried structure is a Codelist. The parent of the codelist, in this scenario, is the Data Structure. The children of the data structure, are the codelist and concept.

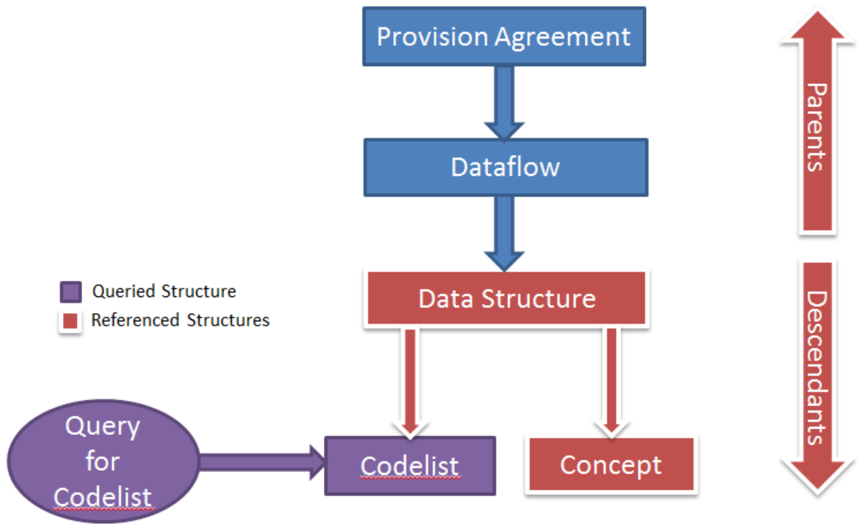

Figure 4 Showing a REST query where references is set to Parents and Siblings

1.3.5 References Specific

It is possible to choose a specific structure type to include in the references. This allows the user to specify which structure type should be returned in the reference resolution. The structure type must directly reference, or be referenced by the target structure, either as a parent, or a child. The allowable arguments are defined in section 4.3.1 of the Web Services Guidelines.

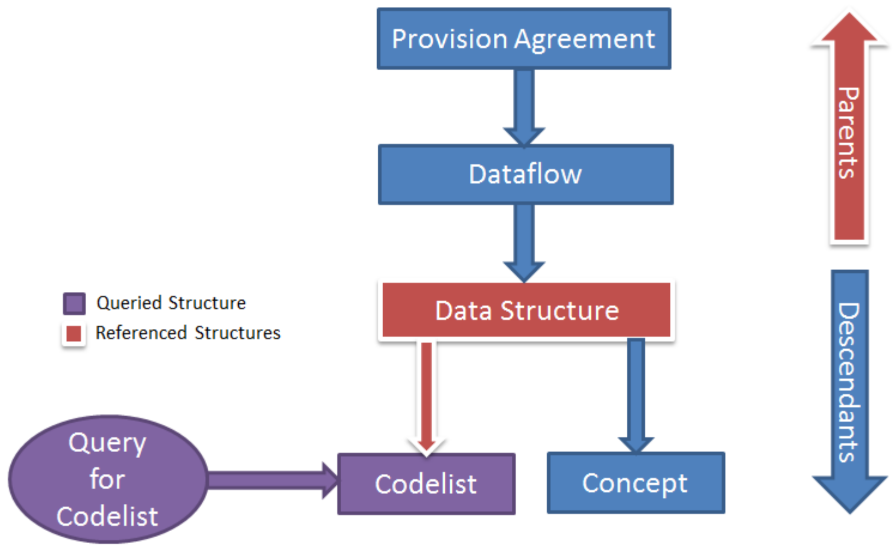

Figure 5 Showing a REST query where references is set to Codelist

1.3.6 References All

If "references" is set to ALL, then the parents and children of the target structure will be returned.

2 SOAP

The Fusion Registry supports SOAP queries via the following URL.

http://[server]:[port]/FusionRegistry/ws/soap

The Web Service Definition Language (WSDL), which describes the functionality offered by the service, is available at the following URL.

http://[server]:[port]/FusionRegistry/ws/soap/sdmxSoap.wsdl

The SOAP interface as defined by the WSDL is much richer than the REST interface with the WSDL specifying more operations and more functionality than provided by the REST service. However it is important to state that the Fusion Registry currently only supports a sub-set of the available functionality declared by the WSDL. The most useful functionality has been supported allowing the most-common SOAP queries to be performed.

As a rule of thumb, the SOAP interface supports the equivalent level of functionality that the REST interface provides.

2.1.1.1 SOAP Functionality

The following methods are supported and will return the appropriate structure:

- getCategorisation
- getCategoryScheme
- getCodelist

- getConceptScheme
- getConstraint
- getDataflow
- getDataSchema
- getHierarchicalCodelist
- getMetadataflow
- getMetadataStructure
- getOrganisationScheme
- getProcess
- getProvisionAgreement
- getReportingTaxonomy
- getStructures
- getStructureSet

2.1.1.2 Detail Level of Returned Structures

An important area where the implementation of the SOAP interface is incomplete, is in the detail level of the structures being returned by the SOAP query. The SOAP WSDL provides the ability to specify the detail level on the structure being directly queried for, and also the SOAP WSDL provides the ability to specify the detail level on any referenced structures. This goes beyond the level of detail that the REST Web Service provides.

For each of these detail levels there are 4 possible options: Full, Stub, CompleteStub or "no value specified". To provide a simple implementation, the Fusion Registry SOAP interface does not really support the "CompleteStub" parameter but will instead supply a Stub when it is appropriate to do so. The table below shows what will be returned through the combination of the "References" Detail and the "Return Details" detail. Whatever the combination of detail levels, there are only 4 possible outcomes:

Full	– All available information for all artefacts will be returned
All Stubs	– All artefacts will be returned as stubs
Reference Stubs	– Only referenced artefacts will be returned as stubs
Error: Unsupported	– An error will be thrown by the SOAP interface.

		"References" Detail			
		Full	**Stub**	**CompleteStub**	**Not Specified**
"Return Details" Detail	Full	Full	Reference Stubs	Reference Stubs	Full
	Stub	Error: Unsupported	All Stubs	All Stubs	All Stubs
	CompleteStub	Error: Unsupported	All Stubs	All Stubs	All Stubs
	Not Specified	Full	Reference Stubs	Reference Stubs	Full

So, through specifying settings at the References and Return Details level, the SOAP interface can return structures in full or as stubs.

2.1.1.3 Reference Resolution

A further setting within the “References” element allows further specification of the scope of which references are returned. The available values are:

1. None (do not resolve any references)
2. Children
3. Descendants
4. Parents
5. ParentsAndSiblings
6. All
7. SpecificObjects (e.g. Codelist)

These options are the same as that of the REST interface.