SDMX REGISTRY SPECIFICATION: LOGICAL INTERFACES

(VERSION 2.0)

November 2005

Contents

151

1 INTRODUCTION

The business vision documents for SDMX envisage the promotion of a "data sharing" model to facilitate low-cost, high-quality statistical data and metadata exchange. Data sharing reduces the reporting burden of organisations by allowing them to publish data once, and let their counterparties "pull" data and related metadata as required. The scenario is based on:

- the availability of an abstract information model capable of supporting any time-series and cross-sectional data, structural metadata and reference metadata (SDMX-IM)
- standardised XML schemas derived from the model (SDMX-ML)
- the use of web-services technology (XML, XSD, WSDL, ebXML Registries)

Such an architecture needs to be well organised, and the SDMX Registry/Repository (SDMX-RR) is tasked with providing structure and organisation for most of the SDMX components required to support the data-sharing vision.

Standard formats for the exchange of aggregated statistical data and metadata as prescribed in SDMX v1.0 (ISO/TS 17369:2005 SDMX) are envisaged to bring benefits to the statistical community because data reporting and dissemination processes can be more efficient.

As organisations migrate to SDMX enabled systems, many XML (and conventional) artefacts will be produced [e.g. Key Family, Metadata Structures, Code List and Concept definitions (often called structural metadata), XML schemas generated from key families, XSLT style-sheets for transformation and display of data and metadata, terminology references, etc.]. The data sharing model is based on interoperability, and the discovery and sharing of these artefacts between parties in a controlled and organized way.

With these fundamental standards in place, a set of architectural standards are needed to address some of the processes involved in statistical data and metadata exchange, with an emphasis on promoting the data-sharing vision. Architectural standards address the 'how' rather than the 'what', and are aimed at enabling existing SDMX standards to achieve their mission. The architectural standards address registry services which initially comprise: structural metadata repository, data provisioning repository, data and metadata registration and query. The registry services outlined in this specification are designed to help the SDMX community manage the proliferation of SDMX assets and to support data sharing for reporting and dissemination.

2 SCOPE AND NORMATIVE STATUS

The scope of this document is to specify the logical interfaces for the SDMX registry in terms of the functions required and the data that may be present in the function call. This interface specification is syntax independent.

In this document, functions and behaviours of the registry interfaces are described in three ways:

- In text
- With tables
- With UML diagrams excerpted from the SDMX Information Model

For those interested in seeing a syntax-bound implementation, the registry interfaces are implemented as part of the SDMX-ML schemas.

Whist the introductory section contains some information on the role of the registry, it is assumed that the reader is familiar with the uses of a registry in providing shared metadata across a community of counterparties. The SDMX Implementers' Guide contains more details on the specific roles the registry plays.

Note that section 5.2 contains normative rules regarding the identification of registry objects. Further, the minimum standard for access to the registry is via a REST interface (HTTP or HTTPS), as described in the appropriate sections. The notification mechanism must support e-mail and HTTP/HTTPS protocols as described. Normative registry interfaces are specified in the SDMX-ML implementation. All other sections of this document are explanatory.

3 OBJECTIVES OF THE SDMX REGISTRY/ REPOSITORY

The objective of the SDMX Registry/Repository is, in broad terms, to allow organisations to publish statistical data and metadata in known formats such that interested third parties can discover these data and interpret them accurately and correctly and within the shortest possible timescale. The mechanism for doing this is set out below in high-level terms:

Setting up structural metadata and the exchange context (referred to as "data provisioning") involves the following steps for maintenance agencies:

- Agreeing and creating a specification of the structure of the data (called a key family) which defines the dimensions, measures and attributes of a dataset and their valid value set
- Defining a subset or view of a key family which allows some restriction of content called a "dataflow definition"
- Agreeing and creating a specification of the structure of metadata (metadata structure definition) which defines the attributes and presentational arrangement of a metadataset and their valid values and content
- Defining a subset or view of a metadata structure definition which allows some restriction of content called a "metadataflow definition"

- Defining which subject matter domains are related to the dataflow and metadataflow definitions to enable browsing
- Defining one or more lists of data providers (which includes metadata providers)
- Defining which data providers have agreed to publish a given dataflow and/or metadataflow definition - this is called a provision agreement

Publishing the data and metadata involves the following steps for a data provider:

- Making the metadata and data available in SDMX-ML conformant data files or databases (which respond to an SDMX-ML query with SDMX-ML data) - the data and metadata files or databases must be web-accessible, and must conform to an agreed dataflow or metadataflow definition (key family or metadata structure definition subset)
- Registering the published metadata and data files or databases with one or more SDMX Registries

Notifying interested parties of newly published or re-published data, metadata or changes in structural metadata:

- The Registry can optionally support a subscription-based notification service which sends an email announcing all published data that meets the criteria contained in the subscription request

Discovering published data and metadata involves the following steps:

- Optionally browsing a subject matter domain category scheme to find dataflow definitions (and hence key families) and metadataflows which structure the type of data and/or metadata being sought
- Build a query, in terms of the selected key family or metadata structure definition, which specifies what data are required
- Submit the query to an SDMX Registry which will return a list of (URLs of) data and metadata files and databases which satisfy the query
- Processing the query result set and retrieving data and/or metadata from the supplied URLs

4 SDMX REGISTRY/REPOSITORY ARCHITECTURE

4.1 Architectural Schematic

The architecture of the SDMX Registry/Repository is derived from the objectives stated above. It is a layered architecture that is founded by a structural metadata repository which supports a provisioning metadata repository which supports the registry services.

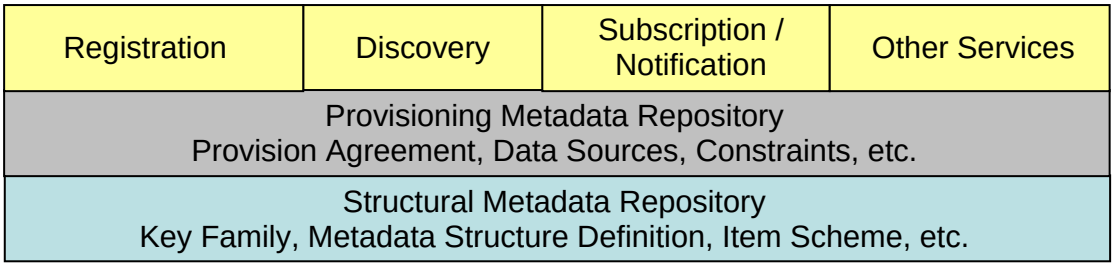


Figure 1: Schematic Architecture of SDMX Registry/Repository

4.2 Structural Metadata Repository

The basic layer is that of a structural metadata service which supports the lifecycle of SDMX structural metadata artefacts such as Maintenance Agencies, Key Families, Reference Metadata Structure Definitions etc. This layer must allow structural definitions to be created, modified, and removed in a controlled fashion. It must also allow the structural metadata artefacts to be queried and retrieved either in part or as a whole. In order for the architecture to be scalable, the finest-grained piece of structural metadata that can be processed by the SDMX-RR is a MaintainableArtefact (see next section on the SDMX Information Model).

4.3 Provisioning Metadata Repository

Supported by the structural metadata service (effectively a repository) the provisioning service allows the definition of the data and metadata supply chain by the introduction of provision agreements. This is analogous to a “service level agreement” whereby a data provider commits to publishing a dataflow or metadataflow according to an agreed schedule. This repository also supports the definition of various types of data-store which model SDMX-conformant databases or files and can be specified for a data provider, or specifically for a data or metadata flow. In addition, the provisioning repository supports the definition of constraints which can define a valid sub-set of a key family or metadata structure definition. Constraints can be used for validation and also for attaching reference metadata to specific points within a dataflow or dataset. This layer must allow provisioning definitions to be created, modified, and removed in a controlled fashion. It must also allow the provisioning metadata artefacts to be queried and retrieved either in part or as a whole.

4.4 Registry Services

4.4.1 Data and Metadata Registration Service

The registration service allows SDMX conformant XML files and web-accessible databases containing published data and reference metadata to be registered in the SDMX Registry. The registration process involves validating the content of the data-sets or metadata-sets, extracting a concise representation of the contents in terms of concept values and storing this as a record in the registry to enable discovery of the original data-set or metadata-set.

The registration of data-sets and metadata-sets (that is, indexing of values found in the data and metadata sets) is supported by a service which depends upon the provisioning metadata repository for the following information:

- Is the data provider allowed to register the data-set or metadata-set?
- Does the content of the data-set or metadata-set meet the validation constraints?
- Has the data-set or metadata-set been published in a timely fashion?
- Has a database already been set up as a data-source for this data provider and data flow?

This service also depends upon the structural metadata repository for the following information:

- What key family or metadata structure definition is used by the registered data?
- What are the components (dimensions, attributes, measures, identifier components etc.) of the key family or metadata structure definition?
- What are the valid representations of the concepts to which these components correspond?

4.4.2 Data and Metadata Discovery

The discovery service allows:

- dataflows (or metadataflows) to be found within a statistical subject-matter category scheme
- data-sets and metadata-sets to be located based on the dataflow (or metadataflow), key family (or metadata structure definition), or provider agreement and a specification of content via concept value definitions.

The discovery of data-sets (or metadata-sets) is supported by a service which depends upon the registration service for the following information:

- Has an SDMX conformant XML file or database been registered for the dataflow (or metadataflow), key family (or metadata structure definition) or provision agreement in question?
- Does the content of the registered data-set (or metadata-set) meet the discovery query constraints (including both concept values and reference periods)?

The discovery service also depends upon the provisioning metadata repository for the following information:

- Has an SDMX-conformant database been registered for the data provider or provision agreement in question?
- Does the content of a selected SDMX-conformant database meet the discovery query constraints (including both concept values and reference periods)?

This service also depends upon the structural metadata repository for the following information:

- What dataflow (or metadataflow) is related to a selected statistical subject-matter category?
- What key family or metadata structure definition is used by the dataflow (or metadataflow)?
- What are the components (dimensions, attributes, measures, identifier components etc.) of the key family or metadata structure definition?
- What are the valid representations of the concepts to which these components correspond?

4.4.3 Subscription and Notification

The data sharing paradigm relies upon the consumers of data and metadata being able to pull information from data providers' dissemination systems. For this to work efficiently, a data consumer needs to know when to pull data, i.e. when something has changed in the registry (e.g. a dataset has been updated and re-registered). Additionally, many SDMX stakeholders will also want to know if a new key family, code-list or metadata structure definition has been added. The subscription and notification service comprises two parts: subscription management, and notification.

Subscription management involves an authenticated user submitting a subscription request which contains:

- A query or constraint expression which defines the events for which the user is interested(e.g. new data for a specific dataflow, or for a domain category, or changes to a key family).
- A list of URIs or end-points to which an XML notification message can be sent. Supported end-point types will be email (mailto:) and HTTP POST (a normal http:// address).

Subscription management must also provide a way to list subscriptions and to delete subscriptions.

Notification requires the structural metadata repository, the provisioning metadata repository and the data and metadata registration service to monitor any event which is of interest to a consumer (the object of a subscription request query) and to issue an SDMX-ML notification document to the end-points specified in the relevant subscriptions.

5 SDMX INFORMATION MODEL (SDMX-IM)

The data requirements presented here are consistent with, and have been derived from, the SDMX-IM version 2.0. In some cases, to aid clarity and to avoid the necessity of the reader to have concurrent access to the SDMX-IM, parts of the model are reproduced here.

In some cases the model presented in this document is a logical model of the data requirements of the interface function and as such is not a part of the SDMX-IM but is drawn here to aid understanding. These specialised class diagrams are annotated as such in this document.

5.1 Identification, Versioning, and Maintenance

All major classes of the SDMX Information model inherit from one of:

- IdentifiableArtefact - gives an object the ability to be uniquely identified (see following section on identification), to carry a multi-lingual name and description, to have a user-defined URI, and to carry multi-lingual annotations.
- VersionableArtefact - has all of the above features plus a version number and a validity period.
- MaintainableArtefact - has all of the above features plus an association to the maintenance agency of the object.

5.1.1 Identification, Versioning and Maintenance Model

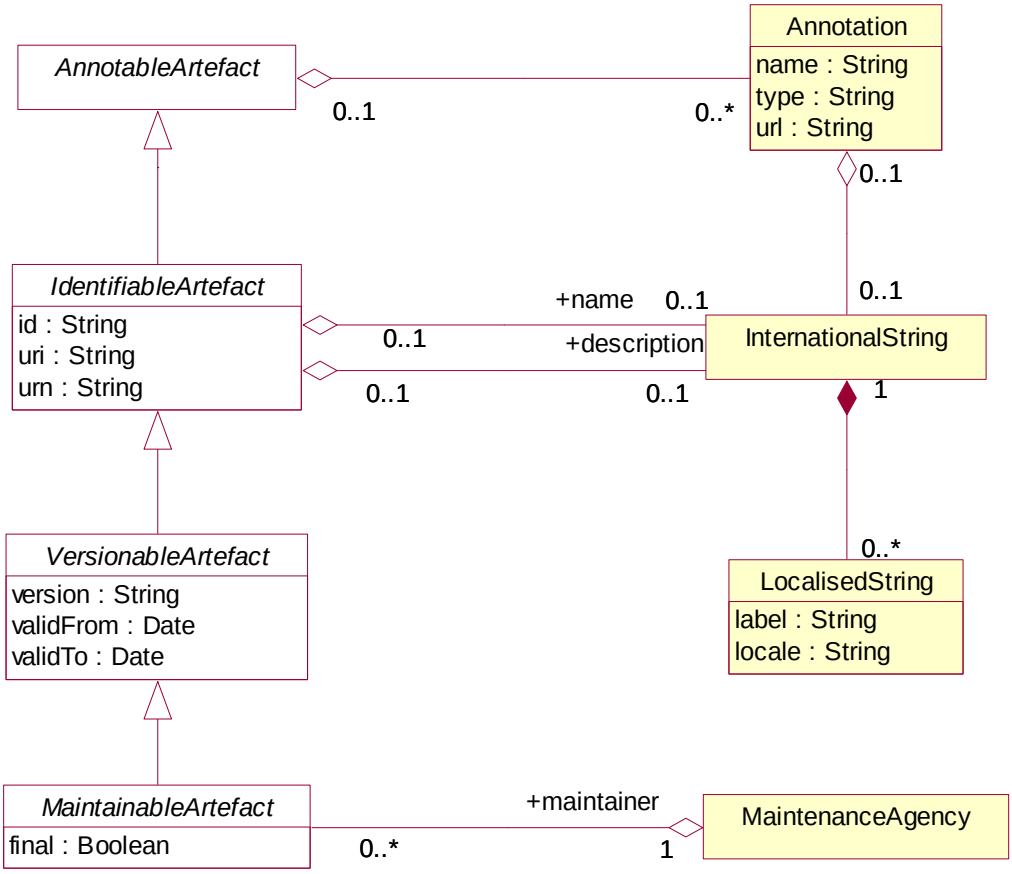


Figure 2: Class diagram of fundamental artefacts in the SDMX-IM

The table below shows the identification and related data attributes to be stored in a registry for objects that are one of:

- Annotable
- Identifiable
- Versionable
- Maintainable

Object Type	Data Attributes	Status	Data type	Notes
Annotable	name	M	string	
	type	C	string	
	uri	C	string	
	annotation text	M		This can have language-specific variants.
Identifiable	id	M	string	
	uri	C	string	

Object Type	Data Attributes	Status	Data type	Notes
	urn	M	string	Although the urn is computable and therefore may not be stored physically, it is logically mandatory as applications must return the urn for each object, and must be able to service a query on an object referenced solely by its urn.
	name	M	string	This can have language-specific variants.
	description	C	string	This can have language-specific variants.
Versionable	All data as for Identifiable plus			
	version	M	string	This is the version number
	dateFrom	C	Date/time	
	dateTo	C	Date/time	
Maintainable	All data as for Versionable plus			
	final		boolean	Value of "true" indicates that this is a final specification and it cannot be changed unless as a new version.
	Maintenance Agency Id	M	string	The object must be linked to a maintenance agency

426 **Table 1: Common Attributes of Object Types**

427 5.2 Unique identification of SDMX objects

428 5.2.1 Universal Resource Name (URN)

429 To provide interoperability between SDMX Registry/Repositories in a distributed
430 network environment, it is important to have a scheme for uniquely identifying (and
431 thus accessing) all first-class (Identifiable) SDMX-IM objects. Most of these unique
432 identifiers are composite (containing maintenance agency, or parent object
433 identifiers), and to build references in XML would be very cumbersome (a reference
434 to a maintenance agency has one component, but to a dimension has five). To
435 simplify this, all SDMX objects also have a globally unique identifier called a universal
436 resource name (URN) which is generated from the actual key components by SDMX-
437 RR APIs. Note that this technique is used in ISO 15000 (ebXML registry
438 specification), thus the adoption of a similar technique for the SDMX-RR makes it
439 easier to map SDMX artifacts into an ISO 15000 compliant registry.
440
441 The structure of the URN is as follows:
442
443 `Prefix.SDMX-IM package name.classname=agency id:object id`

Or, where the object is in a maintained list of objects (called Item Scheme in the SDMX-IM)

```
Prefix.SDMX-IM package name.classname=agency id:Item Scheme
id.object id
```

Or, where the object is in a maintained list of objects that can be hierarchic and where the object id may not be unique in itself but only within the context of the hierarchy

```
Prefix.SDMX-IM package name.classname=agency id:Item Scheme
id.object id.object id
```

(Maintenance) agency identifiers always precede major SDMX artefact identifiers and act as a namespace or definition of ownership. Agency identifiers are always separated from the maintainable artefact identifier by a colon ':'. All other identifiers are separated by a period(.)

Where:

Prefix: `urn:sdmx:`

SDMX-IM package name: `sdmx.infomodel.package.`

The packages are:

```
base
codelist
conceptscheme
keyfamily
categoryscheme
registry
metadatastructure
transformation
process
mapping
```

```
Thus:    urn:sdmx:org.sdmx.infomodel.registry.ProvisionAgreement
```

Examples

The key family EXT_DEBT maintained by the BIS would have the URN:

```
urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=BIS:EXT_DEBT
```

The URN for a code for Argentina could be:

```
urn:sdmx:org.sdmx.infomodel.codelist.Code=ISO:CL_3166A2.AR
```

The URN for a category in a hierarchic category scheme could be:

```
urn:sdmx:org.sdmx.infomodel.categoryscheme.Category=IMF:SDDS_C
ATEGORIES.REAL_SECTOR.BOP
```

5.2.2 Identification components of SDMX objects

The table below describes the identification components for all SDMX object types that have identification. Note the the actual attributes are all Id, but have been prefixed by their class name or multiple class names to show navigation, e.g. conceptSchemeAgencyId is really the Id attribute of the Agency class that is associated to the ConceptScheme. * indicates that the object is maintainable.

SDMX Class	Key attribute(s)	Example
Agency	Note. On the model this is an organisation role and so is theoretically identified the same way as data provider. However, all agencies are in the one scheme, maintained by a fixed and known agency, which does not need to be identified. agencyId	IMF
Concept Scheme*	conceptSchemeAgencyId:conceptSchemeId	SDMX:CORE
Concept	conceptSchemeAgencyId: conceptSchemeId. conceptId	SDMX:CORE.FREQ
CodeList*	codeListAgencyId: codeListId	SDMX:CL_FREQ
Code	codeListAgencyId: codelistId.codeId.codeId	SDMX:CL_FREQ.Q For hierarchical list it is IMF:CL_HIERARCHY.level_1_code.level_2_code
KeyFamily*	keyFamilyAgencyId: keyFamilyId	TFFS:EXT_DEBT
KeyDescriptor Measure Descriptor Attribute Descriptor	keyFamilyAgencyId: keyFamilyId. componentListId	TFFS:EXT_DEBT.KeyDescriptor TFFS:EXT_DEBT.MeasureDescriptor TFFS:EXT_DEBT.AttributeDescriptor
GroupKey Descriptor	keyFamilyAgencyId: keyFamilyId. groupKeyDescriptorId	TFFS:EXT_DEBT.SIBLING
Dimension Measure Attribute	keyFamilyAgencyId: keyFamilyId. ConceptschemeAgencyId: conceptSchemeId. conceptId	TFFS:EXT_DEBT.SDMX:CORE.FREQ
Category Scheme*	categorySchemeAgencyId: categorySchemeId	IMF:SDDS
Reporting Taxonomy*	reportingTaxonomyAgencyId: reportingTaxonomyId	BIS:BASELII

SDMX Class	Key attribute(s)	Example
Category	categorySchemeAgencyId, categorySchemeId. categoryId.categoryId	IMF:SDDS: level_1_category.level_2_category …
Organisation Scheme*	organisationSchemeAgencyId, organisationSchemeId,	SDMX:DATA_PROVIDER
Data Provider	organisationSchemeAgencyId: organsiationSchemeId. dataProviderId	SDMX:DATA_PROVIDER.1F
Metadata Structure Definition*	MSDAgencyId:MSDId	IMF:SDDS_MSD
FullTarget Identifier	MSDAgencyId: MSDId.fullTargetId	IMF:SDDS_MSD.FULL_IDENTIFIER
PartialTarget Identifier	MSDAgencyId: MSDId.partialTargetId	IMF:SDDS_MSD.CATEGORY
Metadata Attribute	MSDAgencyId: MSDId. ConceptschemeAgencyId: conceptSchemeId. conceptId.conceptId	IMF:SDDS_MSD.SDMX:CORE.COMPILATION
Identifiable Object Type	objectTypeSchemeAgency id: objectTypeSchemeId. objectType	SDMX:IM_OBJECT_TYPES.DATA_PROVIDER
Dataflow*	dataflowAgencyId: dataflowId	TFFS:CRED_EXT_DEBT
Provision Agreement*	organisationSchemeAgencyId: organisationSchemeId. dataProviderId. dataflowAgencyId: dataflowId	SDMX:DATA_PROVIDER.1F.TFFS:CRED_EXT_DEBT
Content Constraint	ConstrainableId.ContentConstraintId	TFFS:CREDITOR_DATA.CONTENT_CONSTRAINT
Attachment Constraint	ConstrainableId. AttachmentConstraintId	TFFS:CREDITOR_DATA.ATTACHMENT_CONSTRAINT_ONE
Metadataflow*	metadataflowAgencyId: metadataflowId	IMF:SDDS_CONTACTS
Dataset	organisationSchemeAgencyId: organisationSchemeId. dataProviderId.datasetId	Note that the dataset id is a unique id for the data provider that identifies this data set SDMX:DATA_PROVIDER.1F.DATA123
XSDataset	organisationSchemeAgencyId: organsiationSchemeId. dataProviderId. xsdatasetId	SDMX:DATA_PROVIDER.1F.XSDATA789
Metadata Set	organisationSchemeAgencyId: organsiationSchemeId.	SDMX:DATA_PROVIDER.1F.METADATA456

SDMX Class	Key attribute(s)	Example
	dataProviderId. metadadataSetId	
Component	StructureAgencyId: StructureId. ConceptSchemeAgencyId: ConceptSchemeId. ConceptId	This is an abstract class, so does not have a concrete example. Refer to Dimension, Attribute or Measure which are the concrete manifestations.
Hierarchical Codelist*	hierarchicalCodeListAgencyId: hierarchicalCodeListId	SDMX:CL_REF_AREA
Hierarchy	hierarchicalCodeListAgencyId: hierarchicalCodeListId: hierarchyId	SDMX:CL_REF_AREA.MX For hierarchical list it is IMF:CL_REF_AREA.level_1_code.level_2_code
StructureSet*	StructureSetAgencyId: StructureSetId	SDMX:BOP_STRUCTURES
StructureMap	StructureSetAgencyId: StructureSetId. StructureMapId	SDMX:BOP_STRUCTURES.TABLE1_TABLE2
Component Map	StructureSetAgencyId: StructureSetId. StructureMapId. ComponentMapId	SDMX:BOP_STRUCTURES.TABLE1_TABLE2. REFAREA_REPCOUNTRY
CodeListMap	StructureSetAgencyId: StructureSetId. ComponentMapId. CodeListMapId	SDMX:BOP_STRUCTURES.TABLE1_TABLE2.REFAREA_REPCOUNTRY.CLREFAREA_CLREPCOUNTRY
CodeMap	StructureSetAgencyId: StructureSetId. ComponentMapId. CodeListMapId. CodeMapId	SDMX:BOP_STRUCTURES.TABLE1_TABLE2.REFAREA_REPCOUNTRY.CLREFAREA_CLREPCOUNTRY.DE_GER
Category SchemeMap	StructureSetAgencyId: StructureSetId. CategorySchemeMapId	SDMX:BOP_STRUCTURES.SDMX_EUROSTAT
CategoryMap	StructureSetAgencyId: StructureSetId. CategorySchemeMapId. CategoryMapId	SDMX:BOP_STRUCTURES.SDMX_EUROSTAT.2.3.1_3.4.2.1
Organisation SchemeMap	StructureSetAgencyId: StructureSetId. OrganisationSchemeMapId	SDMX:BOP_STRUCTURES.DATA_PROVIDER_MAP
Organisation RoleMap	StructureSetAgencyId: StructureSetId. OrganisationSchemeMapId. OrganisationRoleMapId	SDMX:BOP_STRUCTURES. DATA_PROVIDER_MAP.IMF_1C0
Concept SchemeMap	StructureSetAgencyId: StructureSetId. ConceptSchemeMapId	SDMX:BOP_STRUCTURES.SDMX_OECD

SDMX Class	Key attribute(s)	Example
ConceptMap	StructureSetAgencyId: StructureSetId. ConceptSchemeMapId. ConceptMapId	SDMX:BOP_STRUCTURES.SDMX_OECD.COVERAGE_AVAILABILITY
Process*	ProcessAgencyId: ProcessId	BIS:PROCESS1
ProcessStep	ProcessAgencyId: ProcessId. ProcessStepId	BIS:PROCESS1.STEP1
Transition	ProcessAgencyId: ProcessId. TransitionId	BIS:PROCESS1.STEP1_STEP3
Transformation Scheme*	TransformationSchemeAgencyId: TransformationSchemeId	BIS:TRANSFORMS
Expression Node	TransformationSchemeAgencyId: TransformationSchemeId. ExpressionNodeId	BIS:TRANSFORMS.EXPRESSION1

Table 2: Table of identification components for SDMX Identifiable Artefacts

6 STRUCTURAL DEFINITION METADATA

6.1 Introduction

The SDMX Registry must have the ability to support maintenance agencies in their role of defining and disseminating structural metadata artefacts. These include key families, code lists, concepts etc. and are fully explained in the SDMX Implementer's Guide. An authenticated maintenance agency may submit valid structural metadata definitions which must be stored in the registry. Note that the term "structural metadata" refers as a general term to all structural metadata (key families/data structure definitions, metadata structure definitions, codelists, concept schemas, etc.)

At a minimum, structural metadata definitions stored in the registry must be accessible via an HTTP/HTTPS GET in the form of an SDMX-ML structure message, or may be submitted as part of the registry interface. The use of SOAP is also recommended, as described in the SDMX Web Services Guidelines. Other protocols may be supported. The message may contain all structural metadata items for the whole registry, structural metadata items for one maintenance agency, or individual structural metadata items.

Structural metadata items may only be modified by the maintenance agency which created them. They may only be deleted by the agency which created them, and then only if they are not referred to by other items. The level of granularity for the maintenance of SDMX Structural Metadata objects in the registry is the Maintainable Artefact. In other words, any function such as add, modify, delete is at the level of the Maintainable Artefact. For instance, if a Code is added to a Code List, or the Name of a Code is changed, the registry function is "Modify Codelist" and whole of the Code List will be presented in the interface.

The following table lists the Maintainable Artefacts.

Maintainable Artefacts		Content
Abstract Class	**Concrete Class**	
Item Scheme	Code List	Code
	Hierarchical Codelist	Hierarchy
	Concept Scheme	Concept
	Category Scheme	Category
	Organisation Scheme	Data Provider, Data Consumer, Maintenance Agency
	Reporting Taxonomy	Category
	Object Type Scheme	Identifiable Object Type
Structure	Key Family	Key Descriptor, Group Key Descriptor, Dimension, Attribute Descriptor, Data Attribute, Measure Descriptor, Measure
	Metadata Structure Definition	Full Target Identifier, Partial Target Identifier, Identifier Component, Report Structure, Metadata Attribute
Structure Usage	Dataflow Definition	
	Metadataflow Definition	
Process	Process	Process Step
Structure Set	Structure Set	Maps of various types
Provision Agreement	Provision Agreement	

Table 3: Table of Maintainable Artefacts for Structural Definition metadata

6.2 Item Scheme, Structure

6.2.1 Structure and Item Scheme: Basic Concepts

The artefacts included in the structural definitions are:

- All types of Item Scheme (Code List, Concept Scheme, Category Scheme, Organisation Scheme, Object Type Scheme)
- All types of Structure (Key Family, Metadata Structure Definition)
- All types of Structure Usage (Dataflow Definition, Metadataflow Definition)

The SDMX Registry must have the ability to support maintenance agencies in their role of defining and disseminating structural metadata artefacts. These include key families, code lists, concepts etc. An authenticated maintenance agency may submit valid structural metadata definitions which must be stored in the registry.

Structural metadata definitions stored in the registry must be accessible via an HTTP/HTTPS GET and optionally via SOAP (as explained in the SDMX Web

Services Guidelines) or via other protocol. The message may contain all structural metadata items for the whole registry, structural metadata items for one maintenance agency, or individual structural metadata items.

Structural metadata items may only be modified by the maintenance agency which created them. They may only be deleted by the agency which created them if they are not referred to by other items.

The SDMX-RR must support a globally unique identification scheme such that all definitions of SDMX artefacts are globally unique and resolvable. The SDMX-RR architecture must be able to resolve the unique identifier of an SDMX artefact to produce an XML definition of that artefact.

6.2.2 Structure and Item Scheme Model

See the SDMX Information Model packages:

- Base – Item Scheme pattern
- Base – Organisations
- Base – Structure pattern
- Code List
- Concept Scheme
- Category Scheme
- Key Family
- Metadata Structure Definition

6.2.3 Structure and Item Scheme: Functions and Behaviour

The table below defines the data that could be present for the various functions. The data required for the reference to “Identifiable, “Versionable”, and “Maintainable” (and which of the attributes are mandatory) are specified in Table 1 above.

An (s) after the Function indicates that multiple objects of the specified type may be presented in the one function call.

577
578

Function	Comprises object types	Data attributes	Status	Notes
Reference to Item scheme and Item in this table infers functions generic to any type of Item Scheme i.e. Code List, Category Scheme, Organisation Scheme, Object Type Scheme				
Add Item Scheme(s)		Item Scheme type	M	
		Maintainable	M	
		- final	C	Set to “true” if the specification is final and cannot be modified
	Item	Parent Item Id	C	Required if the scheme is hierarchic
		Versionable	M	
Modify Item Scheme(s)		Same data as for Add Item Scheme as the whole scheme is replaced	M	Cannot be modified if the “final” attribute is “true”
Delete Item Scheme(s)		Item Scheme urn or Maintainable or both		Deletion is not allowed if the Item Scheme or one of the Items in the scheme is referenced in the registry
Item Scheme – functions specific to a specific type of Item Scheme				
Concept Scheme	Concept	Representation - Date Range - Numeric Range - Pattern - Sequence		Required if a core representation is specified.

Function	Comprises object types	Data attributes	Status	Notes
		Representation is Item Scheme – reference to one of: - Code List - Category Scheme - Concept Scheme - Organisation Scheme		The Id of the relevant Item Scheme (see Table 2). Required if the core representation is an Item Scheme.
Add/Modify/Delete Data Provider Scheme	Data Provider			This is an Organisation Scheme which contains data providers. Therefore, the functions are as for Item Scheme.
Add/Modify/Delete Object Type Scheme	Identifiable Object Type			This is an Object Type Scheme which contains Identifiable Object Types. Therefore, the functions are as for Item Scheme.
Add/Modify/Delete Maintenance Agency Scheme	Maintenance Agency			This is an Organisation Scheme which contains maintenance agencies. Therefore, the functions are as for Item Scheme. For SDMX there is only one such scheme and it is maintained by SDMX
Add Key Family		Maintainable	M	
		- final	C	Set to “true” if the specification is final and cannot be modified.
	Key Descriptor	Identifiable – no data required as the Id is implied	M	
	- Dimension(s)	Identifiable – no data required as the Id is implied	M	The Dimensions must be present in the sequence required.
		Concept reference	M	The Id of the Concept – see Table 2

Function	Comprises object types	Data attributes	Status	Notes
		Representation - Date Range - Numeric Range - Pattern - Sequence	C	Required if a local representation is specified.
		Representation - Code List reference		The Id of the Code List (see Table 2). Required if the local representation is a Code List.
	Group Key Descriptor(s)	Identifiable –Id only	C	Only present if groups are required.
	- Dimension(s)	Dimension reference	M	Must reference one or more of the Dimensions of the Key Descriptor.
	Attribute Descriptor	Identifiable – no data required as the Id is implied	C	
	- Data Attribute(s)	Identifiable – no data required as the Id is implied	M	
		Concept reference	M	The Id of the Concept – see Table 2
		Representation - Date Range - Numeric Range - Pattern - Sequence	C	Required if a local representation is specified.
		Representation - Code List reference		The Id of the Code List (see Table 2). Required if the local representation is a Code List.
	Measure Descriptor	Identifiable – no data required as the Id is implied	C	
	- Measure(s)	Identifiable – no data	M	

Function	Comprises object types	Data attributes	Status	Notes
		required as the Id is implied		
		Concept reference	M	The Id of the Concept – see Table 2
		Representation - Date Range - Numeric Range - Pattern - Sequence	C	Required if a local representation is specified.
		Representation - Code List reference	C	The Id of the Code List (see Table 2). Required if the local representation is a Code List.
Modify Key Family		Same data as for Add Key Family as the whole definition is replaced		Cannot be modified if the "final" attribute is "true".
Delete key Family		Key Family urn		Cannot be deleted if the Key Family is referenced from a Dataflow Definition.
Add Metadata Structure Definition		Maintainable	M	Set to "true" if the specification is final and cannot be changed.
		- final	C	
	Full Target Identifier	Identifiable – Id only	M	
	- Identifier Component(s)	Identifiable – no data required	M	
		Object type reference	M	The id of the Identifiable Object Type – see Table 2.
		Item Scheme – reference to one of: - Code List - Category Scheme - Concept Scheme - Organisation Scheme	C	The Id of the relevant Item Scheme (see Table 2).

Function	Comprises object types	Data attributes	Status	Notes
	Partial Target Identifier(s)	Identifiable – Id only	C	
	- Identifier Component(s)	Identifier Component reference	M	Must reference one or more of the Identifier Components of the Full Target Identifier.
	Report Structure	Identifiable – Id only	M	
	- Metadata Attributes	Identifiable – no data required	M	
		Concept reference	M	The Id of the Concept – see Table 2
		Representation - Date Range - Numeric Range - Pattern - Sequence	C	Required if a local representation is specified.
		Representation - Code List reference	C	The Id of the Code List (see Table 2). Required if the local representation is a Code List.

Table 4: Table of functions and data for Structure and Item Scheme

6.3 Structure Usage

6.3.1 Structure Usage: Basic Concepts

The Structure Usage defines, in its concrete classes of Dataflow Definition and Metadataflow Definition, which flows of data and metadata use which specific Structure, and importantly for the support of data and metadata discovery, the Structure Usage can be linked to one or Category in one or more Category Scheme. This gives the ability for an application to discover data and metadata by "drilling down" the Category Schemes.

6.3.2 Structure Usage Model

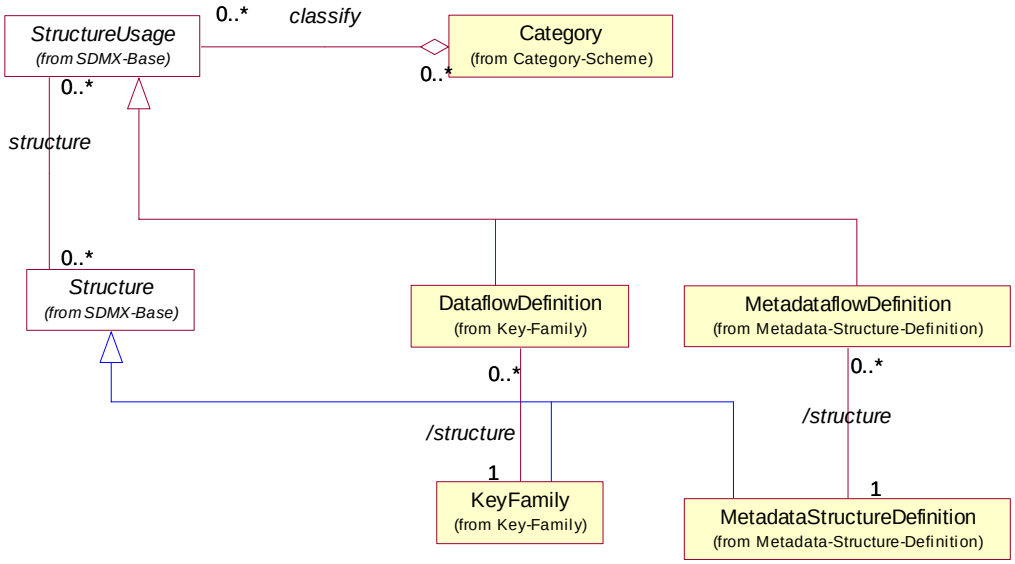


Figure 3: Class diagram of links from Structure Usage in the SDMX-IM

In addition to the maintenance of the Dataflow Definition and the Metadataflow Definition the following links require to be maintained in the registry:

- Dataflow Definition to Key Family
- Category to Dataflow Definition
- Metadataflow Definition to Metadata Structure Definition
- Category to Metadataflow Definition

602 ### 6.3.3 Structure Usage: Functions and Behaviour
603

Function	Linked object types	Data attributes	Status	Notes
Add Dataflow Definition(s)	Dataflow Definition	Maintainable	M	
Modify Dataflow Definition(s)	This is a complete substitution of the Dataflow Definition by the revised specification	Maintainable	M	
Delete Dataflow Definition		urn of the Dataflow Definition		The Dataflow Definition cannot be deleted if there are any Data Sets registered which reference the Dataflow Definition
Add Metadataflow Definition(s)	Metadataflow Definition	Maintainable	M	
Modify Metadataflow Definition(s)	This is a complete substitution of the Metadataflow Definition by the revised specification	Maintainable	M	
Delete Metadataflow Definition		urn of the Metadataflow Definition		The Metadataflow Definition cannot be deleted if there are any Metadata Sets registered which reference the Metadataflow Definition
Link Objects	Category to Key Family	urn of the Category and urn of the Key Family	M	
	Category to Dataflow Definition	urn of the Category and urn of the Dataflow Definition	M	
	Category to Metadata Structure Definition	urn of the Category and urn of the Metadata Structure Definition	M	

Function	Linked object types	Data attributes	Status	Notes
	Category to Metadataflow Definition	urn of the Category and urn of the Metadataflow Definition	M	

Table 5: Table of functions and data for data and metadata flows

7 DATA AND METADATA PROVISIONING

7.1 Provision Agreement, Data and Metadata Sources

7.1.1 Provisioning Agreement: Basic concepts

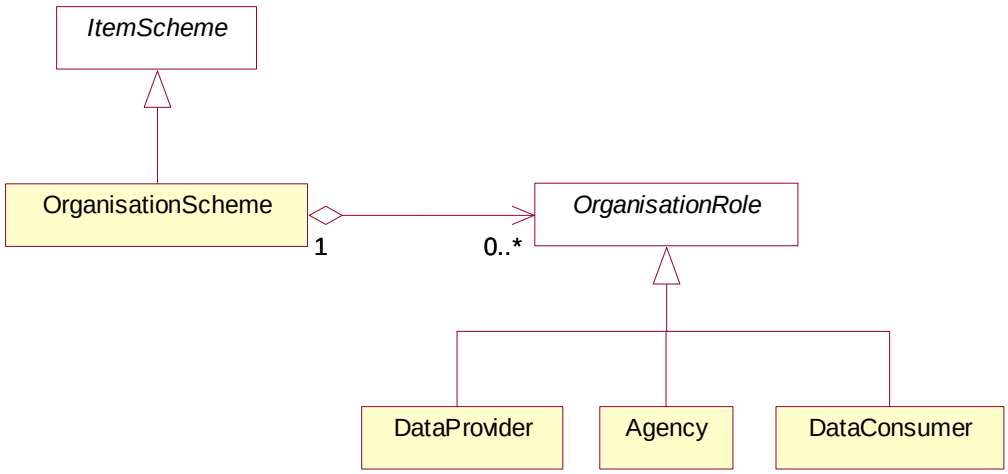
Data provisioning defines a framework in which the provision of different types of statistical data and metadata by various data providers can be specified and controlled. This framework is the basis on which the existence of data can be made known to the SDMX-enabled community and hence on which data can subsequently be discovered. Such a framework can be used to regulate the data content to facilitate the building of intelligent applications. It can also be used to impose service level agreements, or other provisioning agreements in those scenarios that are based on legal directives. Additionally, quality and timeliness information can be supported by this framework which makes it practical to implement information supply chain monitoring.

Note that in the SDMX-IM the class “Data Provider” encompasses both data and metadata and the term “data provisioning” here includes both the provisioning of data and metadata.

Although the SDMX registry work has been conducted with the data-sharing “pull” model in mind, the information model does have limited support for the role of data consumer, and so the specification could be extended to better support “push” exchanges (bilateral and gateway scenarios). It should be noted, too, that in any exchange scenario, the registry functions as a repository of structural metadata.

7.1.2 Provisioning Agreement Model

An organisation (or sub-unit) which publishes statistical data and wishes to make it available to an SDMX enabled community is called a Data Provider. In terms of the SDMX Information model, an Organisation can act in many roles (OrganisationRole) and the diagram below depicts three cases of these: MaintenanceAgency (abbreviated to Agency), DataProvider and DataConsumer. For those readers unfamiliar with the term maintenance agency, it is part of an organisation which defines structural definitions such as code lists and key families (maintainable artefacts). The OrganisationScheme is a special kind of ItemScheme which can hold zero or more DataProviders, Agencies or DataConsumers (since these are types of OrganisationalRole).



The diagram below is a logical representation of the data required in order to maintain Provision Agreements.

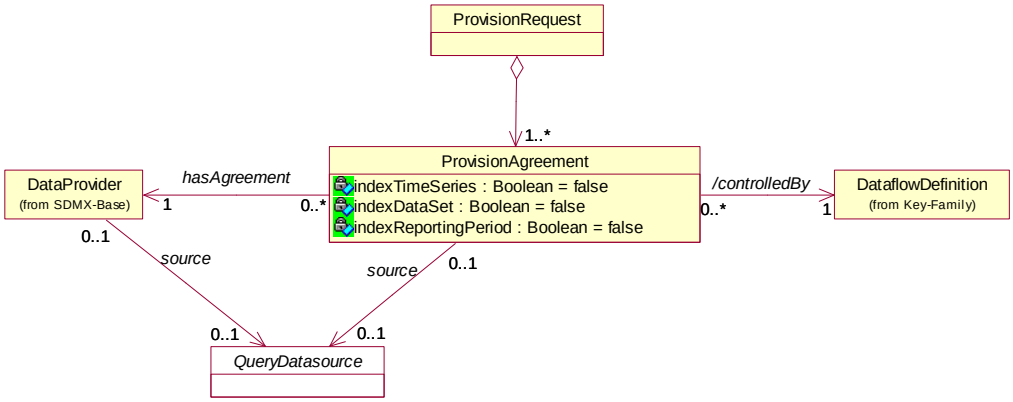


Figure 4: Logical class diagram of the use of the Provision Request for submitting a Provision Agreement

A Provision Request contains a number of Provision Agreements, for each of which there must be a reference to a Data Provider and a Dataflow Definition. The Data Provider and the Dataflow Definition must exist already in order to set up a Provision Agreement.

The Provision Agreement has three Boolean attributes which may be present to determine how an SDMX compliant Dataset indexing application must index any Datasets which are registered against it. The indexing application behaviour is as follows:

If indexTimeSeries is true, a compliant indexing application must index all the time series keys when a Dataset is registered.

If indexDataSet is true, a compliant indexing application must index the range of actual (present) values for each dimension of the Dataset. Note that this requires much less storage than full key indexing, but this method cannot guarantee that a specific combination of Dimension values (the Key) is actually present in the Dataset.

If indexReportingPeriod is true then a compliant indexing application must index the time period range(s) for which data are present in the Dataset.

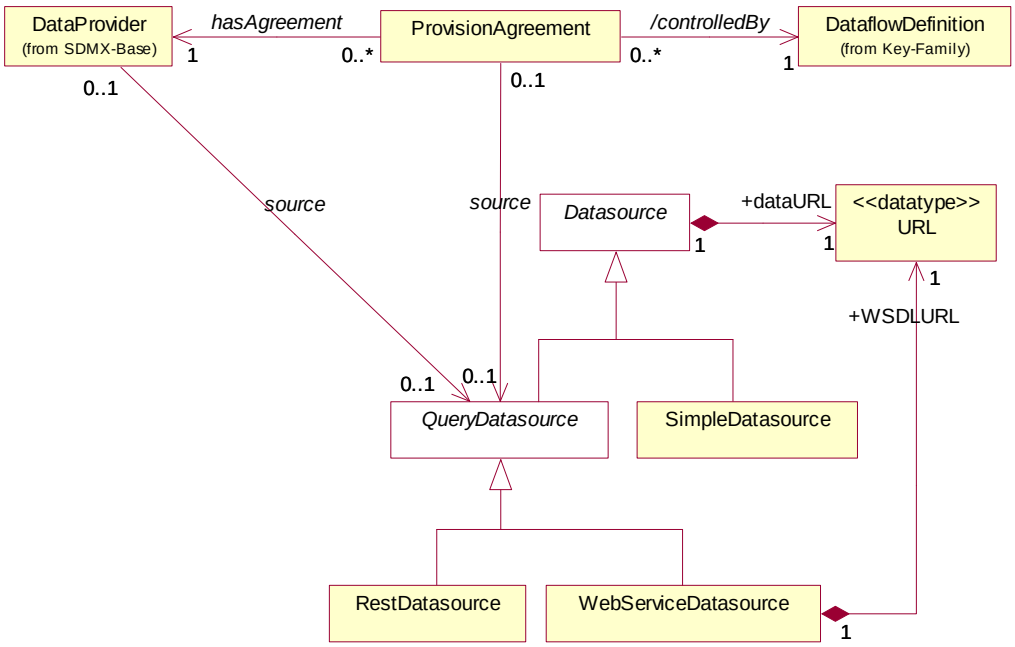


Figure 5: Logical class diagram of data sources

The Query Datasource is an abstract class that represents a data source which can understand an SDMX-ML query and respond appropriately. There are two types of query data source, the Rest Datasource which is accessed via the REST protocol (HTTP/HTTPS) and the Web Service Datasource which supports SOAP based queries. Each of these different data sources inherit the dataURL from Datasource, and the Web Service Datasource has an additional URL to locate a WSDL document to describe how to access it. All other supported protocols are assumed to use the Simple Datasource URL.

A Simple Datasource is used to describe physical SDMX-ML files that are available at a URL. If the URL changes with successive releases of data then the Simple Datasource must be passed to the SDMX registration service for each release (see 8 below).

A Data Provider may use one database system to supply all its SDMX datasets. In this case, it must be a QueryDatasource, and the query passed to it must specify (as a minimum) which data flow must be returned. (Since QueryDatasource is abstract, the Data Provider must specify one of RestDatasource or WebServiceDatasource).

A Data Provider may opt to specify a Datasource per data flow. This would be done by attaching the Datasource to the Provision Agreement.

7.1.3 Provisioning Agreement: Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Add Provision Request	Provision Agreement(s)	DataflowDefinition Reference	M	URN or Maintainable reference to Dataflow Definition
		Data Provider Reference	M	URN. The Data Provider is maintained in an Organisation Scheme
		indexTimeSeries	C	Enables fine-grained indexing of data
		indexDataSet	C	Enables coarse-grained indexing of data
		indexReportingPeriod	C	Enables time-based indexing of data
	QueryDatasource		C	Specifies the data source
		dataUrl	M	Data access URL for data source
		WSDLUrl	C	WSDL URL for data source
Modify Provision Request	Provision Agreement(s)	URN	M	This is a complete replacement and the content is the same as for Add Provision Agreement
Delete Provision Request	Provision Agreement(s)	URN	M	Removes all ProvisionAgreements specified in the Request
Set Datasource	Constrainable Reference	URN	M	Identifies a Data Provider or Provision Agreement
	Datasource	dataURL	M	Data access URL for data source
		WSDLUrl	C	WSDL URL for data source
Unset Datasource	Constrainable Reference	URN	M	Identifies a Data Provider or Provision Agreement

Table 6: Table of functions and data for Provision Agreement

7.2 Data and Metadata Constraints

7.2.1 Data and Metadata Constraints: Basic Concepts

Constraints are metadata about data provisioning artefacts defined above (data providers, data flows and/or provision agreements) which restrict and define these artefacts. Specification of constraints gives enhanced semantics to data provisioning artefacts, enabling more automated processing of the "information supply chain".

Constraints comprise the specification of subsets of key values or attribute values that are contained in a Datasource or is to be provided for a Dataflow or Metadataflow Definition. This is important metadata because the full range of possibilities which is implied by the Key Family Definition (e.g. the complete set of valid keys is the Cartesian product of all the values in the code lists for each of the Dimensions) is often more than is actually present in any specific Datasource, or more than is intended to be supplied according to a specific Dataflow Definition.

Often a Data Provider will not be able to provide data for all key combinations, either because the combination itself is not meaningful, or simply that the provider does not have the data for that combination. In this case the Data Provider would constrain the Datasource (at the level of the Provision Agreement or the Data Provider) by supplying metadata that defined the key combinations or cube regions that are available.

Furthermore, it is often useful to define subsets or views of the Key Family which restrict values in some code lists, especially where many such subsets restrict the same Key Family Definition. Such a view is called a Dataflow Definition, and there can be one or more defined for any Key Family.

Whenever data is published or made available by a Data Provider, it must conform to a Dataflow Definition (and hence to a Key Family). The Dataflow Definition is thus a means of enabling content based processing. A Dataflow Definition may carry confidential data, and this needs to be known to control access to potential consumers.

7.2.2 Data and Metadata Constraints: Model

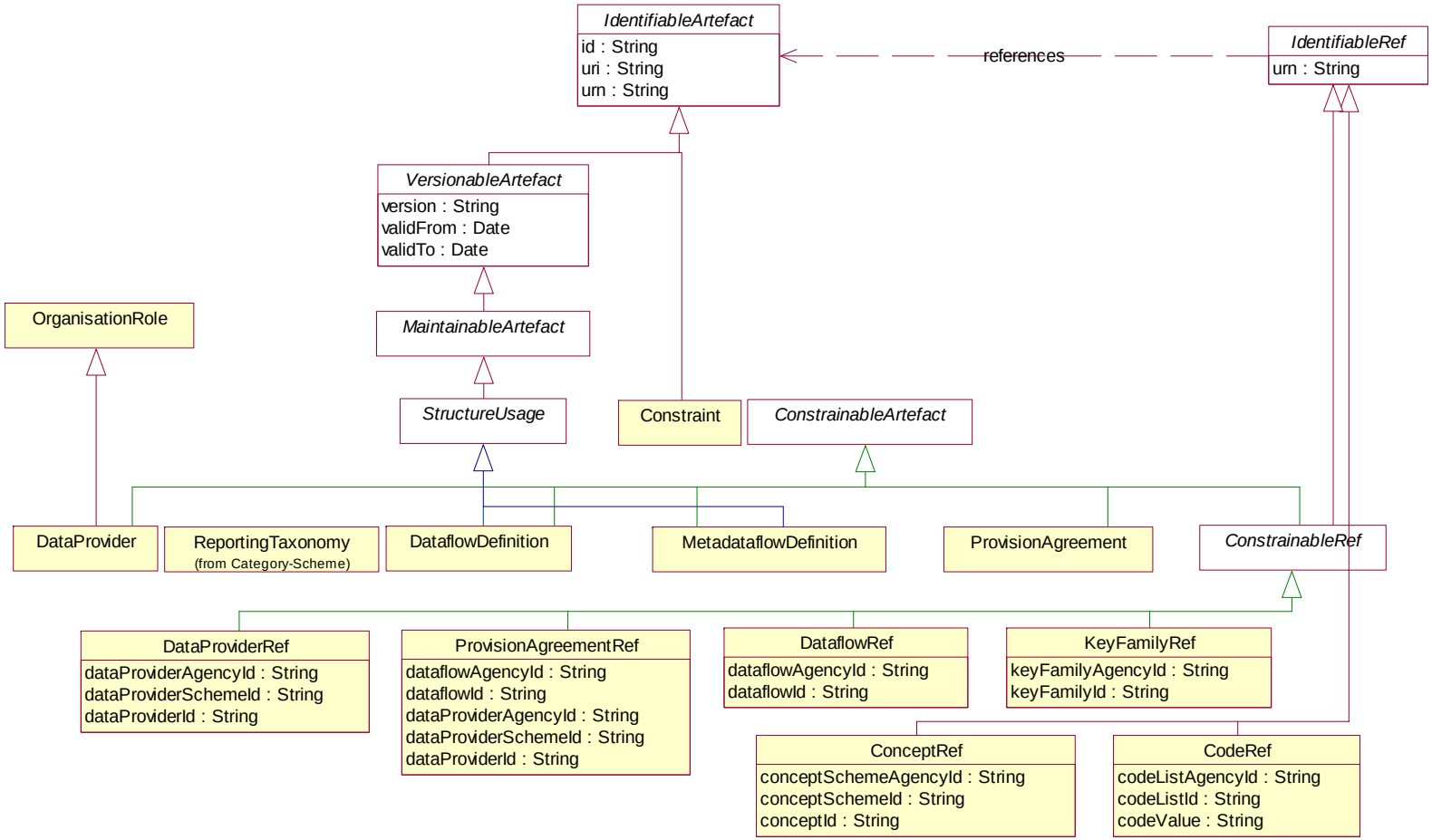


Figure 6: Class diagram inheritance between constrainable artifacts

The class diagram above shows that DataProvider, DataflowDefinition, MetadataflowDefinition, and ProvisionAgreement are all concrete sub-classes of ConstrainableArtefact. Note also the reference classes which inherit from ConstrainableRef: these are used as proxies to refer to the provisioning classes.

All sub-classes of ConstrainableArtefact may have one ContentConstraint and any number of AttachmentConstraints. The content constraint is used for defining legal values that dimension, attributes and measures may take. The content constraint may apply to a DataflowDefinition, thus defining a sub-set of the KeyFamily. If it is applied to a DataProvider or ProvisionAgreement then it specifies the content of the respective QueryDatasource. Constraints are also used in dataset registration and querying.

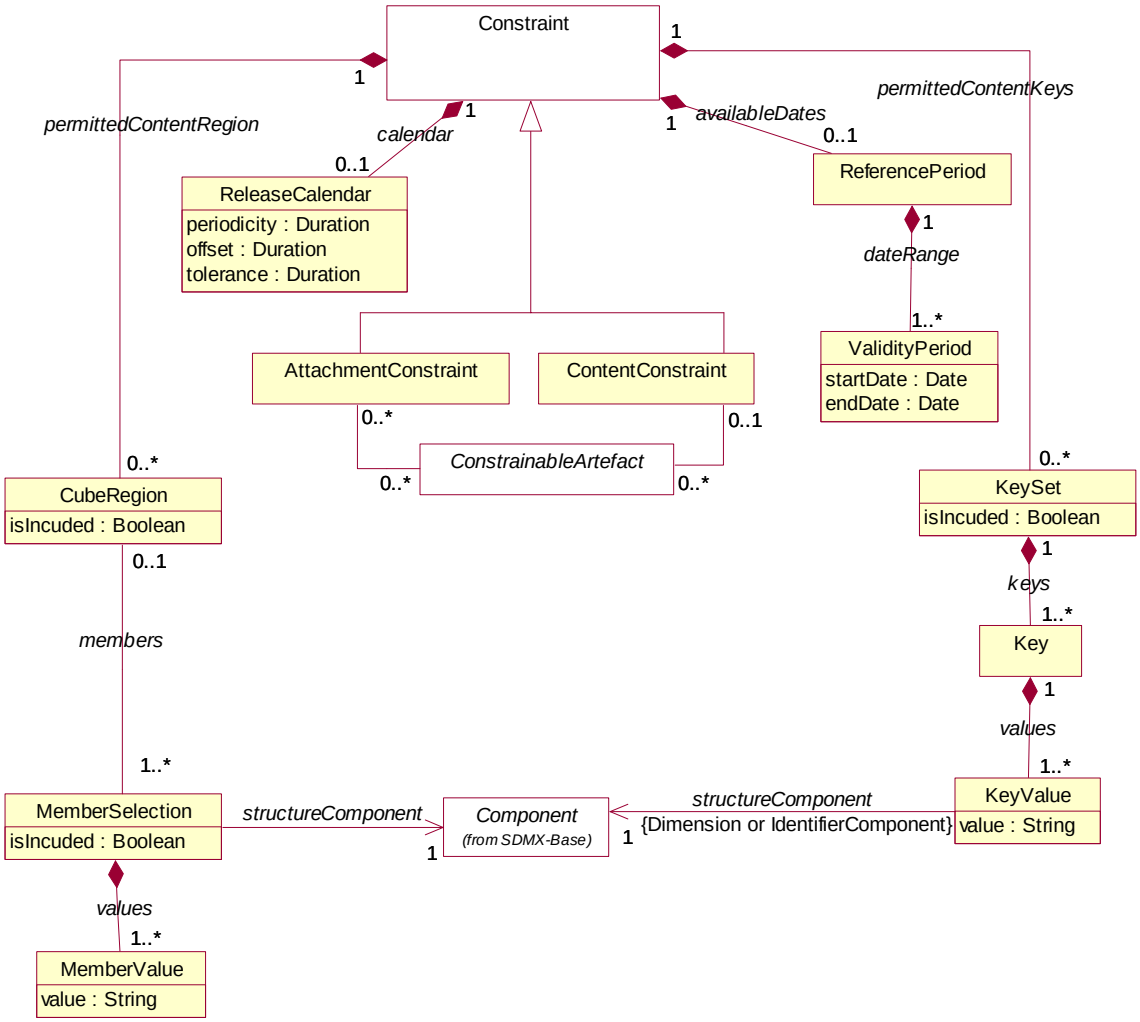


Figure 7: Class diagram showing Constraints

Any number of attachment constraints may be defined. They are used to sub-set the key family for metadata attribute attachment, rather than for constraining content. Attachment constraints are not of direct interest to SDMX registry services.

The content constraint comprises four parts, one or more of which must be specified

Cube Region constraints limit the values that a dimension or coded attribute (Component) may take in a dataset which corresponds to the constrained data flow definition. This is done by specifying the legal values (MemberValue) for any dimension, attribute or measure. During the provisioning process, a Cube Region constraint can be used to define a sub-region of the whole key family. A simple example is that for an external debt dataflow, a cube region constraint could define that the "reporting role" dimension is "creditor", thus no valid dataset would be allowed to contain "debtor" data. This also can be applied to attributes and measures: it allows the definition of, say, a single currency for EXT_DEBT statistics, even though at a national level a different dataflow could be used that published in national currency.

The Key Set constraint performs a similar function, except that it is used to define legal combinations of values that may exist within a dataset. If KeyValues are specified for all Dimensions, then a time-series is defined. This mechanism can be used for building dependencies between dimensions.

It is important to know which time periods a Provision Agreement covers. There is little point directing data consumers interested in 10 year trends to a data source which has data from 2001 onwards. The ReferencePeriod class must provide a standard way of defining time periods for which data is held. There may be many dis-contiguous periods for a data flow agreement.

If data publication falls under a legal directive or commercial contract, it is important to know when each data provider commits to publishing new data. The ReleaseCalendar must support enough calendaring information to support this. Making the data release process transparent amongst all data providers can provide great encouragement to meet targets.

Three attributes of type Duration (as per XML schema specification) are defined:

attribute	Description	example
periodicity	the period between releases of a dataset	P3M
offset	The period between the first publication date in the year and January 1st.	P7D
tolerance	the point at which the dataset is deemed late	P3D

The above example indicates that a datasource is published every quarter, it is published one week after the beginning of a quarter, and it must be released within three days of the expected publish date.

.

7.2.3 Data and Metadata Constraints: Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Set Constraint		ConstrainableArtefactURN	M	URN of constrainable artefact, one of: Data Provider, Provision Agreement, or Dataflow Definition
	Constraint	Id	M	Id has fixed value "Content" for content constraint, otherwise user defined
		Type	M	Content or Attachment
	Cube Regions(s)		C	Defines a subset of the key family
		isIncluded	M	Is this cube region included or excluded from the whole cube?
	Member Selection(s)	isIncluded	M	Are these concept values included or to be excluded?
		ComponentRef	M	Reference to the Component (Dimension, Attribute or Measure) whose values are specified
	MemberValue(s)	Value	M	The value selected for a component
	Key Set(s)		C	Defines a specific combination of single-valued key family components
		isIncluded	M	Is this key set included or excluded from the whole cube?
	Key(s)		M	
	Key Value(s)	ComponentRef	M	Reference to the Component (Dimension, Attribute or Measure) whose value is specified
		Value	M	The value selected for a component

Function	Comprises object types	Data attributes	Status	Notes
	Release Calendar		C	Defines when data is to be published
		Periodicity	M	the period between releases of a dataset
		Offset	M	The period between the first publication date in the year and January 1st.
		Tolerance	M	the point at which the dataset is deemed late

789 **Table 7: Table of functions and data for Data and Metadata Constraints**

790

8 DATA AND METADATA REGISTRATION

8.1 Basic Concepts

A data provider has published a new dataset conforming to an existing dataflow definition (and hence key family). This is implemented as either a web-accessible SDMX-ML file, or in a database which has a web-services (SOAP or REST) interface capable of responding to and SDMX-ML Query with an SDMX-ML data stream.

The data provider wishes to make this new data available to a wide audience without having to physically distribute the data. To do this, the data provider registers the new dataset with one or more SDMX conformant registries that have been configured with structural and provisioning metadata. In other words, the registry "knows" the data provider and "knows" what data-flows the data provider has agreed to make available.

SDMX-RR supports dataset and metadata set registration via the Registration Request, which can be created by the data provider (giving the data provider maximum control) or generated by a registry service on behalf of the data provider. The registry responds to the registration request with a registration response which indicates if the registration was successful. In the event of an error, the error messages are returned as a Registry Exception within the response.

8.2 The Registration Request

8.2.1 Registration Request Model

The following UML diagram shows the composition of the registration request. Each request is made up of one ore more Registrations, one per dataset to be registered. The Registration has a validity period (valid from/to) which instructs the registry whether the registration is active or not. This allows a data provider to retire datasets after they are no longer current. The last updated date is needed during the discovery process to make sure the client knows which data is freshest.

The Registration has an action attribute which takes one of the following values:

`urn:sdmx:registry:register:action:append,`	Add this registration to the registry
`urn:sdmx:registry:register:action:replace`	As above, but replace any registrations for same Provision Agreement Reference
`urn:sdmx:registry:register:action:delete`	Delete registrations for Provision Agreement Reference

Each Registration may optionally specify a Datasource. If a QueryDatasource has already been specified at the provision agreement (or data provider) level, then this may also be used. When a dataset is published as an SDMX-ML file accessible through a web server, then a SimpleDatasource would be specified with the Registration.

The UML diagram also shows that the Registration must specify one ConstrainableRef object. The concrete classes of ConstrainableRef are DataProviderRef, ProvisionAgreementRef and DataflowRef (or MetadataflowRef). Normally, when a dataset is registered, both the data provider and the

dataflow/metadataflow are known, so the Registration would specify a ProvisionAgreementRef. A DataProviderRef would indicate that the contents of the dataset are unknown and a DataflowRef (or MetadataflowRef) would indicate that the data provider is unknown: there may be use cases for these two scenarios, but this is not the focus of the current implementation. The reference objects are just simple references to existing objects in the registry. They contain the URN of the object to which they refer and also have the necessary attributes so that the URN can be calculated. For example, a DataflowRef has attributes or dataflowAgencyId and dataflowId from which a valid URN can be constructed automatically.

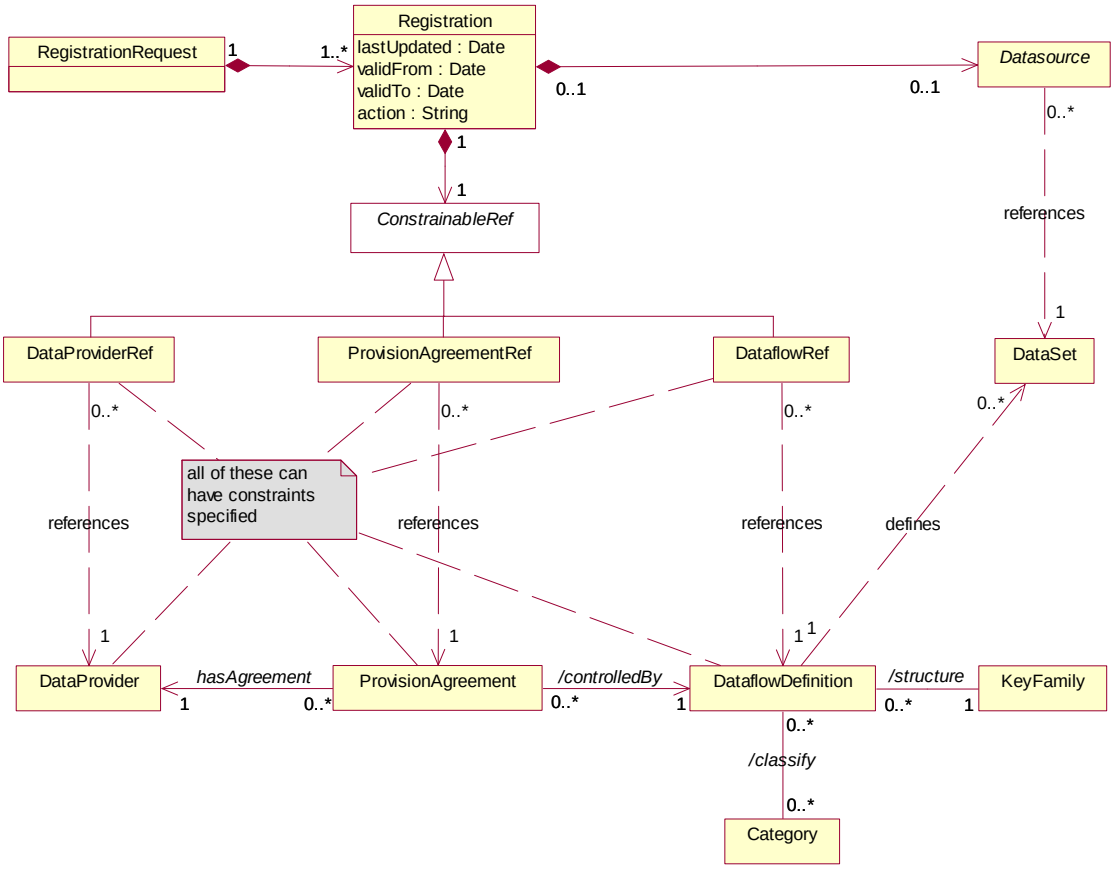


Figure 8: Class diagram showing the registration request

8.2.2 Registration Constraints

All ConstrainableRef objects can carry constraints (as discussed earlier for provisioning) although this is not shown on the preceding UML diagram. During the registration process, content constraints defined in the Registration Request are used to define the contents of the dataset to the registry. This is use case dependent: if the data provider would like clients to be able to locate an individual time-series via the registry, then each time-series key must be represented in a Key Set constraint. If the data provider is registering a database as a QueryDatasource, it is probable that a Cube Region constraint will be used to outline the probablility of finding a particular time series. In scenarios where the registry is used for feeding a database, then content constraints are less important, as the database most likely would be fed using all data available.

857
858 ReferencePeriod constraints would also be specified to indicate the first and last time
859 periods for which data is available.
860
861

8.2.3 Registration Request: Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Submit Registration Request	Registration(s)	lastUpdated	C	The date of when the dataset was last updated
		validFrom	C	The date from which the registry can broadcast the dataset
		validTo	C	The date at which the registry must retire the dataset
		Action	M	Append, Replace or Delete
	Datasource		C	Specifies the data-source for the dataset. If not specified, a data source must have been set up during provisioning
		dataUrl	M	Data access URL for data source
		WSDLUrl	C	WSDL URL for data source
	ConstrainableRef	URN	M	Reference to a constrainable object: one of Data Provider, Provision Agreement or Dataflow Definition
	Constraint		C	Identical to section 6.3.2

Table 8: Table of functions and data for Registration Request

8.2.4 Registration Registry Service

As mentioned earlier, a registry service must be provided which can take a URL of an SDMX-ML file (in generic format) and create a Registration Request from the contents of the file, then perform registration with the request. This service takes information to construct the Registration Request from the SDMX-ML message header and from the dataset itself. The mapping is shown in the table below:

SDMX-ML Element/@attribute	Registration Request class and attribute
Sender/@id	ProvisionAgreementRef.dataProviderId
DataSetAgency	ProvisionAgreementRef.dataflowAgencyId
DataSetID	ProvisionAgreementRef.dataflowId
Extracted	Registration.lastUpdated
ReportingBegin	TimePeriod.startDate
ReportingEnd	TimePeriod.endDate
Value/@concept	Key.ConceptRef.conceptId
Value/@value	Key.ConceptRef.CodeRef.codeValue

The last two rows of the table represent the values for time-series keys: each complete key is built and assembled into a key set constraint.

8.3 Registration Response

8.3.1 Registration Response Model

After a registration request has been submitted to the registry, a response is returned to the submitter indicating success or failure. Given that a registration request can hold many Registrations, then the response must give a result for each. The Registration Result class has a status field which is either set to “Success” or “Failure”.

If the registration has succeeded, a ProvisionAgreementRef (subclass of ConstrainableRef) will be returned - this holds the URN of the newly registered dataset plus a Datasource holding the URL to access the dataset or metadataset.

In the event of registration failure, a set of Registry Exceptions are returned, giving the error messages that occurred during registration. It is entirely possible when registering a batch of datasets, that the response will contain some successful and some failed statuses. The UML for the response is shown below:

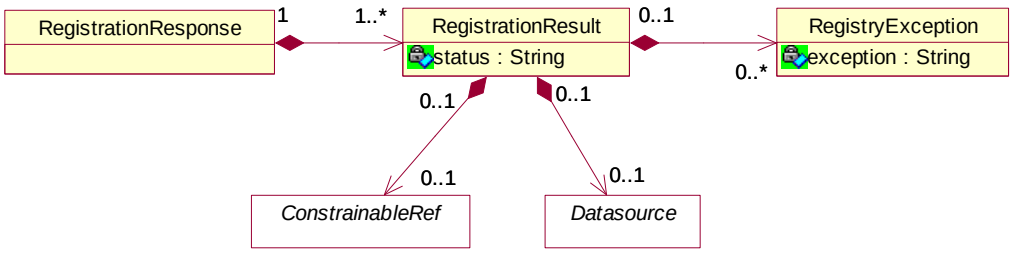


Figure 9: Class diagram showing the registration response

8.3.2 Registration Response: Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Consume Registration Response	Registration Response		M	
	Registration Result(s)		M	One per dataset registered
		Status	M	`urn:sdmx:registry:status:success` or `urn:sdmx:registry:status:failure`.
	Registry Exception(s)		C	On failure
		Exception	M	The error message
	Datasource		C	On success, this gives the datasource that is registered. Note this could have been set up at provisioning time.
		dataUrl	M	Data access URL for data source
		WSDLUrl	C	WSDL URL for data source
	ConstrainableRef		C	On success
		URN	M	Reference to a constrainable object: usually Provision Agreement Ref. This handle can be used to replace or delete the registration later.

Table 9: Table of functions and data for Registration Response

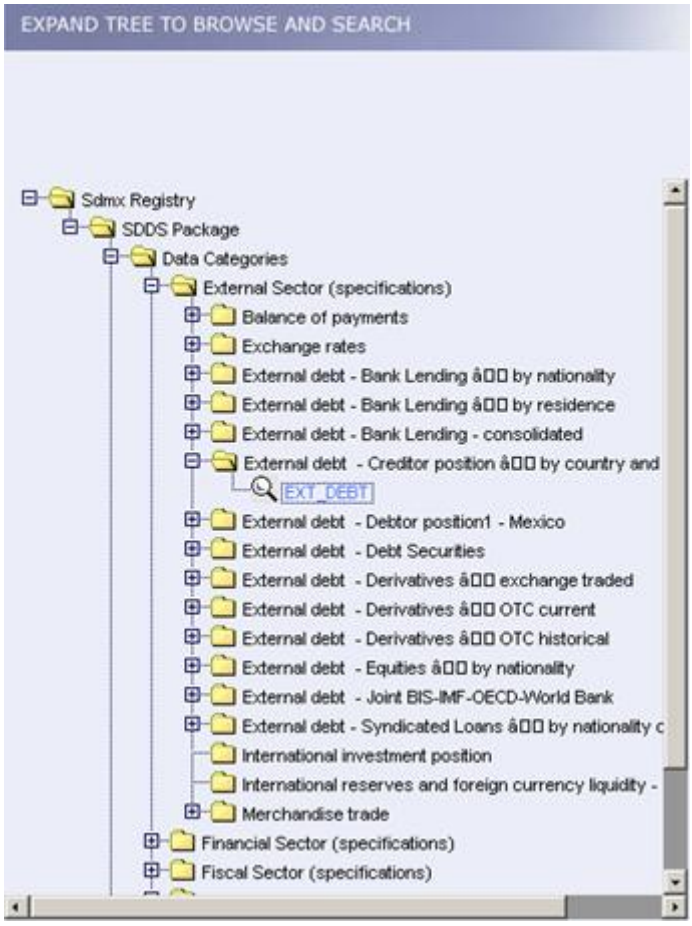
9 DATASET AND METADATA SET DISCOVERY

So far, we have described the ability to set up the SDMX-RR with structural and provisioning metadata, and how to register datasets and metadata sets against this framework. Now it's time to discuss the process of searching for datasets in the registry, referred to as discovery.

One of the main functions of the SDMX-RR is to aid in discovering data. This process may be instigated by either human users via a web interface, or via other application computer processes. The steps involved are outlined below.

9.1 Finding the Data and Metadata We Want

The user (or client program) may not know in advance the dataflow definition or key family that has been used to structure the dataset required. Browsing through subject matter domain categories will lead to dataflows and metadataflows which have been linked to the categories. The image below, taken from the SDMX case-study demonstration, shows how a dataflow called EXT_DEBT has been located by browsing through subject matter domain categories.

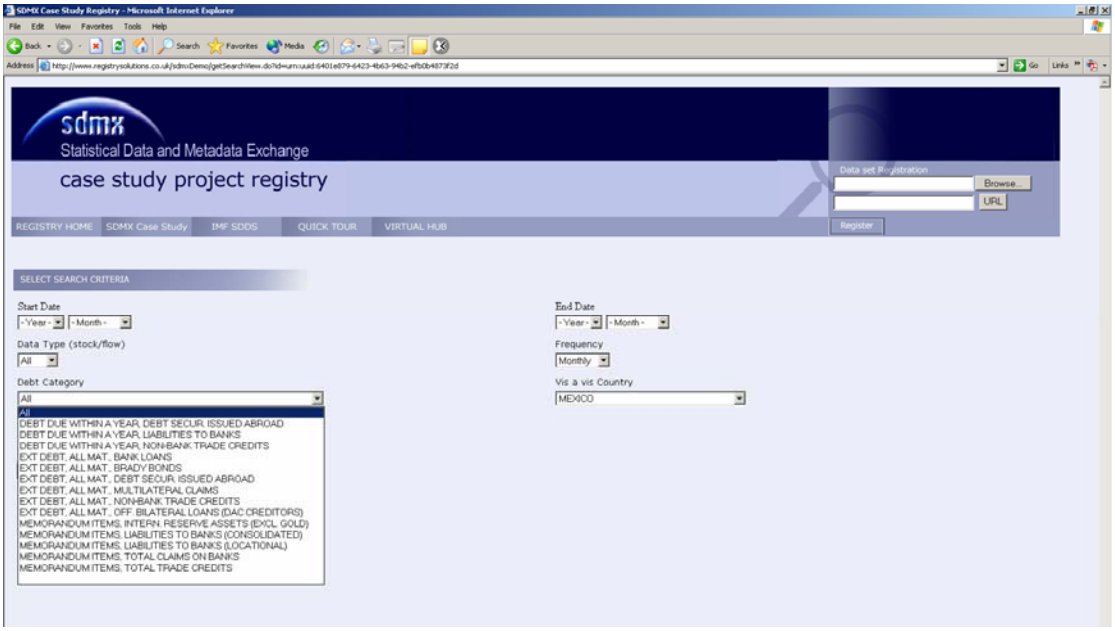


Alternatively, the end-user may be aware of some concepts that they are looking for. A simple application could suggest key families which use these concepts, and thus suggest dataflows which use these key families.

9.2 Building a Query

The registry supports queries at the key family/metadatastructure definition, dataflow/metadataflow, and provision agreement level. Key family queries are the most general and will search amongst all dataflows that use the specified key family. (The parallel is metadata structure definition queries which search amongst all metadataflows). A dataflow or metadataflow query will only target a specific dataflow or metadataflow, and will include datasets or metadatasets from any data provider (this is also true for the key family/metadata structure definition query). A provision agreement query is limited to a specific dataflow/metadataflow from a specific data provider.

An end-user application could construct a search screen based on the underlying key family, with drop-down list for each dimension to allow dimension values to be specified for the query. The following image (again from the SDMX case-study demonstration) shows a screen which has been constructed around the EXT_DEBT dataflow and allows users to select values to build a query.

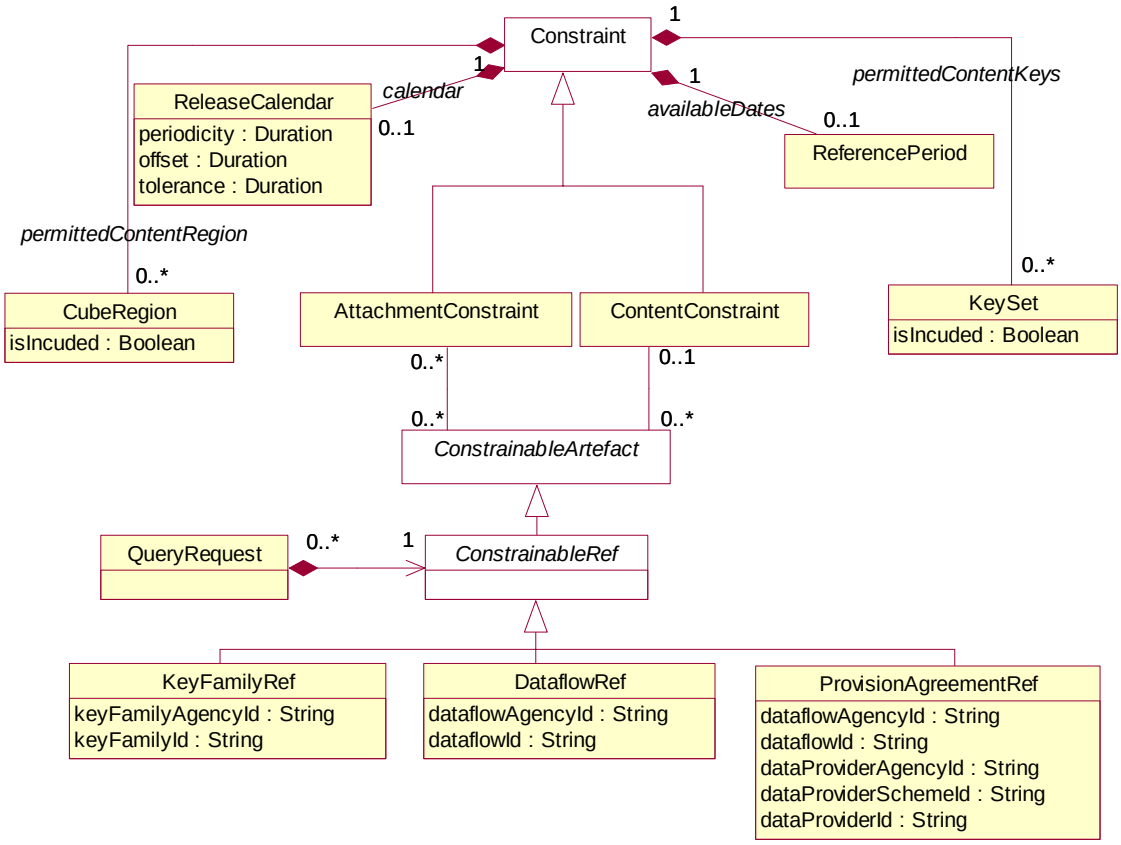


In the preceding image, one can see that the drop-down lists have been populated with all the possible values that a dimension can take. Because dataflows are constrained views of key families, the drop-down lists can actually be populated with code values that are legal for the selected dataflow. This sub-setting of code-lists via constraints on the dataflow reduces the possibility of searching for data which does not exist (i.e. sparse areas).

The user input from such a selection screen is then used to construct either cube region or key set content constraints (as described earlier in the data provisioning section). Key set constraints are used when searching for an actual time-series key and hence specify values for all dimensions. A cube region constraint is specified for a more general search. In the screen above only the vis-à-vis (partner) country has been specified so this would indicate a cube region constraint.

955 The query may also wish to restrict the search in temporal terms. Start and end dates
956 from the query screen would be transformed into a reference period constraint.

957 9.2.1 The Data/Metadata Query Model



958

959 **Figure 10: Class diagram showing the query request**

960 The preceding UML class diagram shows how the query request is constructed from
961 classes already used in data provisioning and registration.
962

963 ### 9.2.2 Data/Metadata Query Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Submit Query Request	QueryRequest		M	The date of when the dataset/metadata set was last updated
	ConstrainableRef	URN	M	Reference to a constrainable object: one of Data Provider, Provision Agreement or Dataflow/Metadataflow Definition
	Constraint		C	Identical to section 6.3.2

964 **Table 10: Table of functions and data for Query Functions**

965

9.3 Processing the Query

There are three types of query that can be presented to the SDMX-RR v1.0, according to the way that constraints have been used within the QueryRequest.

9.3.1 No Content (Key Set or Cube Region) Constraints

This is the simplest case. The registry just has to find all datasets and datasources that have been registered against the key family, dataflow or provision agreement specified in the query. There may be some trimming of results by the registry to comply with any specified ReferencePeriod constraint. This scenario is useful when the registry is being used to feed a database or cache mechanism. Content based queries will then be handled by the database or cache.

It should be remembered that datasources can be registered against data providers and provision agreements during the provisioning process, and these should be returned in the query result.

9.3.2 Cube Region Content Constraint

The process is as above, except all datasets and datasources must satisfy the cube region constraint. That means any concept values in the cube region must be found in the key set or cube region associated with the datasource or dataset in the registry.

9.3.3 Key Set Content Constraint

This process is as above, except that all concept values found in the individual keys comprising the key set must be present in the cube region or key set constraints registered with the datasource or dataset. In the case of a dataset registered with a key set constraint, when a match occurs then it will be an exact match on the key of a time-series, so the “timeSeriesMatch” attribute can be set to true in the query result. If the match is with datasources registered with only a cube region constraint, it is not certain that the particular combination of dimension values occurs in the same time-series, but only that it is possible. Thus the “timeSeriesMatch” must be set to false.

9.4 Handling the Results

After the query request has been processed by the registry, the result is a set of registry objects which correspond to either QueryDatasources which were registered at provisioning time, or SimpleDatasources that were registered along with datasets.

9.4.1 The Results Model

SDMX-RR v1.0 constructs a response to the query which conforms to the UML model shown below. The registry must make sure that no duplicate datasources are returned in the result set. For each datasource in the resultset, a QueryResult object is created, which contains both datasource information containing the URL of the data file or web-service, and whether the datasource was registered against a key family, dataflow or provision agreement (the ConstrainableRef).

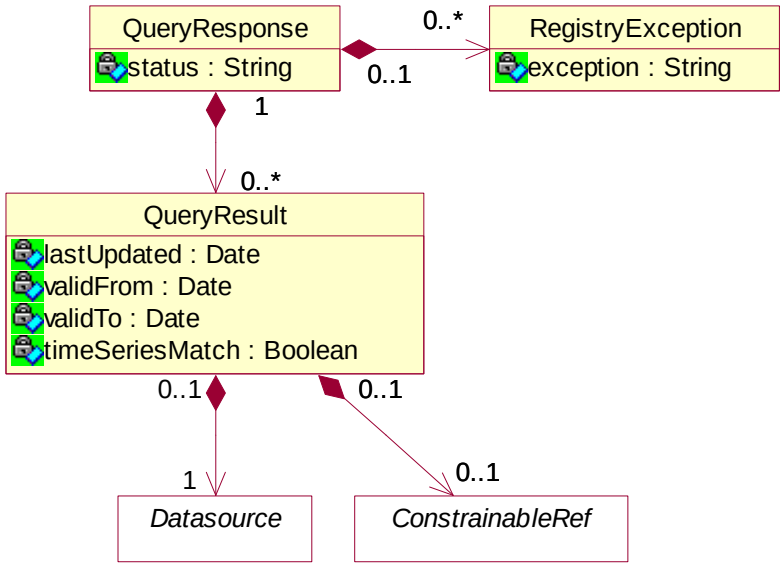


Figure 11: Class diagram showing the query response

Any results which matched a key set query with a key set constrained datasource are marked as an exact time-series match. Information regarding when the last update to the dataset was made, plus the validity period of the datasource are also returned. This information allows consumers to make sure that they are using the most up-to-date information.

9.4.2 Query Response: Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Consume Query Response	QueryResponse		M	
	QueryResult(s)		C	On success
		Status	M	`urn:sdmx:registry:status:success` or `urn:sdmx:registry:status:failure`.
	RegistryException(s)		C	On failure
		Exception	M	The error message
	Datasource		C	On success, this gives the datasource that is registered. Note this could have been set up at provisioning time.
		dataUrl	M	Data access URL for data source
		WSDLUrl	C	WSDL URL for data source
	ConstrainableRef		C	On success
		URN	M	Reference to a constrainable object: usually Provision Agreement Ref. This handle can be used to replace or delete the registration later.

Table 11: Table of functions and data for Query Response

10 SUBSCRIPTION AND NOTIFICATION SERVICE

The contents of the SDMX Registry/Repository will change regularly: new code lists and key families will be published, new datasets and metadata-sets will be registered. To obviate the need for users to repeatedly query the registry to see when new information is available, a mechanism is provided to allow users to be notified when these events happen.

A user can store a subscription in the registry that defines which events are of interest, and either an email and/or an HTTP address to which qualifying events will be delivered. The subscription will be identified in the registry by a URN which is returned to the user when the subscription is created. If the user wants to delete the subscription at a later point, the URN is used as identification. Subscriptions have a validity period expressed as a date range (startDate, endDate) and the registry is free to delete any expired subscriptions, and will notify the subscriber on expiry.

When a registry/repository artefact is modified, any subscriptions which are observing the object are activated, and either an email or HTTP POST is instigated to report details of the changes to the user specified in the subscription. This is called a "notification".

10.1 Subscription

10.1.1 Common Information

Regardless of the type of Registry/Repository events being observed, a subscription always contains:

- A set of URIs describing the end-points to which notifications must be sent if the subscription is activated. The URIs can be either mailto: or http: protocol. In the former case an email notification is sent; in the latter an HTTP POST notification is sent.
- A user-defined identifier which is returned in the response to the subscription request. This helps with asynchronous processing.
- A validity period which defines both when the subscription becomes active and expires. The subscriber will be sent a notification on expiration of the subscription.
- A selector which specifies which type of events are of interest. The set of event types is:

Event Type	Comment
STRUCTURAL_REPOSITORY_EVENTS	Life-cycle changes to Maintainable Artefacts in the structural metadata repository. These include: KeyFamily, CategoryScheme, ConceptScheme, CodeList, MetadataStructureDefinition, DataflowDefinition, OrganisationScheme, MetadataflowDefinition
PROVISIONING_REPOSITORY_EVENTS	Life-cycle changes to ProvisionAgreement. Addition or deletion of Constraints and Datasources to ProvisionAgreement and to

Event Type	Comment
	DataProvider and DataflowDefinition by reference.
ALL_REPOSITORY_EVENTS	Both of the above
DATA_REGISTRATION_EVENTS	Whenever a published dataset is registered. This can be either a SDMX-ML data file or an SDMX conformant database.
METADATA_REGISTRATION_EVENTS	Whenever a published metadataset is registered. This can be either a SDMX-ML reference metadata file or an SDMX conformant database.
ALL_REGISTRATION_EVENTS	Both of the above
ALL_EVENTS	All of the above

10.1.2 Structural Metadata Events

Whenever a maintainable artefact (key family, concept scheme, code list, metadata structure definition, category scheme, etc.) is added to, deleted from, or modified in the structural metadata repository, a structural metadata event is triggered. Subscriptions may observe all such events, or focus on specific artefacts such as a key family and may use a wildcard text pattern match. This is specified in the same way as for SQL, so that JEDH% would match any artefact whose ID begins with JEDH.

10.1.3 Provisioning Metadata Events

Whenever a Provision Agreement is created, updated or deleted and whenever a datasource or constraint is added to or removed from a data provider or dataflow (via a reference) a provisioning metadata event occurs. As for structural metadata events, wildcard expressions can be used to limit the scope of the subscription.

10.1.4 Registration Events

Whenever a dataset or metadata-set is registered a registration event is created. A subscription may be observing all data or metadata registrations, or it may focus on specific registrations as shown in the table below:

Selector	Comment
DataProvider	Any datasets or metadata-sets registered by the specified data provider will activate the subscription
ProvisionAgreement	Any datasets or metadata-sets registered by the specified data provider and dataflow (or metadataflow) will activate the subscription
Dataflow (&Metadataflow)	Any datasets or metadata-sets registered for the specified dataflow (or metadataflow) will activate the subscription
KeyFamily (&MetadataStructureDefinition)	Any datasets or metadata-sets registered for the those dataflows (or metadataflows) that are based on the

Selector	Comment
	specified key family will activate the subscription
Category	Any datasets or metadata-sets registered for the those dataflows (or metadataflows) that fall within the specified category (statistical domain) will activate the subscription

The event will also capture the semantic of the registration; is this a deletion or replacement of an existing registration or a new registration. As for other subscription selection criteria, the use of the SQL wildcard syntax (at least %) should be supported.

10.2 Notification

A notification is an XML document that is sent to a user via email or http POST whenever a subscription is activated. It is an asynchronous one-way message. It has three forms according to the type of event that generated it:

- structural metadata notification
- provisioning metadata notification
- registration notification

All three share some common information:

- Date and time that the event occurred
- The URN of the artefact that caused the event
- The URN of the subscription that produced the notification
- Notification type: Structural, Provisioning or Registration.
- Event type: Add, Replace, or Delete.

Additionally, supplementary information will be contained in the notification as detailed below.

10.2.1 Structural Metadata Notification

The notification will contain the MaintainableArtefact that triggered the event in a form similar to the SDMX-ML structural message (using elements from that namespace).

10.2.2 Provisioning Metadata Notification

The notification will contain the ConstrainableArtefact that triggered the event in the form of an SDMX-ML provisioning message. The ConstrainableArtefacts in question are DataProviderRef, DataflowRef (&MetadataflowRef), and ProvisionAgreement. Any specified datasources and constraints may also be returned (except in the case of a deletion).

10.2.3 Registration Notification

The notification will contain the Registration which has attributes of lastUpdated, validFrom and validTo. It will also contain any Datasource associated with the registration in URL form. Constraints may optionally be returned (in some cases they could be very large).

10.2.4 Subscription and Notification: Functions and Behaviour

Function	Comprises object types	Data attributes	Status	Notes
Submit Subscription Request	SubscriptionRequest		M	
		validFrom	M	Date when subscription comes into effect.
		validTo	M	Date when subscription expires.
		eventType	M	What events are being observed: `urn:sdmx:registry:subscription:event:repository:structure`, `urn:sdmx:registry:subscription:event:repository:provisioning`, `urn:sdmx:registry:subscription:event:repository:all`, `urn:sdmx:registry:subscription:event:registration:data`, `urn:sdmx:registry:subscription:event:registration:metadata`, `urn:sdmx:registry:subscription:event:registration:all`, `urn:sdmx:registry:subscription:event:all`.
		userToken	C	A user-defined token that is returned in the Submit Subscription Response
	EndPoint(s)		M	Where to send the notification, may be repeated
		endPointType	M	`urn:sdmx:registry:subscription:event:endpoint:mailto` or `urn:sdmx:registry:subscription:event:endpoint:http:post`
	MaintainableArtefact		C	For subscribing to structural metadata events
		URN	C	Either URN or objectType, agencyId and id must be specified. Wildcards in SQL form may be used (i.e. %)
		objectType	C	KeyFamily, CodeList, ConceptScheme etc.
		agencyId	C	To identify a MaintainableArtefact
		id	C	To identify a MaintainableArtefact

Function	Comprises object types	Data attributes	Status	Notes
	ConstrainableArtefact		C	For subscribing to structural metadata events and registration events
		URN		Either URN or fully qualified identifying attributes must be specified. Wildcards in SQL form may be used (i.e. %)
		objectType	C	Constrainable type: one of DataProvider, ProvisionAgreement, DataflowDefinition, Metadataflow Definition
		dataProviderAgencyId	C	To identify a DataProvider or ProvisionAgreement
		dataProviderSchemeId	C	To identify a DataProvider or ProvisionAgreement
		dataProviderd	C	To identify a DataProvider or ProvisionAgreement
		dataflowAgencyId	C	To identify a (meta)Dataflow or ProvisionAgreement
		dataflowId	C	To identify a (meta)Dataflow or ProvisionAgreement
		keyFamilyAgencyId		To identify a (meta)Dataflow or ProvisionAgreement
		keyFamilyId		To identify a (meta)Dataflow or ProvisionAgreement
		CategorySchemeAgencyId		To identify a (meta)Dataflow or ProvisionAgreement
		CategorySchemeId		To identify a (meta)Dataflow or ProvisionAgreement
		CategoryId		To identify a (meta)Dataflow or ProvisionAgreement

1111 **Table 12: Table of functions and data for Subscription and Notification**

1112

11 SDMX-RR LOGICAL MODEL AND SDMX-ML

There is a natural correspondence between the logical interfaces described in the preceding sections and the SDMX-ML implementation of them. This section describes the basic document flows from a functional perspective.

Note that all registry interfaces are contained in a single XML element "RegistryInterface", and thus always appear inside a standard SDMX-ML Message as the contents of this element. The message types used for interactions with the SDMX-RR are characterized by which of the child elements of RegistryInterface they contain – each message is only allowed to contain a single child element/registry interface.

Thus, there are a limited number of SDMX-ML Registry/Repository interfaces, or "message types":

- SubmitSubscriptionRequest
- SubmitSubscriptionResponse
- NotifyRegistryEvent
- SubmitRegistrationRequest
- SubmitRegistrationResponse
- QueryRegistrationRequest
- QueryRegistrationResponse
- SubmitStructureRequest
- SubmitStructureResponse
- QueryStructureRequest
- QueryStructureResponse
- SubmitProvisioningRequest
- SubmitProvisioningResponse
- QueryProvisioningRequest
- QueryProvisioningResponse

Several patterns are evident from the naming conventions of the message types/interfaces. First, all interactions with the registry – with the exception of NotifyRegistryEvent – are designed in pairs. The first document – the one which invokes the SDMX-RR interface, is a "Request" document. The message returned by the interface is a "Response" document.

It should be noted that all interactions are assumed to be synchronous, with the exception of NotifyRegistryEvent. This document is sent by the SDMX-RR to all subscribers whenever an even occurs to which any users have subscribed. Thus, it does not conform to the request-response pattern, because it is inherently asynchronous.

The second pattern which is evident is that the SDMX-ML registry interface messages correspond to the layers of the logical SDMX-RR architecture. For each layer - the structural repository, the provisioning repository, and the data and metadata registry - there are two sets of SDMX-ML registry interface messages:

Structural Repository:

SubmitStructureRequest & SubmitStructureResponse

QueryStructureRequest & QueryStructureResponse

Provisioning Repository:

SubmitProvisioningRequest & SubmitProvisioningResponse

QueryProvisioningRequest & QueryProvisioningResponse

Data and Metadata Registry:

SubmitRegistrationRequest & SubmitRegistrationResponse

QueryRegistrationRequest & QueryRegistrationResponse

Additionally, there is a set of messages which apply to all three layers of the architecture, related to subscription/notification:

Subscription/Notification:

SubmitSubscriptionRequest & SubmitSubscriptionResponse

NotifyRegistryEvent

The third pattern is the division between “submit” and “query” messages. All “submit” messages are used to register or submit objects to the appropriate level of the architecture. Thus, structural objects are submitted to the structural repository, while data sets are registered in the data and metadata registry layer. Submit messages can be understood as those which are used for maintenance activities – adding, modifying, or deleting maintained objects.

Query messages are those which are used to discover the objects in each of the architectural layers. Again, the subscription and notification is a special case, because – although subscriptions are submitted in a normal fashion – the notifications are outside the basic model.