SDMX Standards: Section 3A PaRT VI

SDMX-ML:

Schema and Documentation

Samples

Version 3.0

September 2021

© SDMX 2021

http://www.sdmx.org/

Contents

[1 Introduction 1](#_Toc77174570)

[2 Structure Samples 1](#_Toc77174571)

[3 Data Samples 5](#_Toc77174572)

# Introduction

This document provides a brief overview of the SDMX-ML XML sample files.

All samples use the xsi:schemaLocation attribute to define the location of the XSD schema files for the relevant namespaces. The samples assume that the schemas are held locally in a directory called ‘schemas’ and are referenced as a relative file path. For example:

xsi:schemaLocation="http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/message ../../schemas/SDMXMessage.xsd"

Copies of the SDMX-ML XSD schemas files will need to be in the appropriate directory in order to validate correctly.

# Structure Samples

The Structure Samples provide examples of SDMX-ML messages within the *structure* namespace which is used for all structural metadata artefacts.

**Codelist**

codelist.xml A simple enumerated code list directly equivalent to those in version 2.1 and earlier.

This example shows the SDMX:CL\_AGE ‘Age’ cross domain code list.

codelist – extended.xml Extended code list.

Illustration of the feature introduced in SDMX 3.0 allowing code lists to be defined as an extension of others. One use case envisaged is adding additional codes to standard cross domain code lists without needing to take and maintain an explicit copy.

The sample message has two codelists:

* SDMX:CL\_AGE, a simple enumerated codelist; and
* EXAMPLE:CL\_EXTENDED\_AGE, an extension of SDMX:CL\_AGE.

CL\_EXTENDED\_AGE defines two additional codes “I” and “S”.

In addition, the ExclusiveCodeSelection option to exclude the “Y” code from the CL\_AGE code list. The result is CL\_AGE, minus “Y”, plus “I” and “S”. The InclusiveCodeSelection option could have been used if a specific set of codes were needed from CL\_AGE.

codelist – discriminated union.xml Discriminated union of code lists.

This sample illustrates the discriminated union of code lists use case where a classification or breakdown has multiple “variants” which are all valid or mutually exclusive.

Economic activity has been used with ISIC4 and NACE2 examples of two mutually exclusive variants.

In the sample:

* Code list extension is used to create a new EXAMPLE:CL\_ACTIVITY code list consisting of both the SDMX:CL\_ACTIVITY\_NACE2 and SDMX:CL\_ACTIVITY\_ISIC4 codelists.
* The code list extension prefix feature has been used to prefix the NACE codes with “NACE2\_”, and similarly the ISIC codes with “ISIC4\_”. This ensures there is no ambiguity where the same code appear in both NACE and ISIC code lists.
* A data structure definition has been created with an enumerated ACTIVITY dimension using the EXAMPLE:CL\_ACTIVITY code list.
* Two data flows are created referencing the data structure definition. One data flow has a data constraint attached with a CubeRegion for ACTIVITY and Value = “NACE2\_%”. The other has a similar data constraint with Value = “ISIC4\_%”. The “%” is the wildcard character for constraints introduced in version 3.0.

The discriminated unions are achieved by requesting either of the data flows with references=”all” and detail=”referencepartial”. The result being CL\_ACTIVITY with the extensions resolved and the relevant data constraint applied. Thus CL\_ACTIVITY will only contain codes prefixed according to the data flow: either beginning “NACE2\_” or “ISIC4\_”

valuelist.xml A simple enumerated value list.

Value lists were introduced in version 3.0 to allow the definition of enumerations where the codes do not need to comply with the strict SDMX rules for identifiers. Thus a “value” can be any string of characters.

In SDMX-ML valuelists are treated as a class of enumeration so appear in the structure message under Codelists alongside simple and extended codelists, and specialised geospatial codelist variants.

Currency is used as the example with the values being currency symbols including “$”, “£”, “¥” and “﷼”.

**Concept Scheme**

conceptscheme.xml The example illustrates a single concept scheme ECB:ECB\_CONCEPTS containing multiple individual concepts.

**Data Structure Definition**

datastructuredefinition.xml The example illustrates a data structure definition (DSD) for ECB:EXR exchange rates. The concept of primary measure has been deprecated in SDMX 3.0 and the DSD’s MeasureList can contain multiple measures. In this case, a single measure OBS\_VALUE is defined.

**Dataflow**

dataflow.xml A simple data flow for the ECB:EXR data set.

**Geospatial**

geospatial geocomponents.xml An example of the use of the “GeospatialInformation” representation type. The GeospatialInformation type can be assigned to a Dimension, DataAttribute, MetadataAttribute or a Measure in conjunction with the "GEO" role. Values of are included directly in data messages or metadata reports and must conform to the Geo Feature Set syntax defined in Section 6 of the Technical Specifications.

In the sample:

* A simple illustrative data structure definition has been defined with IDENTIFIER and TIME\_PERIOD dimensions, plus a series attribute AREA.
* The AREA attribute has the SDMX Concept Role “GEO” defined identifying it generally as a geospatial component following guidelines.
* The AREA attribute also carries a LocalRepresentation with textType=”GeospatialInformation”.

geospatial geographiccodelist.xml An example of the definition of GeographicCodelists, a specialised form of code list introduced in SDMX 3.0. GeographicCodelists are used in the same was as simple code lists to assign a closed enumerated list of values to a component. The difference being that the definition of each GeoFeatureSetCode explicitly specifies the geospatial features to which it relates as a string expression using the Geo Feature Set syntax set out in Section 6 of the Technical Specifications. As with any other coded component, data sets only contain the codes and not the Feature Set information itself. Thus, GeoFeatureSetCodes act as pointers to the Geo Feature Set information held in the GeographicCodelist.

In the sample:

* A single illustrative GeographicCodelist has been defined which must be under Codelists in the structure message.
* The GeographicCodelist geoType attribute must be given the value of “GeographicCodelist”.
* Three illustrative GeoFeatureSetCodes have been defined with IDs “A1”, “A2” and “A3”.
* The GeoFeatureSetCode Value attribute specifies the geospatial features to which the code relates as a string expression. Note that the values shown are conceptual examples and are not syntactically valid Geo Feature Set expressions.

geospatial geogridcodelist.xml An example of the definition of GeoGridCodelists, the second specialised form of codelist for modelling gridded geographies.   
Like GeographicCodelists, GeoGridCodelists act in the same way as simple code lists to assign a closed enumerated list of values to a component.  
GeoGridCodelists define a geographical grid by setting the X and Y coordinates of a reference point, and the dimensions of each cell. In addition, each GeoGridCode specifies to which cell in the grid it relates using row and column numbers.

In the sample:

* A single illustrative GeoGridCodelist has been defined which must be under Codelists in the structure message.
* The GridDefinition element defines the grid for the code list using a string expression conforming to the Geo Grid syntax set out in Section 6 of the Technical Specifications.
* Each GeoGridCode has an ID and a Name in common with the codes of simple code lists, but also has a GeoCell element containing the row and column of the grid cell as a string expression using the Grid Cell syntax set out in Section 6 of the Technical Specifications.

**VTL Transformations**

VTL Sample 1.xml Illustrates the use of VTL Rulesets to validate an SDMX dataflow. The validation process adds additional identifiers and measures to the resulting VTL dataset indicating, on a row by row basis the validation rule applied and the outcome in terms of a valid / invalid. If invalid, details of the error found and its severity are added.

A second transformation takes the resulting dataset filtering out high-severity validation errors and persists the result back to a SDMX dataflow using VTL mappings.

VTL Sample 2.xml Illustrates aggregation of an SDMX dataflow using VTL hierarchical rulesets, filtering and user defined operators.

After the initial aggregation, unaggregated observations removed are removed from the result dataset. A user defined operator is subsequently applied to calculate a percentage measure with the final persistent result mapped from VTL to a SDMX dataflow.

VTL Sample 3.xml Illustrates the calculation of a GDP per capita dataset from an input mapped SDMX dataflow containing GDP and population indicators on a country by country basis.

# Data Samples

SDMX-ML 3.0 has a single format for data transmission - the Structure Specific Data message. Alternative formats from version 2.1 and earlier such as Generic, Utility and Cross Sectional are deprecated.

The Structure Specific Data message is characterised by the XML elements and attributes being derived from the data structure definition of the data set. This provides two main benefits:

1. The message content is relatively compact.
2. It is possible to use an XML schema to validate that the data set contains the correct values as defined by the data structure definition and any constraints attached to the dataflow, data structure definition or data provision agreement.

The samples therefore include both the Structure Specific Data XML, and a corresponding example validation schema XSD. Successful schema validation confirms that the XML is both a valid Structure Specific Data message, and the content is valid according to the SDMX structural metadata.

**Aggregated Time Series**

ECB EXR.xml Structure specific data message.  
  
Excerpt from the ECB’s Exchange Rates dataset.

In this example which is representative of aggregated time-series datasets with a single measure, observations are grouped together under series, which in turn are grouped under the dataset element.

The dimension values and those of series-level attributes are expressed as XML attributes on each series element. Similarly, the TIME\_PERIOD, OBS\_VALUE and any observation-level attributes are expressed as XML attributes on each obs element.

Note that a structure specific namespace ns1 is defined using xsi:schemalocation to reference the validation schema location, in this case ECB\_EXR\_Dataflow.xsd.

ECB EXR Dataflow.xsd Structure specific schema for validating the dataset.

The schema defines a complex type called DataSetType derived from the data set’s data structure definition and constraints, in this case those attached to the dataflow.

**Aggregated Time Series with Complex Attributes**

Complex attributes include those with multilingual text and arrays of values.

EXB EXR CA.xml Structure specific data message.  
  
A modification of the ECB’s Exchange Rates dataset illustrating the following complex attribute use cases:

TITLE – multi-lingual series-level array attribute

SOURCE\_AGENCY – coded series-level array attribute specifying a minimum of three values

SOURCE\_PUB – uncoded string series-level array attribute with maximum length of 350 characters

OBS\_STATUS – coded observation-level array attribute

Note that a structure specific namespace ns1 is defined using xsi:schemalocation to reference the validation schema location, in this case ECB\_EXR\_Dataflow.xsd.

ECB EXR CA Dataflow.xsd Structure specific schema for validating the dataset.

The schema defines a complex type called DataSetType derived from the data set’s data structure definition and constraints, in this case those attached to the dataflow.

Complex types based on the abstract dsd:CompType are defined for each complex attribute.

ECB EXR CA DSD.xml Structure message describing the Data Structure Definition for the above dataflow illustrating, in particular, the definition of complex attributes:

The XML attribute isMultiLingual=”true” on the <str:TextFormat…> element specifies a multi-lingual string representation.

Setting the maxOccurs XML attribute to a number greater than 1 (or unbounded) on the <str:LocalRepresentation…> element specifies an array.