SDMX Standards: Section 3A

SDMX-ML:

Schema and Documentation

Version 2.1

April 2011

© SDMX 2011

http://www.sdmx.org/

Contents

[1 BACKGROUND 1](#_Toc290377474)

[1.1 History and Version 2.0/2.1 Developments 1](#_Toc290377475)

[1.2 The XML Design 1](#_Toc290377476)

[1.3 Fostering the Use of a Standard SDMX-ML 2](#_Toc290377477)

[2 NORMATIVE REFERENCES 2](#_Toc290377478)

[3 CONFORMANCE 2](#_Toc290377479)

[4 DESIGN OVERVIEW 2](#_Toc290377480)

[4.1 Scope and Requirements 2](#_Toc290377481)

[4.2 Design Approach 4](#_Toc290377482)

[4.3 SDMX-ML Packaging: Namespace Modules 5](#_Toc290377483)

[5 RELTATED DOCUMENTS 7](#_Toc290377484)

# BACKGROUND

## History and Version 2.0/2.1 Developments

The SDMX Technical Standards Version 1.0 established an information model which described aggregated statistical data sets and the structural metadata needed to exchange them in a standard fashion. This drew on the earlier example of the GESMES/TS standard. Based on the SDMX information model, several formats were developed: XML formats for exchange of structural metadata, data sets, and queries for these (SDMX-ML), and EDIFACT formats for the structural metadata and data sets (SDMX-EDI). These standards supported a number of exchange patterns, characterized as "bilateral", "gateway", and "data-sharing" models, as described in the Framework document in the Version 1.0 standards package.

Versions 2.0 and 2.1 build on this foundation to provide a higher degree of support for all of these models, with an emphasis on data sharing in the form of a set of standard registry services interfaces. It has also expanded to include support for new types of metadata exchange and reporting, with a focus on "reference metadata" concerned with quality, methodology, and other issues. Further, the ability to provide metadata about the relationships between data sets and structures has been expanded, providing more support for data cubes. Finally, experience has shown that some minor additions to the existing structural metadata and dataset structures will allow SDMX to support more different types of statistical information.

The scope of the Version 2.1 SDMX Technical Standards is thus much broader, and is accompanied by a larger set of message types in the SDMX-ML formats. While the XML formats described here have grown in number and scope, the EDI formats remain relatively unchanged.

## The XML Design

All of these document types will share a common "envelope" at the message level ("SDMXMessage.xsd"), as well as a set of common low-level components (“SDMXCommon[subset].xsd”) so that header information and basic structure of a message will always be the same.

* Schema and sub-modules for describing all types of structural metadata – for data sets (data structure definitions), for metadata sets (metadata structure definitions), for related groups of metadata and data structures, and for all types of structural objects involved in registry-based exchanges ("SDMXStructure[*subset*].xsd")
* Generic data schema for data-sharing exchange ("SDMXDataGeneric[*subset*].xsd")
* Structure specific data schema for for updates, revisions/bilateral exchange ("SDMXDataStructured[*subset*].xsd")
* Generic query schema and sub-modules for invoking web services ("SDMXQuery[*subset*].xsd")
* Generic schema and sub-modules for registry interfaces (“SDMXRegistry [*subset*].xsd”)
* Generic schema for reference metadata sets (“SDMXMetadataGeneric.xsd”)
* Metadata-structure-definition-specific schema for metadata sets (“SDMXMetadataStructured.xsd”)

## Fostering the Use of a Standard SDMX-ML

In addition to these different formats, standard mappings and corresponding transformation tools have been developed for the creation of data structure-specific schemas from structure descriptions, to transform XML data instances from one XML data description format to another, and from these formats into the corresponding SDMX-ML messages. This level of free tools support will foster the early use of SDMX and permit the data to be easily used across all processes, which is otherwise a difficult requirement to meet. Ultimately, it is the fact that all formats share a common information model that enables this approach to meet the wide set of SDMX requirements.

# NORMATIVE REFERENCES

W3C XML Schema Definition Language, version 1.0 (URL: http://www.w3c.org/XML/Schema#dev), World Wide Web Consortium

W3C Extensible Markup Language, version 1.0, Third Edition (URL: http://www.w3c.org/TR/2004/REC-xml-20040204/), World Wide Web Consortium

# CONFORMANCE

Part IV (SDMX\_2\_1\_SECTION\_03A\_PART\_IV\_DATA) of this document is normative, providing rules for the creation of conformant SDMX-ML XML instances and W3C XML Schemas.

# DESIGN OVERVIEW

## Scope and Requirements

To understand the relationships between the several document types, it is important to have some familiarity with the requirements they are designed to fulfil.

* Large amounts of data must be captured in a reasonably compact format, because of the potential size of databases being exchanged.
* It must be possible to send incremental updates, rather than entire, complete databases. The validation of such exchanges demands not that an entire data set be exchanged, but only that enough information be sent to ensure accurate updating and revision processes.
* Structural information as well as data will need to be transmitted.
* There must be a reliable transformation to and from the GESMES/TS EDIFACT syntax.
* It should be possible to present natural-language information in multiple, equivalent languages.
* To support web services and similar technological approaches, there is a requirement to send queries to information sources as well as data and structures.
* Users (and registry services) may not know about a specific data structure, and will need to be able to handle data across data structures, and even (for, say, a comparison service) to put data structured according to multiple data structures in a single XML instance.
* The XML should conform to the Information Model as closely as possible, and promote simple mapping from the XML instances to model based objects
* The XML should behave as “normally” as possible within standard XML tools such as web development environments, parsers, guided editing tools, etc.
* Data should be able to be structured in one or two levels, with the two level format allowing for a single dimension which serves to disambitguate the observations. In addtion, it should still be possible to restrict this more generalized structure to time series only formats. However, doing so should not add any additional burden in terms of how data is processed.
* Data structure and metadata structure specific formats should be able to be generically processed, without having to understand the entire structure; which is to say that the structure-specific schemas should serve to validate the information, but not be required to process it.
* XML formats should promote re-use of common semantics, concepts, and codelists to the greatest possible extent, while still recognizing the agency which maintains a specific resource (a codelist, a data structure, a data set, etc.)
* XML formats must support interactions of applications with standard registry services, based on standard interfaces. These must function both as web services, and as services operating over http and similar protocols.
* XML formats must support the reporting of reference metadata which is not structural in nature, but which constitutes a primary information flow of metadata attached to other parts of the statistical collection, reporting, processing, exchange, and dissemination. Quality initiatives, methodological metadata, administrative metadata, and similar types of metadata reporting must be supported, and must be user-configurable.
* XML formats for describing the relationships between groups of metadata sets and data sets, by mapping concepts and codelists between these structures, and by allowing for common querying of data and metadata described with not only a single structural definition, but with a related set of structural definitions, based on these mappings.
* Allow for time-related concepts which are not related to the time of the observation to be used in data structures.
* Allow for simple, un-coded incremental identifiers in data structure definitions, to be used to dis-ambiguate data series/observations which do not have a simple 1-to-1 relationship with the time period of the observation.
* Allow for un-coded identifiers and descriptors to be associated with data structure definitions which establish an external entity or identifier to disambiguate between otherwise identical series/observations (ie, when a data set describes a group of organisations, or a set of accounts, which might otherwise have identical key values).
* Allow for non-numeric observation values (usually but not always coded)
* Allow “cube”-based systems (such as OLAP) to interoperate with less sophisticated systems, without necessarily losing the richness of metadata found in the more sophisticated systems.

This is a very broad set of requirements, and in examining these it becomes evident that some of the requirements are very much at cross-purposes. It is almost impossible to design a single XML document type for any single function (exchange of data, exchange of reference metadata, querying, etc.) which will satisfy all of these requirements. At the same time, it was very much felt that whatever design was adopted should have a clear relationship with the information model.

## Design Approach

The basic design for the XML Schema started with the information model. Just as the model does, the structural metadata objects are built up from the abstract classes on which they are based. This approach means that more advanced XML tools that understand such inheritance can use thes constructs to more easily deserialize the XML instances into object based on the models.

Another fundamental approach was to make constructs as consistent as possible so that tools built to process the XML could make use of reuse. This approach extends not only to the basic information model properties such as identificaitons and version, but also to areas such as referencing and query messages.

Through the years of the standard, it was identified that users typically implemented on the previously named CompactData message and/or the GenericData message. It was determined that the UtilityData message was rarely used,and the CrossSectionalData message was used, but often in inconsistent manners due to the ambiguity in its definition. Therefore, the apporoach for this verison was to harmonize these data formats as much as possible. The result is two basic data formats; a generic format and a data structure definition specific format. As much as possible these formats have been structured so that they are very similar in structure. In addition to the harmonisation of the generic and structure specific data formats, the requirements of the CompactData and CrossSectionalData messages have been combined into one format. This format is now flexible in that it allows for any single dimension to exist at the observation. In order to accomodate existing applications that did only use the CompactData message due to its time series orientation, time series only variations of the structure specific and generic data messages have been created. Care was taken to ensure that these time series only variations could be processed just as the general format counterparts are so as to introduce no additional requirements.

These same principles were applied to the reference metadata messages as well. In addition, the base structure specific schemas from which data and metadata structure specific variations are created, have been structured so that they enforce a specific structure and can be easily processed in a generic mannner.

Another new approach that has been introduced in this version is to remove some of the generalities from the messages that existed to allow for web services to be more specific as to the contract of their functions. As opposed to have a single query message that can accomodate any structural metadata query and data and reference metadata queries serving as the input into a codelist query function­­–a specific message is created that only allows for what a codelist query should reasonably be expected to handle.

## SDMX-ML Packaging: Namespace Modules

In the proposed XML Schema design, there is a packaging scheme based on the idea that XML namespaces can be used as “modules”, so that any given user or application need only be familiar with a subset of the entire library in order to use it. This approach fit very well with the design described above, and is often used in major XML standards for other domains.

The other major benefit of namespaces – especially in light of the requirement that maintenance agencies be tracked across the potential reuse of the structures and data they maintained – is that it allows SDMX to own certain namespace modules, and allows other maintenance agencies to own namespaces specific to the key-families or metadata structure definitions they also maintain.

The result is a set of namespace packages which agree with the design approach described above. Some large schemas are divided into sub-modules, all within the same namespace, for ease of use. Each module or sub-module is a single instance of the W3C XML Schema Language’s schema element, associated with the apropriate XML namespace. Where these modules and sub-modules have dependencies on one another, they use the XML Schema importing mechanism to draw on constructs described in another module or sub-module. Further, the namespces themselves reflect a sort of relationsip. For example, all data constructs are in a namespace based on http://www.sdmx.org/resources/sdmxml/schemas/v2\_1/data. These relationships in the namespace are meant to reflect the relationship of the constructs.

* An SDMX Namespace Module containing the common message constructs, including the common header information (“SDMXMessage.xsd”) - uses all other SDMX-ML namespace modules
* An SDMX Namespace Module containing the message footer constructs, including the common header information (“SDMXMessageFooter.xsd”) - used only by the message namespace.
* An SDMX Namespace Module containing the descriptions of structural metadata such as key families, concepts, and codelists (“SDMXStructure [*sub-module*].xsd”)
* An SDMX Namespace Module containing constructs shared in common across all of the SDMX message types (“SDMXCommon[*sub-module*].xsd”) – needed for all other SDMX-ML namespace modules (also included for convenience is the XML namespace [“xml.xsd”] provided by the W3C for including the xml:lang attribute in schemas).
* An SDMX Namespace Module describing the generic (non-data structure-specific) format for formatting data (“SDMXDataGeneric.xsd”)
* An SDMX Namespace Module for structured (data structure-specific) data (“SDMXDataStructured.xsd”)
* An SDMX Namespace Module for describing the structure of the generic query message (“SDMXQuery[*sub-module*].xsd”) – for web services developers and users, etc.
* A set of namespaced modules created and maintained by those who create data structure-specific schemas – not maintained by SDMX
* An SDMX Namespace Module providing a generic format for reporting of reference metadata, regardless of metadata structure definition (“SDMXMetadataGeneric.xsd”).
* An SDMX Namespace Module providing the common framework to be used for all metadata-structure-specific schemas for reference metadata reporting (“SDMXMetadataStructured.xsd”).
* A set of namespaced modules created and maintained by those who create metadata-structure-specific “Metadata Report” schemas – not maintained by SDMX.
* An SDMX Namespace Module providing standard interfaces for interactions with a set of registry services (“SDMXRegistry[*sub-module*].xsd”).

The following sections describe in detail the proposed XML formats, which should be examined alongside the documentation provided. These proposed schemas are divided into the generic schemas, for which a complete set of schema definitions can be provided, and data structure-specific schemas, for which a core structure is provided (with schema code), plus a guide to how a specific data structure or metadata structure definition can be mapped onto the core structure.

# RELTATED DOCUMENTS

To all for more managable doucments, the remainder of this document has been divided into seven different parts, each a separate document. These parts are as follows:

1. SDMX\_2\_1\_SECTION\_03A\_PART\_I\_MESSAGE: This contains the description of the message namespaces schemas
2. SDMX\_2\_1\_SECTION\_03A\_PART\_II\_COMMON: This details the common namespace schemas
3. SDMX\_2\_1\_SECTION\_03A\_PART\_III\_STRUCTURE: This details the structure namespace (structural metadata) schemas
4. SDMX\_2\_1\_SECTION\_03A\_PART\_IV\_DATA: This details the data and metadata namespaces, as well as detailing the rules for creating strucutre specific schemas
5. SDMX\_2\_1\_SECTION\_03A\_PART\_V\_QUERY: This details the query namespace schemas
6. SDMX\_2\_1\_SECTION\_03A\_PART\_VI\_REGISTRY: This details the registry namespace schemas
7. SDMX\_2\_1\_SECTION\_03A\_PART\_VII\_SAMPLES: This provides a brief explanation of the currently available sample files.