SDMX STANDARDS

SUMMARY OF MAJOR CHANGES AND NEW FUNCTIONALITY

Version 3.0

October 2021

Revision History

Revision	Date	Contents
DRAFT 1.0	May 2021	Draft release updated for SDMX 3.0 for public consultation
1.0	October 2021	Public Release for SDMX 3.0

Contents

1 Overview 4
2 Summary of Breaking Changes in 3.0 6
2.1 Web Services API 6
2.2 Transmission Formats 7
2.3 Information Model 8
3 Information Model 10
3.1 Version 3.0 Information Model 10
3.2 Key Changes from Version 2.1 11
3.3 Areas Unchanged from Version 2.1 12
3.4 Reference Metadata 13
3.5 Microdata Exchange 14
3.6 Geospatial Data Exchange 15
3.7 Structure Mapping 16
3.8 Constraints 17
3.9 Code List Extension 17
3.10 Discriminated Union of Code Lists 17
3.11 Code Hierarchies 17
4 Versioning of Structural Metadata Artefacts 19
5 REST Web Services API 20
5.1 Simplified list of resources 20
5.2 Improved data queries 20
5.3 Improved reference metadata queries 21
5.4 Structural metadata maintenance 21
6 XML, JSON, CSV and EDI Transmission formats 22
6.1 SDMX-ML 22
6.2 SDMX-JSON 24
6.3 SDMX-CSV 24
6.4 EDI deprecation 25
Appendix A – Version 2.1 Information Model 26

1 Overview

SDMX version 3.0 introduces new features, improvements and changes to the Standard in the following key areas:

Information Model

- Simplification and improvement of the reference metadata model
- Support for microdata
- Support for geospatial data
- Support for code list extension and discriminated union of code lists
- Improvements to structure mapping
- Improvements to code hierarchies for data discovery
- Improvements to constraints

Versioning of Structural Metadata Artefacts

- Adoption of the three-number semantic versioning standard for structural metadata artefacts (https://semver.org)

REST Web Services Application Programming Interface (API)

- Change to a single 'structure' resource for structure queries simplifying the REST API specification by reducing the number of resources to five
- Improvements to data queries
- Improvements to reference metadata queries
- Support for structural metadata maintenance using HTTP PUT, POST and DELETE verbs

SOAP Web Services API

- The SOAP web services API has been deprecated with version 3.0 standardising on REST

XML, JSON, CSV and EDI Transmission formats

- The SDMX-ML, SDMX-JSON and SDMX-CSV specifications have been extended and modified where needed to support the new features and changes such as reference metadata and microdata
- Obsolete SDMX-ML data message variants including Generic, Compact, Utility and Cross-sectional have been deprecated standardising on Structure Specific Data as the sole XML format for data exchange
- The SDMX-EDI transmission format for structures and data has been deprecated
- The organisation of structures into 'collections' in SDMX-ML and SDMX-JSON structure messages has been flattened and simplified

- The option to reference structures in SDMX-ML and SDMX-JSON messages using Agency, ID and Version has been deprecated with URN now exclusively used for all non-local referencing purpose

Breaking Changes

Many of the changes made are ‘breaking’ meaning that, while conversion between versions may be possible in certain circumstances, the 3.0 specification is not directly backwardly compatible with earlier versions of the Standard.

A summary of the main breaking changes is given in chapter 2.

Content of the Document

The remainder of the document provides a summary of the main changes. More detailed information can be found the SDMX 3.0 Technical Specifications, in particular:

- Section 2 – Information Model
- Section 5 – Registry Specification
- Section 6 – Technical Notes
- SDMX-TWG GitHub for the REST API and the XML, JSON and CSV formats

2 Summary of Breaking Changes in 3.0

Version 3.0 introduces breaking changes into the web services API, transmission formats and information model. A summary is given in the table below.

2.1 Web Services API

REST API	The REST API is not backwardly compatible due to modifications to the URLs and query parameters resulting in breaking changes in four of the five main resources: • Structure queries • Data queries • Metadata queries • Availability queries Schema queries are backwardly compatible. *Guidance for implementors* REST API implementors may provide partial backward compatibility by using web server URL rewriting rules to translate version 2.1 structure queries to the 3.0 equivalent. Implementors are also recommended to version their API services providing users with an explicit choice of which version to use.
SOAP API	The SOAP API has been deprecated.

2.2 Transmission Formats

SDMX-ML	The following legacy XML data messages have been deprecated: SDMX-ML 1.0/2.0 Generic (time-series) data message SDMX-ML 1.0/2.0 Compact (time-series) data message SDMX-ML 1.0/2.0 Utility (time-series) data message SDMX-ML 1.0/2.0 Cross-Sectional data message SDMX-ML 2.1 Generic data messages (for observations, time-series and cross-sectional data) Structure Specific is the only data message option in version 3.0 but is not backwardly compatible with version 2.1 due to several changes including deprecation of the option to reference structures like the DSD, Dataflow and Provision Agreement using their Agency, ID and Version. The time series variant of the message has also been deprecated. The SDMX-ML structure message is not backwardly compatible primarily due to: • Changes to the information model • Changes to the way the structures are organised into 'collections' within the message • Deprecation of the Agency, ID, Version option for referencing of structures in messages
SDMX-JSON	The JSON data message is not backwardly compatible with version 2.1 primarily due to changes needed to support the improved REST API data queries, in particular the ability to retrieve in one operation data from multiple datasets with potentially different Data Structure Definitions. The JSON structure message is not backwardly compatible primarily due to: • Changes to the information model • Changes to the way the structures are organised into 'collections' within the message • Deprecation of the Agency, ID, Version option for referencing of structures in messages
SDMX-EDI	The EDI format for both structures and data has been deprecated.
SDMX-CSV	The CSV data and reference metadata messages are not backwardly compatible with those under version 2.1 due to changes to the structure of the messages needed to support new features such as the improved REST API data queries.

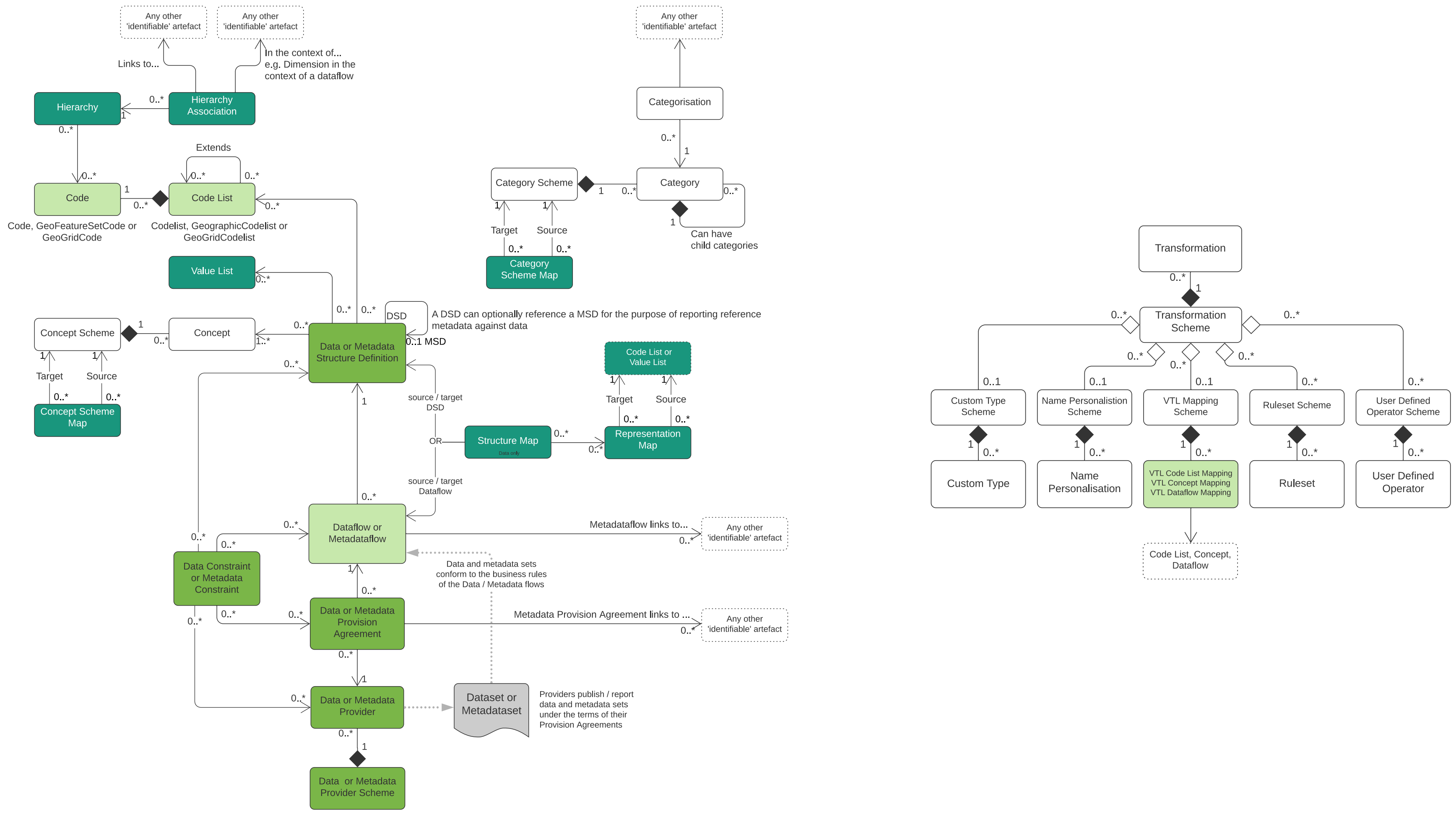
2.3 Information Model

Data Structure Definition	The version 3.0 Data Structure Definition (DSD) model is not directly backwardly compatible with 2.1 primarily due to the deprecation of the special MeasureDimension. *Conversion guidance for implementors* Version 2.1 DSDs can be converted to the 3.0 model by creating a measure with the “MEASURE” concept role applied as described in paragraph 3.5. Version 3.0 DSDs cannot be reliably converted to the 2.1 model due to the introduction of new features such as multiple measures and value arrays for measures and attributes.
Structure mapping model	The structure mapping model has changed significantly in version 3.0 with deprecation of the Structure Set maintainable artefact and introduction of five new ones: Representation Map and four variants of item scheme map. *Conversion guidance for implementors* Version 2.1 structure sets can be practically converted to the version 3.0 structure mapping model. Conversion from the version 3.0 structure mapping model to 2.1 is generally possible. However, when attempting to convert mapping rules from 2.1 to 3.0 and back to 2.1, the resulting Structure Set will not be precisely the same as the original. In converting to version 3.0, the system must generate IDs for each of the new maintainable artefacts, but details of the original Structure Set artefacts are lost.
Reference metadata model	The reference metadata model has changed in version 3.0 with modifications to the role of the Data Structure Definition, Metadata Structure Definition and Metadataflow artefacts. Metadata Provision Agreement and Metadata Provider Scheme have been added. Metadatasets are now identifiable. Version 2.1 reference metadata models are not valid in version 3.0. *Conversion guidance for implementors* A version 2.1 Metadata Structure Definition can be converted to the version 3.0 model under some circumstances, but target information is either lost or has to be translated into a metadataflow. Further, conversion of a Data Structure Definition for collecting reference metadata against a dataset would need to make changes to the dataset’s Data Structure Definition. As the Data Structure Definition may not actually be specified, judgement would need to be taken, perhaps determining the most likely candidate by examining which

	already have metadata reported against their datasets. A 2.1 metadata report could be converted to a version 3.0 Metadataset if it is attached to a structure, but requires a Metadata Provision Agreement which would need to be created if not already in existence. Conversion from the version 3.0 model to version 2.1 cannot be performed reliably. The process would need target information to be derived from analysis of the Metadataflows and Metadata Provision Agreements. Depending on the complexity it may not be possible to express that information in a version 2.1 Data Structure Definition.
Constraint model	The version 2.1 Content Constraint artefact has been deprecated in version 3.0 and replaced by the Data Constraint for data, and the Metadata Constraint for reference metadata. *Conversion guidance for implementors* 2.1 Content Constraints can be converted without loss to the equivalent version 3.0 Data Constraint model. Conversion from 3.0 to 2.1 presents challenges where wildcards have been used, in those cases requiring expansion of the wildcard into explicit values.
Hierarchical codelist structures	The version 2.1 Hierarchical Codelist artefact has been deprecated in version 3.0 and replaced by two new artefacts, Hierarchy and Hierarchy Association. *Conversion guidance for implementors* Version 2.1 Hierarchical Codelists can be successfully converted to the version 3.0 hierarchy model. Information on which artefacts to link the hierarchies to on what context would need to be added as a separate procedure. Conversion from the version 3.0 model to version 2.1 is possible, but with loss of the linking information.

3 Information Model

3.1 Version 3.0 Information Model



Any other 'identifiable' artefact
Links to...
In the context of... e.g. Dimension in the context of a dataflow
Hierarchy
Hierarchy Association
Extends
Code
Code List
Code, GeoFeatureSetCode or GeoGridCode
Codelist, GeographicCodelist or GeoGridCodelist
Value List
Concept Scheme
Concept
Target
Source
Concept Scheme Map
Data or Metadata Structure Definition
DSD
0..1 MSD
A DSD can optionally reference a MSD for the purpose of reporting reference metadata against data
source / target DSD
OR
source / target Dataflow
Structure Map
Data only
Code List or Value List
Representation Map
Categorisation
Category Scheme
Category
Can have child categories
Category Scheme Map
Dataflow or Metadataflow
Metadataflow links to...
Data and metadata sets conform to the business rules of the Data / Metadata flows
Data Constraint or Metadata Constraint
Data or Metadata Provision Agreement
Metadata Provision Agreement links to ...
Data or Metadata Provider
Dataset or Metadataset
Providers publish / report data and metadata sets under the terms of their Provision Agreements
Data or Metadata Provider Scheme
Transformation
Transformation Scheme
Custom Type Scheme
Name Personalistion Scheme
VTL Mapping Scheme
Ruleset Scheme
User Defined Operator Scheme
Custom Type
Name Personalistion
VTL Code List Mapping
VTL Concept Mapping
VTL Dataflow Mapping
Ruleset
User Defined Operator
Code List, Concept, Dataflow
1
0..*
0..1
1..*

Figure 1 Version 3.0 simplified Information Model UML class diagram with 'heat map' illustrating the areas with most change

The schematic above is a simplified UML class diagram of the SDMX 3.0 information model illustrating the major areas of change as a ‘heat map’. Darker colours indicate where new structures have been added in version 3.0 or where structures have been significantly changed.

A number of ancillary structures including organisation schemes, process and reporting taxonomy are unchanged and have not been shown. Similarly, Organisation Scheme Map and Reporting Taxonomy Map have been omitted for simplicity. A schematic of the 2.1 model is given in Appendix A for comparison purposes.

3.2 Key Changes from Version 2.1

New Maintainable Artefacts

- Structure Map
- Representation Map
- Organisation Scheme Map
- Concept Scheme Map
- Category Scheme Map
- Reporting Taxonomy Map
- Value List
- Hierarchy
- Hierarchy Association
- Metadata Constraint
- Data Constraint
- Metadata Provision Agreement
- Metadata Provider Scheme
- Metadataset

New Identifiable Artefacts

- GeoFeatureSetCode
- GeoGridCode
- Metadata Provider

Removed Maintainable Artefacts

- Structure Set – replaced by Structure Map and the four item scheme maps
- Hierarchical Codelist – replaced by Hierarchy and Hierarchy Association
- Constraint – replaced by Data Constraint and Metadata Constraint

Changed Maintainable Artefacts

- Data Structure Definition – support for microdatasets and reference metadata linked to data

- Metadataflow – simplifies exchange of reference metadata, in particular those linked to structures
- Metadata Structure Definition – simplified model for reference metadata
- Codelist – support for codelist extension and geospatial specialised codelists (GeographicCodelist, GeoGridCodelist)
- VTL Mapping Scheme – VTL Concept Mapping Scheme removed to align the VTL / SDMX interface with the 3.0 model

New Component Representation Types

- GeospatialInformation – a string type where the value is an expression defining a set of geographical features using a purpose-designed syntax

3.3 Areas Unchanged from Version 2.1

The following areas of the information model are unchanged from version 2.1:

- Categories
- Concepts
- Data providers
- Agencies
- Data consumers
- VTL transformation and expressions – with the exception of VTL mapping scheme as already noted
- Reporting taxonomy
- Process

3.4 Reference Metadata

Reference metadata has been substantially re-designed for version 3.0 to simplify the model and better support practical use cases.

Simplify Metadata Structure Definition

The Metadata Structure Definition (MSD) has been simplified to remove target information, and the support of multiple report structures. The MSD now only contains Metadata Attributes which are used to define the structure of a report.

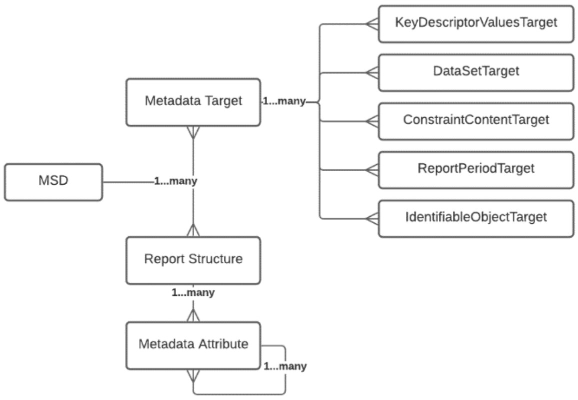


Figure 2 version 2.1 Metadata Structure Definition (MSD)

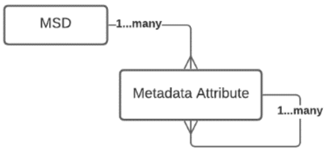


Figure 3 the simplified version 3.0 MSD

Change to reference metadata reported against data

Reference metadata associated with datasets, data series or observations are now reported with the data. The dataset's DSD must reference an MSD to define the structure of its reference metadata. In practice reference metadata for data are transmitted as part of the data message. The metadata attributes are treated in a similar way to the data attributes appearing in the message at the dataset, data series or individual observation level as appropriate. In contrast to simple data attributes, metadata attributes defined by an MSD can be organised into a hierarchical structure as illustrated in Figure 3 above. For this reason, metadata attributes appear in data messages structured in the same way as metadata messages.

The SDMX-ML example below is an excerpt from a structure specific data message illustrating reporting of reference metadata with a hierarchical structure at the observation level.

For completeness, the excerpt also shows:

- OBS_STATUS – a simple observation-level data attribute
- TITLE – a multi-lingual data attribute
- SOURCE_AGENCY – a multi-value data attribute

```
<Obs xsi:type="dsd:ObsType" OBS_VALUE="112" OBS_STAUS=”A” TIME_PERIOD="2010-09">
        <!—- complex multi-value and multi-lingual data attributes -->
        <Comp id="TITLE" xsi:type="ns1:TITLE_ATTRIBUTE">
                <Value>
                <common:Text xml:lang="en">Some English Text</common:Text>
                <common:Text xml:lang="fr">Quelques textes en anglais</common:Text>
                </Value>
        </Comp>
        <Comp id="SOURCE_AGENCY" xsi:type="ns1:SOURCE_AGENCY_ATTRIBUTE">
                <Value>4F0</Value>
                <Value>4D0</Value>
                <Value>CZ2</Value>
        </Comp>
        <!-- metadata attributes are reported like in metadata messages -->
        <Metadata>
                <Attribute id="COLLECTION">
                        <Attribute id="METHOD">
                                <Text lang="en">AAA</Text>
                        </Attribute>
                </Attribute>
                <Attribute id="CONTACT">
                        <Value>CONTACT 1</Value>
                        <Attribute id="NAME">
                                <Value>Contact 1 Name 1</Value>
                        </Attribute>
                        <Attribute id="NAME">
                                <Value>Contact 1 Name 2</Value>
                        </Attribute>
                </Attribute>
                <Attribute id="CONTACT">
                        <Value>CONTACT 2</Value>
                        <Attribute id="NAME">
                                <Value>Contact 2 Name 1</Value>
                        </Attribute>
                        <Attribute id="NAME">
                                <Value>Contact 2 Name 2</Value>
                        </Attribute>
                </Attribute>
        </Metadata>
</Obs>
```

New - Metadata Provision Agreement

In version 2.1 a Provision Agreement could be used to report information against a Dataflow or Metadataflow. From version 3.0 this is managed by two separate structures, the Data Provision Agreement and the Metadata Provision Agreement.

Move target to Metadataflow and Metadata Provision Agreement

For reference metadata that is reported against structures, the allowable targets information which is used to specify what structures the reference metadata can be reported against, has moved to the Metadataflow and can be further refined in the Metadata Provision Agreement.

Add maintainable properties to reference metadata

A Metadataset now has mandatory identification information, (owner id, id, version) enabling metadata providers to uniquely identify their reports for create, update or delete maintenance operations.

3.5 Microdata Exchange

Several changes have been made the Data Structure Definition to support microdata use cases in addition to aggregated time series.

Multiple measures

Multiple measures are a common characteristic of microdatasets. To support this use case, the MeasureDimension has been deprecated and replaced with the option to define zero or more measures. Measures now act like any other component in that they use concepts, can have their own local coded or uncoded representation defined within the Data Structure Definition, and can be either mandatory or conditional. Creating a measure with the “MEASURE” concept role applied emulates the version 2.1 MeasureDimension behaviour as illustrated in the SDMX-ML example below:

```
 <str:MeasureList id=”MeasureDescriptor”>
    <str:Measure id=”OBS_VALUE” minOccurs=”1” maxOccurs=”1” usage=”mandatory” >
      <str:ConceptIdentity>
        <Ref id=”OBS_VALUE” maintainableParentID=”CONCEPTS” agencyID=”SDMX”
          maintainableParentVersion=”1.0.0” />
      </str:ConceptIdentity>
      <str:LocalRepresentation>
        <str:TextFormat textType=”String” isMultiLingual=”true” />
      </str:LocalRepresentation>
      <str:ConceptRole>
        <Ref id=”MEASURE” maintainableParentID=”SDMX_CONCEPT_ROLES” agencyID=”SDMX”
          maintainableParentVersion=”1.0.0” />
      </str:ConceptRole>
    </str:Measure>
       ...
    <str:Measure>
  </str:MeasureList>
```

Multi-value measures and attributes

Both measures and attributes have been extended with the option to take ‘arrays’ of multiple coded or uncoded values. This supports use cases like multiple observation status flags. New *minOccurs* and *maxOccurs* properties define the valid number of values. The *usage* property separately defines whether the measure or attribute is *mandatory* or optional. In the SDMX-ML measure example above, the properties *minOccurs=”1” maxOccurs=”1” usage=”mandatory”* specify that OBS_VALUE must be reported, and can only consist of a single value.

Attributes relationship to measures

In addition to attaching attributes to a specific level within the dataset, their relationship to measures can also be defined.

Value lists

Value lists help in modelling microdata by providing an enumeration similar to code lists but allowing any string values without being restricted to the rules of SDMX identifiers. That allows ValueItems (the equivalent to Code) to contain symbols like ‘¥’ and ‘€’, but also means they are not identifiable.

3.6 Geospatial Data Exchange

The version 3.0 model has been extended to provide explicit support for geospatial data.

GeospatialInformation type

A new GeospatialInformation string type has been added which can be used as the representation for any dimension, attribute or measure component. The value which is a string expression conforming to the syntax defined in Section 6 of the technical specifications precisely defines a ‘Geo Feature Set’ – a collection of geographical features like points, lines or polygons. Its use is recommended in conjunction with the “GEO_FEATURE_SET” concept role.

Geospatial code lists

Two new specialised types of code list have been added where the definition of each code includes additional geospatial information in addition to the standard ID, name and description:

- GeographicCodelist – each item includes an element to represent a specific Geo Feature Set which is described using the same expression syntax as for GeospatialInformation type.
- GeoGridCodelist – A code list defining a geographical grid composed of cells representing regular squared portions of the Earth. Each item references a cell within the grid.

3.7 Structure Mapping

The Structure Set in version 2.1 is a container for many mapping structures including Data Structure Map, Codelist Map and Concept Map. For version 3.0 the Structure Set artefact has been deprecated and replaced with a number of new maintainables giving better flexibility and reusability, specifically: Structure Map, Concept Scheme Map, Representation Map, Reporting Taxonomy Map, Category Scheme Map and Organisation Scheme Map.

The version 2.1 Codelist Map been replaced with Representation Map which allows mappings to be defined between any combination of Code Lists, Value Lists and non-coded representations such as text strings and numbers.

Many-to-many source and target components

Structure mapping rules may be defined with both multiple source components and multiple target components in contrast to version 2.1 where only one source and target was allowed. That supports many-to-many (n-n) mapping use cases where the output of a mapping rule may be dependent on the combination of a number of input components. For instance:

Set the output component INDICATOR=”DE_A” if the input components are FREQ=”A” and REF_AREA=”DE”.

Similarly, an n-n rule may also set the values of any number of output components:

Set the output components FREQ=”A”, REF_AREA=”DE” if the input component INDICATOR=”DE_A”.

Fixed source and target

The Structure Map may now define input or output components which have a fixed value.

Time representations mapping

Non SDMX time representations may now be described in a Structure Map, allowing them to be mapped into SDMX time formats.

Regular expression and substring mappings

All item maps allow the use of regular expressions and substrings to match source values, specifically: Concept Scheme Map, Reporting Taxonomy Map, Category Scheme Map and Organisation Scheme Map.

Item maps validity period

Item maps may further define the period for which the mapping is valid, meaning the mapping rule will only be applied if the row of information being mapped is within the period.

3.8 Constraints

Constraints in version 3.0 are modelled using two separate artefacts which replace the version 2.1 content constraint:

- data constraint for data; and
- metadata constraint for reference metadata.

Metadata constraint differs from its data counterpart in having a simplified cube region model better suited to reference metadata reporting use cases and not carrying details of the constrained targets – that information instead being defined directly within the metadataflow and Metadata Provision Agreement. Thus, metadata related constraints only specify constraints to the values of metadata attributes.

The ‘%’ wildcard character can now be used when defining cube region constraints to match multiple codes with a single expression, for instance for economic activity, ISIC4_% matches all codes beginning with ‘ISIC4_’ avoiding the need to maintain an explicit list.

The validity period definition has been moved from the constraint to the individual constraining terms, specifically CubeRegion, DataKeySet and MetadataTargetRegion providing more granular control.

Attachment constraints have been deprecated due to a lack of use cases.

3.9 Code List Extension

In addition to the two new specialised geospatial forms, the option has been added to define a code list as an extension of, or by inheriting codes from, other lists. An optional prefix can be added to inherited codes to disambiguate duplicates.

This feature allows new code lists to be easily derived from existing lists without the need to make and manually maintain copies. When querying for extended code list structures using the REST API, the option has been added to retrieve either the definition or the materialised list. Traditional literal lists of codes continue to be supported.

3.10 Discriminated Union of Code Lists

Combining code list extension with wildcarded constraints solves the discriminated union of code lists problem where a classification or breakdown has multiple “variants” which are all valid but mutually exclusive. A common example is economic activity where several alternative classification schemes are in use including ISIC revisions 1 to 4 and NACE as used in the European Community.

3.11 Code Hierarchies

Code hierarchies allow the definition of complex hierarchies of codes from potentially multiple lists for data discovery purposes. Hierarchical Codelist has been deprecated and replaced by two new artefacts: Hierarchy – the actual hierarchy of codes, and Hierarchy

Association links hierarchies directly to any other identifiable object, a capability missing from the version 2.1 model. Further, the linkage can be within a particular context, for instance linking a hierarchy to a dimension within the context of a specific Dataflow (dimension REF_AREA in the context of the ECB:EXR Dataflow).

4 Versioning of Structural Metadata Artefacts

Version 3.0 adopts semantic versioning principles for versioning of metadata artefacts following the rules set out at https://semver.org. However, this is not mandatory, and organisations may continue to use the pre-existing two-digit versioning strategy, or not to version artefacts by omitting the *version* property. The version number no longer defaults to 1.0 if not explicitly set.

Semantic version numbers are three digits:

```
MAJOR.MINOR.PATCH
```

Where

- The first digit (major) indicates that changes (either new features or bug fixes) are not backward compatible.
- The second digit (minor) indicates that features have been added in a backward compatible manner.
- The third digit (patch) indicates that bugs have been fixed in a backward compatible manner.

Examples:

SDMX:CL_AREA(1.0.0)

SDMX:CL_AREA(2.3.2)

Dependency management

Additional constructs are possible for dependency management when referencing structures. For instance:

2.3+.1	Means the currently latest available version >= “2.3.1” and < “3.0.0” (all backwards compatible versions >= “2.3.1”).
2+.3.1	Means the currently latest available version >= “2.3.1” (even if not backwards compatible).

Draft structures

A key principle is that semantically versioned structures are immutable and must not be changed without a corresponding change to the version number, except where explicitly marked as draft using extensions to the version number.

```
MAJOR.MINOR.PATCH-EXTENSION
```

1.10.0-draft	Means that version 1.10.0 is still being modified and may change – equivalent to setting isFinal=false in SDMX 2.1.
1.10.0-unstable	Alternative to -draft.
1.10.0-notfinal	Alternative to -draft.

The SDMX 2.1 isFinal property is deprecated in 3.0.

5 REST Web Services API

5.1 Simplified list of resources

The version 3.0 REST API has just five main resources:

- structure
- data
- schema
- availability
- metadata

All structure and item queries have been organised under the structure resource in contrast to the version 2.1 API which specified a separate resource for each structure.

This and changes in the URLs and query parameters on the data, availability and metadata resources means that, with the exception of schema queries, the version 3.0 API is not backwardly compatible.

5.2 Improved data queries

Data queries have been changed to provide more granular selections from contexts wider than just a Dataflow.

Extend the context of data retrieval

Version 2.1 data queries always retrieved data from a single specific Dataflow. In version 3.0, the query context may be specified as:

- Dataflow;
- Data Structure Definition – i.e., all Dataflows that use it; or
- Provision Agreement – i.e., all Dataflows associated with it.

Data queries may also search across datasets, for instance “retrieve all data about a country”.

Component-based filters

Expressions filtering on individual components can now be included as part of the data query URL.

```
/data/dataflow/ESTAT/ICP?c[REF_AREA]=CH&c[CONF_STATUS]=F
```

Support for operators

Filter expressions can also include operators.

```
/data/dataflow/ESTAT/ICP?c[REF_AREA]=DE&c[ICP_ITEM]=sw:01&c[TIME_PERIOD]=ge:2015
```

Operators include:

eq	Equals
ne	Not equal to
le	Less than
ge	Greater than or equal to
sw	Starts with

Support for multiple keys

Queries can now specify multiple series keys.

```
/data/dataflow/ESTAT/ICP/1.0.0/M…A.ANR,M…A.INX,M…B.CTG
```

5.3 Improved reference metadata queries

Reference metadata queries have been improved with a number of new options to retrieve metadata reports.

Get metadata reports by ID

```
/metadata/metadataset/ESTAT/QUALITY_REPORT/1.0.0
```

Get metadata reports by Dataflow

```
/metadata/metadataflow/ECB/METHODOLOGY/*/FR2
```

Get metadata reports about a Data Structure Definition

```
/metadata/structure/datastructure/BIS/BIS_CBS/1.0
```

5.4 Structural metadata maintenance

Support has been added for maintenance of structural metadata.

HTTP verbs PUT, POST and DELETE may be used to submit SDMX-ML or SDMX-JSON structure messages to an SDMX registry for the purposes of adding, updating or deleting structural metadata artefacts.

6 XML, JSON, CSV and EDI Transmission formats

6.1 SDMX-ML

The SDMX-ML XML messages have been modified and updated for version 3.0. While they broadly follow the same principles, there have been significant changes which break backward compatibility.

Structure message

The SDMX-ML structure message is used for transmission of structural metadata. It closely reflects the SDMX information model and has therefore been significantly updated for version 3.0 with the addition of new structures, modifications where structures have changed, and removal of deprecated structures like Structure Set.

Additionally, the way the individual artefacts are organised into 'collections' within the message has been significantly revised with a simpler flat structure adopted as set out in the following table:

Artefact type	Version 2.1 Collection	Version 3.0 Collection
AgencyScheme	OrganisationSchemes	AgencySchemes
DataConsumerScheme	OrganisationSchemes	DataConsumerSchemes
DataProviderScheme	OrganisationSchemes	DataProviderSchemes
MetadataProviderScheme	OrganisationSchemes	MetadataProviderSchemes
OrganisationUnitScheme	OrganisationSchemes	OrganisationUnitSchemes
GeographicCodelist	Codelists	GeographicCodelists
GeoGridCodelist	Codelists	GeoGridCodelists
ConceptScheme	Concepts	ConceptSchemes
ValueList	Codelists	ValueLists
StructureMap	StructureMappings	StructureMaps
RepresentationMap	StructureMappings	RepresentationMaps
ConceptSchemeMap	StructureMappings	ConceptSchemeMaps
CategorySchemeMap	StructureMappings	CategorySchemeMaps
OrganisationSchemeMap	StructureMappings	OrganisationSchemeMaps
ReportingTaxonomyMap	StructureMappings	ReportingTaxonomyMaps
DataConstraint	Constraints	DataConstraints
MetadataConstraint	Constraints	MetadataConstraints
MetadataProvisionAgreement	ProvisionAgreement	MetadataProvisionAgreements
CustomTypeScheme	CustomTypes	CustomTypeSchemes
VtlMappingScheme	VtlMappings	VtlMappingSchemes
NamePersonalisationScheme	NamePersonalisations	NamePersonalisationSchemes
RulesetScheme	Rulesets	RulesetSchemes
TransformationScheme	Transformations	TransformationSchemes
UserDefinedOperatorScheme	UserDefinedOperators	UserDefinedOperatorSchemes

No changes have been made to the way the following artefacts are organised in the structure message:

Artefact type	Collection
Dataflow	Dataflows
Metadataflow	Metadataflows
CategoryScheme	CategorySchemes
Categorisation	Categorisations

Codelist	Codelists
Hierarchy	Hierarchies
HierarchyAssociation	HierarchyAssociations
MetadataStructure	MetadataStructures
DataStructure	DataStructures
ReportingTaxonomy	ReportingTaxonomies
Process	Processes
ProvisionAgreement	ProvisionAgreements

From version 3.0, collections can appear in any order within a structure message.

Data messages

All legacy SDMX-ML data messages have been deprecated with the exception of Structure Specific Data which becomes the sole standard format for transmission of SDMX data in XML in version 3.0.

Specifically, the following data messages are not supported in version 3.0:

- SDMX-ML 1.0/2.0 Generic (time-series) data message
- SDMX-ML 1.0/2.0 Compact (time-series) data message
- SDMX-ML 1.0/2.0 Utility (time-series) data message
- SDMX-ML 1.0/2.0 Cross-Sectional data message
- SDMX-ML 2.1 Generic data messages (for observations, time-series and cross-sectional data)

The Structure Specific Data message has been extended to support the transmission of microdata sets, in particular those with multiple measures and array values for measures and attributes.

As detailed in paragraph 3.4, the message now additionally allows data's reference metadata to be reported as an integral part of the dataset. Like data attributes, these metadata attributes are included in the data message at the dataset, series or observation level as appropriate.

The time series variant of the Structure Specific Data message is no longer used.

Reference metadata message

The Generic Metadata message remains the standard format for transmission of reference metadata sets in XML but has been modified to support the revised version 3.0 reference metadata model.

Registry structural metadata 'query' messages

As a consequence of the deprecation of the SOAP API and standardisation on REST, the structural metadata 'query' messages have all been removed. In version 3.0, querying an SDMX Registry for structural metadata is performed solely using REST GET.

Structure referencing

The option to reference structures using Agency, ID and Version has been removed. From SDMX version 3.0 URN is used for all referencing purposes with the exception of local references such as where groups reference dimensions within a DSD.

6.2 SDMX-JSON

Like SDMX-ML, the SDMX-JSON messages have been significantly modified and updated for version 3.0. They are not backwardly compatible with version 2.1.

Structure message

The SDMX-JSON structure message closely replicates the SDMX-ML equivalent. Like that of SDMX-ML it has been updated to align it with the version 3.0 information model with addition, deletion and modification of artefacts as required. The organisation of the structure collections has also been revised as detailed in paragraph 6.1.

Data message

The SDMX-JSON data message has similarly be updated. Additional changes have been made to allow a single message to carry data from multiple datasets with potentially different Data Structure Definitions to support REST data queries of the form "retrieve all data about a country". For this reason, the version 3.0 SDMX-JSON is not backwardly compatible with version 2.1 data messages. Support has been added for the transmission of microdata and reporting of reference metadata on data as an integral part of the dataset.

Reference metadata message

The SDMX-JSON metadata message has also been updated to support the version 3.0 reference metadata and Metadataset specifications.

Structure referencing

As for SDMX-ML, the option to reference structures using Agency, ID and Version has been removed with URN used for all non-local referencing purposes.

6.3 SDMX-CSV

CSV in SDMX is used transmission of data and reference metadata only.

Data message

The SDMX-CSV data message has been modified to align with the version 3.0 information model, support the enhanced REST API and ensure that data can be freely converted to and from the XML and JSON formats without loss. These changes include:

- An additional column identifying the type if the artefact defining the structure of the data: "dataflow", "datastructure" or "dataprovision";
- A column for the structure artefact's identification of the form `ESTAT:NA_MAIN(1.6.0)` which replaces the dataflow identifier in version 2.1; and
- A column for the dataset action: information, append, replace or delete, which is consistent with both the the SDMX-ML and SDMX-JSON data messages.

Reference metadata message

The SDMX-CSV metadata message is new for version 3.0 and, like the SDMX-ML and SDMX-JSON equivalents, is used for the transmission reference metadata sets.

6.4 EDI deprecation

The EDI format for transmission of both structures and data has been deprecated. Version 3.0 is therefore not backwardly compatible with legacy EDI messages.

Appendix A – Version 2.1 Information Model

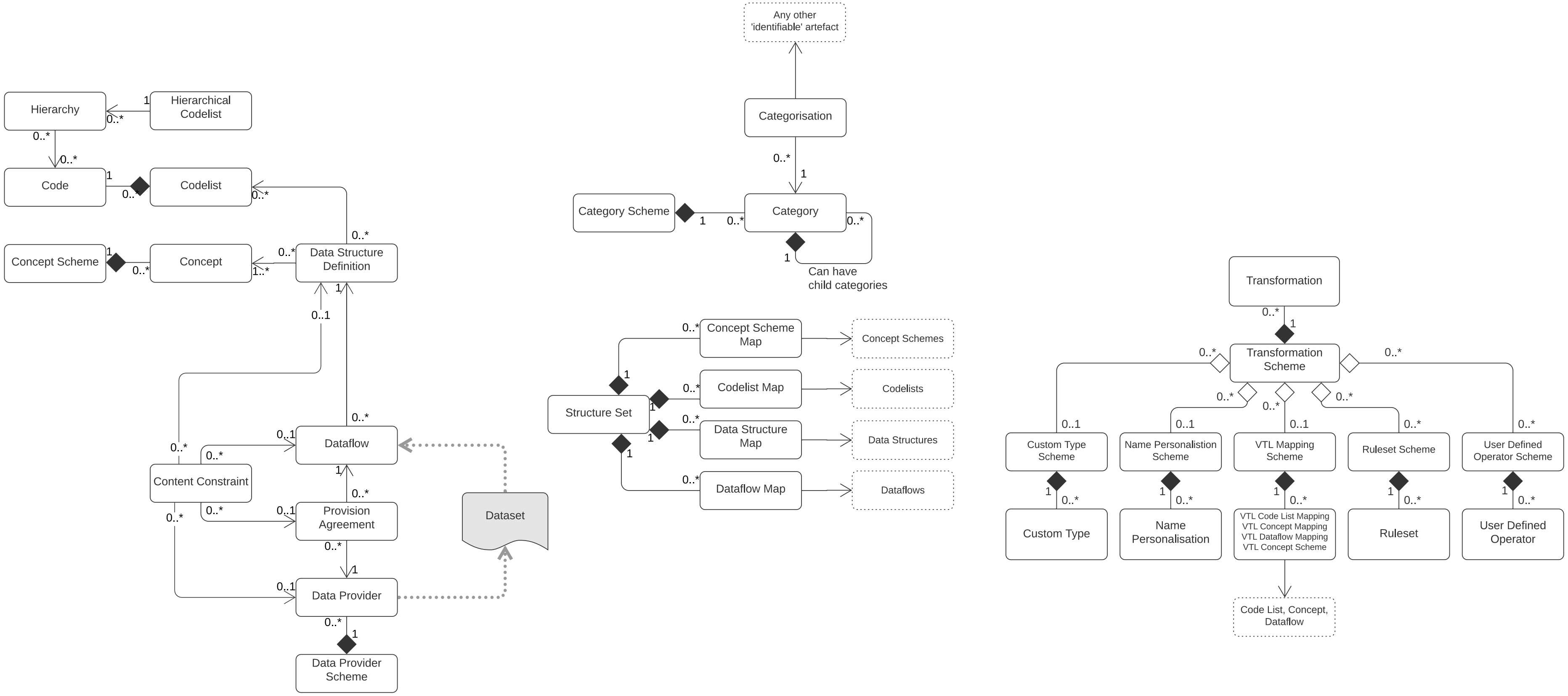


Figure 4 Version 2.1 simplified Information Model UML class diagram