sdmx
Statistical Data and Metadata eXchange

SDMX STANDARDS

SUMMARY OF MAJOR CHANGES AND NEW FUNCTIONALITY IN VERSION 2.1

April 2011

sdmx

Statistical Data and Metadata eXchange

Contents

1 Structural Metadata

1.1 Data Structure Definition (DSD)

1.1.1 Terminology Changes

The term “Key Family” is discarded and the technical standards use the term “Data Structure Definition” or DSD. The term Data Structure Definition has been used as an equivalent term to Key Family for some time, and at version 2.1 this term is used in the technical standards.

The term “cross-sectional” has been discarded in favour of having a unified model that describes all data representations. Note that in the XML time series data is a particular case of the unified model.

1.1.2 Data Attribute Attachment

This has been enhanced to support better the mechanism for specifying to which construct(s) a Data Attribute can be attached. The mechanism in version 2.0 is to specify the “attachment level” which is one of: Data Set, Group, Series, Observation. The revised mechanism in version 2.1 is to specify an Attribute Relationship. Essentially, this relationship identifies the structural constructs that could trigger a change in the value of the Data Attribute if the value of that structural construct changes. This can be one of: Data Set, one or a set of Dimension(s), Observation. The essential change here is the set of Dimensions: here it is recognised that the attribute may be related to only a sub set of the Dimensions in a series key and so creating such a relationship will ensure that the attribute value in a Data Set will be reported at the appropriate place in the Data Set. This revised mechanism simplifies the placement of the Data Attribute in a data set, especially for a non-time-series-oriented data set.

The version 2.0 “cross sectional attach” attributes are eliminated in version 2.1.

1.1.3 Measure Dimension

While in version 2.0 the construct of a “Measure Dimension” is supported by a concept role which is implemented as a boolean “isMeasureDimension” in the schema, in version 2.1 there is an explicit element called MeasureDimension. This dimension references a Concept Scheme as its enumerated content (for other Dimensions a Codelist is used as an enumerated list): in other words, the value of the dimension in a key of a data set must be one the Concepts in the Concept Scheme. In version 2.0 the Dimension that is the “MeasureDimension” must have a codelist enumerating the measures and in addition all of the “measures” must be present in a Concept Scheme, and the map between the two is declared in the DSD. Therefore, the use of a Concept Scheme for the Measure Dimension greatly simplifies the DSD.

1.1.4 Concept Roles

In version 2.0 the specific role that a Data Attribute or Dimension can play in the DSD is specified as a boolean in the schema (e.g. isEntity, isFrequency, etc.). The number of requested new roles will continue to increase as new use cases are found for the SDMX standards. Therefore, at version 2.1 these Boolean XML attributes are removed and replaced with a reference to a Concept in a Concept Scheme. These Concepts identify the role. The Concept scheme is maintained at the level of the community and this allows fundamentally different communities to maintain different schemes of role Concepts. However, in order to maintain interoperability within a major community it is expected that this community will maintain and recommend the use of a single scheme for those “role” Concepts that are deemed to be important for interoperability on a community-wide basis.

1.2 Metadata Structure Definition (MSD)

1.2.1 Terminology

The naming and the method of identifying the object to which the metadata in the Metadata Set is to be attached has been modified to make this much simpler and to use more meaningful terms. The Full Target Identifier and Partial Target Identifier, in version 2.0 are replaced by the single Metadata Target and the Identifier Component is renamed Target Reference and is contained by the Metadata Target.

1.2.2 Support for more complex target objects

At version 2.0, the constructs for identifying the object to which metadata can be attached is limited to simple object types such a Code, Category etc. In version 2.1, the type of “target object” is extended to include whole or partial data or metadata keys, as well as better support for identifying a data and metadata set. Furthermore, it is possible to specify a URN as the identification mechanism for a structural object such as a Concept or Code.

1.2.3 Metadata Attribute

The functionality introduced in version 2.1 for a Metadata Attribute is:

- it is possible to specify the representation of a Metadata Attribute as XHTML.
- the Metadata Attribute can occur many times at a specific location in the Metadata Report

1.3 Constraint

In version2.0, the Constraint is embedded in the object it constrains, such as a Dataflow or Provision Agreement. Furthermore, the Constraint at version 2.0 is only available for use in a Registry context as it is only available in the Registry schemas.

In version 2.1, the Constraint is independently maintained (it is a Maintainable Artefact) and references the object or objects that it constrains. Therefore, the same Constraint can be “used” to constrain multiple objects, including multiple objects of the same type (e.g. the same constraint can be used to constrain the content of a Code List used in multiple DSDs). Importantly, the “attachment” of a Constraint to a DSD, Dataflow etc. does not affect the version of the DSD, Dataflow and can be attached even if the object is marked as final”.

Furthermore, the Constraint is structural metadata in version 2.1 and is contained in the Structure schemas.

Note that a Constraint always constrains the allowable or actual content of structures or data sets that relate to a DSD. Rules have been specified to define how

Constraints are inherited or “cascaded” when constraints are specified for one or more objects that are related to the same DSD (e.g. DSD and Dataflow).

1.4 Code List

In version 2.1, the ability to disseminate or exchange a “partial” code list is introduced. The content of the partial code list is specified on a Constraint and can be specified for any object to which a Constraint may be attached. This makes it possible to use common (and often quite large) Code Lists in multiple DSDs and then to Constrain their content for use in a specific DSD. A sub set code list can be built using the Constraint and exchanged or disseminated in a Code List marked as “partial”. This is implemented as a Boolean XML attribute on the Code List.

In addition to the use case for a partial code list it was deemed useful to allow the exchange of other partial “list” and so the version 2.1 standard allows the exchange of partial “lists” for all types of “Item Scheme”: Codelist, Concept Scheme, Category Scheme, Agency Scheme, Data Provider Scheme, Data Consumer Scheme, Organisation Unit Scheme, Reporting Taxonomy.

1.5 Organisation Scheme

The Organisation Scheme has been restructured in version 2.1 to better support the maintenance of the specific Organisation Schemes of Agency, Data Provider, and Data Consumer whilst still providing support for a standard Organisation Scheme (now called Organisation Unit Scheme). This is particularly important for ensuring that the identification of a Maintenance Agency is unique.

1.6 Categorising Structures

In version 2.0, a Category can only reference a Dataflow or Metadataflow. Furthermore, the reference is embedded in the both the Category and the reverse reference is placed in the object it is referencing. This two-way referencing is extremely brittle and the versioning rules are not clear.

In version 2.1, a Category can categorise any type of Identifiable Artefact (e.g. DSD, Process). In order to support this new object type of “Categorisation” is introduced. The Categorisation is independently maintainable and references the Category and the object that is categorised. This enables a Category to categorise many objects and objects types and any identifiable objects to be categories by many Categories. Importantly, a Categorisation has no effect on the version of the objects and can reference and a “final” object.

1.7 Process

At version 2.0, it is possible to identify the type of object that is the source or target of a Process Step but not the object itself. At version 2.1, it is possible to identify actual source and target objects (e.g. a specific DSD). At version 2.1, it is possible to give details about the software package used in the Process Step.

1.8 Provision Agreement

At version 2.0, the Id of the Provision Agreement did not form a part of its URN (which is constructed as a concatenation of the URNs of the Datafow/Metdataflow and Data Provider that is references), and the Provision Agreement is not Maintainable. This can cause maintenance and interoperability problems as different implementations will take different assumptions about which maintenance agency is responsible for the Provision Agreement. At version 2.1, the Provision Agreement is made maintainable and its Id is included in the URN. The Provision Agreement is structural metadata in version 2.1 thus enabling it to be queried and returned from a non-registry based structural metadata repository: at version 2.0, the Provision Agreement is only available in the Registry.

1.9 Transformations and Expressions

This package has been revised to better reflect the metamodel on which it is based (Common Warehouse Metamodel). It is retained in the Information Model although there is no schema implementation in either version 2.0 of version 2.1. It is anticipated that the model will be refined and an implementation will be made available in a future release of the standard.

2 Data Set

2.1 Message Changes

In version 1.0 and 2.0, there are 4 data messages, each with a distinct format. Because of the design, data in some formats could not always be related to another format. In version 2.1, this issue has been addressed by merging some formats and eliminating others. The version 2.0 data sets and their 2.1 equivalents are:

Type of data	**Version 2.0**	**Version 2.1**
Generic cross sectional	Not supported	*GenericData*
Generic time series	*Generic*	*GenericTimeSeriesData*
Structure-specific cross sectional	*Cross Sectional*	*StructureSpecificData*
Structure-specific time series	*Compact*	*StructureSpecificTimeSeriesData*

In general, there are now two types of data formats:

- GenericData
- Structure-specificData, i.e. specific to one Data Structure Definition

Both of these formats are now flexible enough to allow for data to be oriented in series with any dimension used to disambiguate the observations (as opposed to only time or a cross sectional measure in 2.0). The formats have also been expanded to allow for ungrouped observations.

To allow for applications which only understand time series data, particular cases of these formats have been introduced in the form of two data messages; *GenericTimeSeriesData* and *StructureSpecificTimeSeriesData*. It is important to note that these variations are built on the same root structure and can be processed in the same manner as the base format so that they do NOT introduce additional processing requirements. Therefore there are now 4 data messages which are based on two general formats:

- GenericData
    - o GenericTimeSeriesData
- StructureSpecificData
    - o StructureSpecificTimeSeriesData

2.2 Structure-specific Data Mechanism Revised

Since version 1.0, the (data) structure specific data formats (namely compact and cross sectional) were derived in such a manner that made them difficult to process in a reliable manner, and even more difficult to validate that the derived schemas conformed to the standard. This resulted in data being published in these formats that appeared valid, but were not properly structured. In 2.1, this issue has been addressed by revising the base structure-specific data schemas and the mechanisms used to derive the data structure definition specific schemas and instances so that better defined and consistent data messages could be created. In addition, this mechanism allowed for the base format to be described in a generic manner which make understanding the structure of the data messages clear. Finally, these changes allow for the data messages to move away from a time series centric format, yet still allowing for time series only formats with very little additional processing overhead.

The rules for specification for creating the structure specific schemas has been revised to allow for more variation in the mechanisms used to create the structure specific schemas. As opposed to stating exact rules for the naming and creating of every schema type and element, the new specification allows for flexibility where appropriate, while at the same time being specific as to what the allowable content should be. The result is a consistent end result that recognizes the means to achieving the structure might vary.

3 Metadata Set

3.1 Message Changes

The names of the reference metadata messages have been revised to be more consistent with the data message. The version 2.0 data sets and their 2.1 equivalents are:

Type of Metadata Set	Version 2.0	Version 2.1
Generic Metadata Set	*GenericMetadata*	*GenericMetadata*
Structure-specific Metadata Set	*MetadataReport*	*StructureSpecificMetadata*

3.2 Alignment of Formats

In version 2.0, the generic metadata report message was structured much differently from the metadata structure specific format and the terminology used in the formats was not consistent. This has been changed in version 2.1. The format of the generic (*GenericMetadata*) and structured metadata (*StructureSpecificMetadata*) sets are nearly identical, with the same names being used for each level of the metadata set structure.

3.3 Structure-Metadata Mechanism Revised

In version 2.0, the mechanism for deriving the metadata specific schemas did not allow for a metadata attribute to have different content within different portions of a report structure; the metadata structure definition made no such restriction. In 2.1 the mechanisms for defining the base structured metadata schemas and for deriving the metadata structure specific schemas and instances have been revised to solve this issues and to create a more consistent message structure, which is much simpler to process in a generic manner.

In the same manner that the derivation of the data structure specific schemas has changed, so has the derivation of the metadata structure specific schemas. The specification no longer dictates particular methods of schema generation where they are not necessary. Instead, the focus is on creating structures with appropriate content model, regardless of the means in which this model is derived.

4 Web Services

4.1 RESTful interface

Support for the RESTful interface has been added to the web services specification in version 2.1. This support applies to data, reference metadata, and structural metadata. In addition to SDMX web services, the support for this interface has also been included in the SDMX Registry functionality.

4.2 Revised Query

In SDMX 2.0, the query messages were inconsistent and allowed for non-sensible queries. In version 2.1, these have been refined so the structure of the query closely follows the information model, resulting in consistent query messages which are applicable to the object being queried. In addition, specific messages have been added for each type of query (e.g. a specific DataQuery message, a specific DataStructureQuery message). These specific messages allows of a web service function to more accurately define what the acceptable input message is.

4.3 Error Codes

Version 2.1 includes a list of standard error codes to be used by queryable data, metadata, and structural metadata services.

4.4 WSDL and WADL

Version 2.1 added a formal WSDL (Web Services Description Language) for defining SOAP based web services and a formal WADL (Web Application Description Language) for defining RESTful web services.

5 Registry

5.1 Subscription/Notification

At version 2.1 the subscription includes the identification of an organisation that is responsible for the subscription. This will enable a registry service to support subscription maintenance whilst not enforcing a subscriber to be a maintenance agency.

5.2 Revised Query

At version 2.0 the Registry has a different query schema from that used by non-registry based structural metadata repository. At version 2.1 the same query schema is used, and this is the schema for the Structure Where construct of the query message.

5.3 Registration

The Registration is made identifiable at version 2.1, thus enabling it to be maintained within a Provision Agreement. Registrations can only be made for Provision Agreements at version 2.1: in version 2.0 the Registration could be attached to a Dataflow and Metadataflow) but this was not implementable as there is no reference to the Data Provider (i.e. the publisher of the data or metadata).

5.4 Refinement of the Meaning of "final" for Structural Metadata

In version 2.0 the implication is that if a Maintainable Artefact, such as a Code List or a Data Structure Definition, is marked as "final" its contents cannot be changed unless a new version is created. Strict adherence to this rule makes it impossible, for instance, to add a "validTo" date once a "final" version is replaced by a new version. A review of the constructs was undertaken and certain constructs are allowed to be amended, such as validFrom, validTo, textual constructs such as Name and Description. The responsibility for deciding whether the change represents a version change is given to the Maintenance Agency. However, a general guideline is suggested that any documentation change which alters the meaning of an object should be treated as new version. For example, changing the name of a code to something entirely different should result in a new version of the parent code list, whereas correcting a typographical error in the name could be treated as a non-versioning change (assuming the corrected error does not drastically alter the meaning of the code). Note that it is not possible to insert or delete contained or referenced objects such as a code in a code list or a Dimension in a DSD. The list of constructs that can be amended when the construct is marked as "final" is specified in part 5 - the SDMX Registry specification.Other Topics

5.5 Time Periods

The rules and usage of the time period and time formats have been revised to ensure that time values are processed in a consistent manner and can meet a wider range of requirements.

There are essentially 4 categories of Time Periods:

1. Gregorian Periods: standard ISO 8601 date formats (i.e. year, year and month, and date).

2. Standard Reporting Periods: (these are based on predefined portions (year, semester, trimester, quarter, month, week, and day) of a reporting year, which can start on any given calendar day). The start day of the reporting year is specified in a special attribute with a fixed identifier of "REPORTING_YEAR_START_DAY" and a fixed representation of a month and day.

3. Distinct Time Ranges: based on the ISO 8601 Time Interval, expressed as a start date or date time and a duration

4. Distinct point in time (based on ISO 8601 date-time)

There is no ambiguity in these formats so that for any given value of time, the category of the period (and thus the intended time period range) is always clear. It should also be noted that by utilizing the ISO 8601 format, and a format loosely based on it for the report periods, the values of time can easily be sorted chronologically without additional parsing. A detailed description of the formats and how to determine the exact date range covered by any of these time periods, as well as how they should be treated in a query message is contained in Part 06 of the standards - “Implementers Notes”.

The time format attribute is still possible in SDMX-ML, this is no longer required, even for compatibility with SDMX-EDI. The format of time is unambiguously apparent from the value itself. Part 06 ("Implementers Notes") of the standard provides documentation on how the time format might be represented and how to convert time period value in SDMX-ML to and from SDMX-EDI.

5.6 Time Zones

The specification and default handling of time zones has been specified in Part 06 ("Implementers Notes") of the standard. A time zone offset is available on any representation of a time period and a default value can be provided at the message sender level. If a time zone offset is not provided, the standard is now clear in how time periods should be interpreted.

5.7 Header

The elements in the Header have been extended to support the identification of structural metadata that is important in the processing of the message.

The table of elements and on which messages they are used is shown below.

Field	Structure	Generic Data & Generic Time Series Data	Structure Specific Data & Structure Specific Time Series Data	Generic Metadata	Structure Specific Metadata	Registry	Message Group
ID	1	1	1	1	1	1	1
Test	1	1	1	1	1	1	1
~~Truncated~~							
Name	*	*	*	*	*	*	*
Sender	1	1	1	1	1	1	1
Receiver	*	*	*	*	*	1	*
Structure	0	?	1	?	1	0	+
DataProvider	0	?	?	?	?	0	?
DataSetAction	0	?	?	?	?	0	?
Extracted	0	?	?	?	?	0	?
ReportBegin	0	?	?	0	0	0	?
ReportEnd	0	?	?	0	0	0	?
EmbargoDate	0	?	?	0	0	0	?
Source	*	*	*	*	*	*	*

Where:

1 = 1 occurrence (mandatory)

*   = zero to many occurrences

? = zero or 1 occurrence

+ = 1 to many occurrences

Note that the Truncated field in the Header has been deemed not only unnecessary, but also unusable. A message would be truncated during a streaming process when a limitation on the output has been reached. In such a scenario, it would not be possible to set a Header field. This information will be conveyed in a message footer section which will contain error/warning/information after the payload of the message.

6 Technical Changes to the SDMX-ML Schemas

Major changes have been made to the structure of the SDMX-ML schemas in order to align them more closely to the Information Model and, in particular, to make them more object oriented to aid software implementation. In order to ease the maintenance effort and readability of the schemas, they have been broken into module files at appropriate levels. Each namespace contains a root schema which includes the necessary modules, so that there is still a single point of entry for any given namespace.

Note that whilst the internal structure of the schemas have changed, the actual xml documents are very similar in structure to the version 2.0 format unless there have been changes in the functionality, such as in the re-design of the Metadata Structure Definition.