SDMX

IMPLEMENTORS GUIDE

(VERSION 2.0)

November 2005

Contents

1 PURPOSE OF THIS DOCUMENT

The intention of this document is twofold:

1. To document certain aspects of SDMX that are specific to implementations. This covers several topics, but has a focus on implementation of Key Families and Data Sets, especially where there is a need to support interoperability between SDMX-ML and SDMX-EDI formats.

2. To explain the SDMX Information Model (SDMX-IM), by representing the model first in simple diagrammatic terms, and then by mapping the diagrams to the UML model. In addition examples of the use of the model are described, and in some cases screen shots are provided from the implementation of some aspects of the model in a sample graphical user interface.

The intended audience is technical specialists who wish to gain a high-level overview of the scope of the SDMX Information Model and how it is organised in order to:

- use it as an introduction to the SDMX Information Model and specific aspects of implementation – this will be useful for developers of tools and other applications that implement the SDMX structures

- use it as a guide to the information model for those readers who do not need to understand all of its technical aspects

This document describes the scope of the SDMX-IM and describes its structures and how they are derived from basic patterns in the model. It builds up the model from the basic structures needed for identification, versioning and maintenance, to the structures that form basic "patterns" that are re-used in various forms in other parts of the model, to the functional part of the model that uses these patterns.

2 General Notes On This Document

Some UML diagrams are included in this document. In some instances an explanation is given on some aspects of the diagrammatic notation in UML. However, for a general explanation of the diagrammatic notation used in the SDMX-IM please refer to Appendix I of the SDMX Information Model version 2.0.

The SDMX-IM models a class called "Key Family". This term is not familiar to many people and its name is taken from the model of SDMX-EDI (previously known as GESMES/TS). A more familiar name is "Data Structure Definition" and this is the name that is used in this document. The exception to this is that Key Family is used where it is felt that the reference is directly to the class in the model or to the element name in the schema. Sometimes the text "Data Structure Definition (Key Family)" is used. In general, the terms "Key Family" and "Data Structure Definition" are interchangeable.

3 GUIDE FOR SDMX FORMAT STANDARS

3.1 Introduction

This section of the guide exists to provide information to implementors of the SDMX format standards – SDMX-ML and SDMX-EDI – that are concerned with data, i.e. Key Families (also known as Data Structure Definition) and Data Sets. This section is intended to provide information which will help users of SDMX understand and implement the standards. It is not normative, and it does not provide any rules for the use of the standards, such as those found in *SDMX-ML: Schema and Documentation* and *SDMX-EDI: Syntax and Documentation*.

This document is organized into parts:

- A guide to the SDMX Information Model relating to Key Families and Data Sets
- Statement of differences in functionality supported by the different formats and syntaxes for Key Families and Data Sets
- Best practices for use of SDMX formats

3.2 SDMX Information Model for Format Implementors

3.2.1 Introduction

The purpose of this sub-section is to provide an introduction to the SDMX-IM relating to Key Families and Data Sets for those whose primary interest is in the use of the XML or EDI formats. For those wishing to have a deeper understanding of the Information Model, the full SDMX-IM document, and other sections in this guide provide a more in-depth view, along with UML diagrams and supporting explanation. For those who are unfamiliar with key families, an appendix to the SDMX-IM provides a tutorial which may serve as a useful introduction.

The SDMX-IM is used to describe the basic data and metadata structures used in all of the SDMX data formats. There is a primary division between time series and cross-sectional data and the metadata which describes the structure of that data. The Information Model concerns itself with statistical data and its structural metadata, and that is what is described here. Both structural metadata and data have some additional metadata in common, related to their management and administration. These aspects of the data model are not addressed in this section and covered elsewhere in this guide or in the full SDMX-IM document..

The Key Family and Data Set parts of the information model are consistent with the GESMES/TS version 3.0 Data Model, with these exceptions:

- the “sibling group” construct has been generalized to permit any dimension or dimensions to be wildcarded, and not just frequency, as in GESMES/TS. It has been renamed a “group” to distinguish it from the “sibling group” where only frequency is wildcarded. The set of allowable partial “group” keys must be declared in the key family, and attributes may be attached to any of these group keys;

- the section on data representation is now a convention, to support interoperability with EDIFACT-syntax implementations (see section 3.3.2);

- cross-sectional data formats are derived from the model, and some supporting features for deriving cross-sectional and time-series views of a single data set structure have been added to the structural metadata descriptions.

Clearly, this is not a coincidence. The GESMES/TS Data Model provides the foundation for the EDIFACT messages in SDMX-EDI, and also is the starting point for the development of SDMX-ML.

Note that in the descriptions below, text in courier and italicised are the names used in the information model (e.g. *DataSet*).

3.2.2 Fundamental Parts of the Information Model

The words in Italics refer to classes in the SDMX-IM.

The statistical information in SDMX is broken down into two fundamental parts - structural metadata (comprising the *KeyFamily*, and associated *Concepts* and *Code Lists*) – see Framework for Standards -, and observational data (The *DataSet*). This is an important distinction, with specific terminology associated with each part. Data - which is typically a set of numeric observations at specific points in time - is organized into data sets (*DataSet*) These data sets are structured according to a specific Data Structure Definition (*KeyFamily*), and are described in the data flow definition (*DataflowDefinition*) The Data Structure Definition describes the metadata that allows an understanding of what is expressed in the data set, whilst the data flow definition provides the identifier and other important information (such as the periodicity of reporting) that is common to all of its component data sets.

Note that the role of the Data Flow (called *DataflowDefintion* in the model) and Data Set is very specific in the model, and the terminology used may not be the same as used in all organisations, and specifically the term Data Set is used differently in SDMX than in GESMES/TS. Essentially the GESMES/TS term “Data Set” is, in SDMX, the “Dataflow Definition” whist the term “Data Set” in SDMX is used to describe the “container” for an instance of the data.

3.2.3 Data Set

Data sets are made up of a number of time series or sections (the cross-sectional organization of observations at a single point in time). In addition to the numeric observation (*Observation*) and the related date (*TimePeriod*), which are the core of the time series, there may be attributes (*AttributeValue*) indicating the status of the observation, e.g. whether the value is a normal or break value, etc. These attributes may be optional (or “conditional”), and may have coded or free text values. They may pertain to any part of the data set - each observation might have a different value for the attribute, or there might be only a single attribute value describing the entire data set, or for each time series, etc.

Each time series can be identified by the values of its dimensions. Time series data can be seen as n-dimensional. A given time series will have exactly one value

(`KeyValue`), of the set of permissible values, for each of its dimensions (`Dimension`), and a set of observations (*Observation*): one value for each specific point in time (`TimePeriod`). A specific time series might have dimensions of "frequency", "topic", "stock or flow", "reporting country", etc., with a single corresponding value for each dimension. Taken together, this set of values uniquely identifies the time series within its data set, and is called the time series key (`TimeSeriesKey`).

Cross-sectional representations of the data may be derived from the same Data Structure Definition from which time-series representations are structured, so long as the needed additional structural metadata is provided. This functionality allows multiple measures to be declared in the Data Structure Definition, associated with the representational values of one dimension. When data is structured to represent a set of multiple observations at a single point in time, the "section" – one or more observations for each declared measure – replaces the series in the data structure. Each measure carries at least one dimension of the key (the "measure dimension") at the observation level, while the time period is attached at a higher level in the data structure (the Group level – see below). The remainder of the key is found at the Section level (or above), similar to the way in which it is attached at the Series level for time series data structures.

Support for cross-sectional data representation is not as complete as that for representing time-series data. The intended functionality is to allow key families which are to be used to represent cross-sectional data to be created with this application in mind. Because time-series data representations are also possible for any Data Structure Definition which has time period as a concept, these data structures may also be derived from the Data Structure Definition. The functional result is that two complementary types of data structures may be provided: the needed cross-sectional view, and the time-series oriented view which may be useful to systems which may not be configured to process data in any other fashion. The Data Structure Definition created to support cross-sectional structuring of data will support the predictable (and thus, automatable) transformation of data from the cross-sectional structure into the time-series structure.

Data sets may be organized into "groups" of time series or sections (`GroupKey`); this is a particularly useful mechanism for attaching metadata to the data. One such group is called the "sibling group", which shares dimension values for all but the frequency dimension (the frequency dimension is said to be "wildcarded"). In the Data Structure Definition, all legitimate groups are declared and named. All members of the group will share key values for a stated set of dimensions. Attributes may be attached at this level in the data formats, as are the shared key values for those formats where message size is an issue. In cross-sectional formats, time period (a period or point in time) is attached at the group level.

The data structure definition is a description of all the metadata needed to understand the data set structure. This includes identification of the dimensions (`Dimension`) according to standard statistical terminology, the key structure (`KeyDescriptor`), the attributes (`MetadataAttribute`) associated with the data set, the code-lists (`CodeList`) that enumerate valid values for each dimension and coded attribute (`CodedAttribute`), information about whether attributes are required or optional and coded or free text. Given the metadata in the data structure definition, all of the data in the data set becomes meaningful.

It is also possible to associate annotations (`Annotation`) with both the structures described in key families and the observations contained in the data set. These annotations are a slightly atypical form of documentation, in that they are used to describe both the data itself - like other attributes - but also may be used to describe other metadata. An example of this is methodological information about some particular dimension in a data structure definition structure, attached as an annotation to the description of that dimension. Regular "footnotes" attached to the data as documentation should be declared as attributes in the appropriate places in a data structure definition – annotations are irregular documentation which may need to be attached at many points in the data structure definition or data set.

The following section provides more complete definitions of the SDM-IM as it relates to statistical data, for easy reference by syntax implementors.

3.2.4 Attachment Levels and Data Formats

It is worth looking briefly at the available formats in light of the discussion above:

- SDMX-ML and SDMX-EDI both have a format for describing key families, concepts, and codelists.

- In SDMX-EDI, there is a single message format for transmitting data-related messages. This format allows for very compact expression of different types of packages of information: just data, just documentation, delete messages, etc. This format is time-series-oriented. Time is specified either as a range for a set of observation values with a known frequency, or is associated on a one-to-one basis with observation values.

- In SDMX-ML, the Generic Data Message requires that all key values be specified at the Series level, and that attribute values be attached at the level in which they are assigned in the data structure definition (if any attribute values are to be transmitted). This is a time-series-oriented format, which requires that a time be specified for each observation value.

- In SDMX-ML, the Compact Data Message requires the values of keys to be specified at the Series level. Attribute values are specified at the level assigned to them in the data structure definition, if provided. This is a time-series-oriented format, which associates time with observations either on a one-for-one basis, or expressed as a range for a set of observations with a known frequency.

- In the SDMX-ML Utility Data Message, all key values are attached at the Series level, and all attribute values are attached at the level of assignment in the data structure definition. Attribute values must be provided – there is no concept of a "delete" message or partial message (for updates, documentation-only, etc.) as there is for other data formats. This is a time-series-oriented format which requires that time be specified for each observation on a one-for-one basis.

- In the SDMX-ML Cross-Sectional Data Message, attachment levels vary more than in other formats. Key values may be attached at any level or combination of levels, as declared in the data structure definition, with the exception that time is always attached at the group level for those key families which use time as a concept. Key values may be attached at the observation level for each type of declared measure. Attribute values may be provided at any of the levels assigned in the data structure definition. This is the only non-time-series-oriented format.

- SDMX-ML has a Query message, but discussion of the attachment levels is not relevant for this message.

3.2.5 SDMX-IM Concepts, Definitions, and Rules

This sub-section provides a narrative description of the SDMX-IM.

3.2.5.1 Key Family (Data Structure Definition)

A Key Family defines the valid content of a Data Set in terms of the Concepts comprising the structure of the Data Set, how the Concepts are related in terms of their role in the Data Set, and the valid content of each of the Concepts when used in a Data Set.

A Key Family comprises:

- A mandatory Key Descriptor that defines the structure of the Key when data are reported in a Data Set
- Optional additional Group Key Descriptor that are a sub set of the Key Descriptor, each of which identifies a part of the Key that are reported in a Data Set, and whose purpose is to define a part of the Data Set to which an Attribute may be attached
- A Measure Descriptor which identifies the Measures for which data can be reported
- An Attribute Descriptor which identifies the Data Attributes for which metadata can be reported

The Key Descriptor specifies an ordered set of Dimensions, each of which takes its semantic from a Concept in a Concept Scheme which defines the semantic and the core Representation and core Type to which the Dimension must conform when the value of the Dimension is contained in a Key in a Data Set.

A Dimension can optionally identify an alternative Representation and Type to which the Dimension must conform when the value of the Dimension is contained in a Key in a Data Set. If such an alternative Representation and Type is identified it overrides any core Representation and Type that may have been defined for the Concept.

The Dimension may play a specific role (Concept Role) in the Key Family (such as Frequency, Time).

A Measure Type Dimension is a special type of Dimension that specifies the type of measure (such a index, count, percentage change) for an Observation in the Data Set. There can be many Measure Type Dimensions.

The Group Key Descriptor defines the context of a sub set of the Data Set for which an Attribute Value may be reported. This context is defined in terms of the Dimensions that can comprise a Group Key in the Data Set, such Dimensions must be present in the Key Descriptor (i.e. the contents Group Key Descriptor is a sub set of the contents of the Key Descriptor).

The Measure Descriptor specifies the Measures that can be reported in a Data Set. Each Measure can be a Coded Measure or an Uncoded Measure.

A Coded Measure identifies a Concept in a Concept Scheme which defines the semantic and the core Representation and core Type to which the Coded Measure must conform when metadata is reported in a Data Set. It can also optionally identify a (local) Code List and Type which contains the valid values that the Coded Measure can take when reported in a Data Set. If such a Code List is identified it overrides any core Representation that may have been defined for the Concept.

An Uncoded Measure identifies a Concept in a Concept Scheme which defines the semantic and the core Representation and Type to which the Uncoded Measure must conform when reported in a Data Set. It can also optionally identify an alternative (local) non coded Representation and Type to which the Uncoded Measure must conform when reported in a Data Set If such an alternative Representation and Type is identified it overrides any core Representation and Type that may have been defined for the Concept.

The Measure may play a specific role (Concept Role) in the Key Family (such as Primary Measure).

The Attribute Descriptor specifies a flat structure of one or more Data Attributes, each of which is either a Coded Data Attribute or an Uncoded Data Attribute.

A Coded Data Attribute identifies a Concept in a Concept Scheme which defines the semantic and the core Representation and Type to which the Coded Data Attribute must conform when metadata is reported in a Data Set. It can also optionally identify a (local) Code List and Type which contains the valid values that the Coded Data Attribute can take when reported in a Data Set. If such a Code List is identified it overrides any core Representation that may have been defined for the Concept.

An Uncoded Data Attribute identifies a Concept in a Concept Scheme which defines the semantic and the core Representation and Type to which the Uncoded Data Attribute must conform when reported in a Data Set. It can also optionally identify an alternative (local) non coded Representation and Type to which the Uncoded Data Attribute must conform when reported in a Data Set If such an alternative Representation and Type is identified it overrides any core Representation and Type that may have been defined for the Concept.

The Data Attribute may play a specific role (Concept Role) in the Key Family.

The Data Attribute is specified as being linked to one of the Key Descriptor, specific Group Key Descriptor, specific Measure or the Data Set. This specification defines the allowable Identifiable Artefacts for which the Data Attribute can be reported in a Data Set.

3.2.5.2 Data Set

The Data Set contains data and related metadata whose content conforms to the specification of a Data structure definition.

The Data Set comprises:

- One or more Timeseries Keys, each of which defines the key of a timeseries which, when combined with a Time Period uniquely identifies an Observation
- One of more Group Keys which comprises a set of Timeseries Keys for which Attribute Values can be reported
- One or more Attribute Values, which are reported for a specific object as identified by an Attachable Artefact. An Attachable Artefact is one of Timeseries Key, Group Key, Observation, or Data Set.

A Timeseries Key comprises one or more Key Values, each of which is a value for the relevant Dimension specified in the Key Descriptor of the Key Family.

A Group Key comprises one or more Key Values, each of which is a value for the relevant Dimension specified in the Group Key Descriptor of the Key Family.

An Observation is one of Coded Observation or Uncoded Observation.

A Coded Observation contains a Code consistent with the Code List specified for the relevant Coded Measure in the Key Family.

An Uncoded Observation contains a value that is consistent with the Representation and Type specified for the relevant Uncoded Measure in the Key Family.

An Attribute Value is one of Coded Attribute Value or Uncoded Attribute Value.

A Coded Attribute Value is the value of the metadata reported for a Coded Data Attribute as specified in the Key Family. The value is reported for a specific Timeseries Key, Group Key, Observation, or Data Set.

An Uncoded Attribute Value is the value of the metadata reported for an Uncoded Data Attribute as specified in the Attribute Descriptor of the Key Family. The value is reported for a specific Timeseries Key, Group Key, Observation, or Data Set.

3.2.5.3 Code Lists, Category Schemes, Concept Schemes

3.2.5.3.1 Code List

A Code List is maintained by a Maintenance Agency.

A Code List comprises one or more Codes. The Code may have one or more child Codes thus forming a simple hierarchy of Codes.

Both a Code List and a Code may have additional Properties specified which define metadata for the Code List or Code. The Property is an extensible mechanism for defining additional metadata and specifies a name, a type (data type), and a value.

3.2.5.3.2 Category Scheme

A Category List is maintained by a Maintenance Agency.

A Category List comprises one or more Categories. The Category may have one or more child Categories thus forming a simple hierarchy of Categories.

Both a Category List and a Category may have additional Properties specified which define metadata for the Category List or Category. The Property is an extensible mechanism for defining additional metadata and specifies a name, a type (data type), and a value.

3.2.5.3.3 Concept Scheme

A Concept Scheme is maintained by a Maintenance Agency.

A Concept scheme comprises one or more Concepts. The Concept may have one or more child Concepts thus forming a simple hierarchy of Concepts.

Both a Concept Scheme and a Concept may have additional Item Properties specified which define metadata for the Concept Scheme or Code. The Item Property is an extensible mechanism for defining additional metatdata and specifies a name, a type (data type), and a value.

A Concept can have both a Type and a Representation specified. If specified this is the core representation that the Concept will inherit when used in a structure using the Concept such as a Key Family, or Metadata Structure Definition, if not overridden by a local representation specified specifically for its use in that structure. The valid Representations are:

A CodeList, Category Scheme, Concept Scheme, Organisation Scheme, Object Type Scheme

- Date Range – this supports a data range
- Numeric Range – this supports a numeric range
- Pattern – this supports a pattern which is expressed as a string. Such expressions can be useful for content validation
- Sequence – this supports an incremental sequence of integers

3.2.5.4 Metadata Structure Definition

A Metadata Structure Definition defines the valid content of a Metadata Set in terms of the Concepts comprising the structure of the Metadata Set, how the Concepts are related in terms of their role in the Metadata Set, and the valid content of each of the Concepts when used in a Metadata Set.

A Metadata Structure Definition comprises:

- A mandatory Full Target Identifier that identifies all Identifiable Object Types that are within the scope of the definition, and which of itself may identify a specific Identifiable Object Type to which metadata may be attached

- Optional additional Partial Target Identifiers that are a sub set of the Full Target Identifier each of which identifies a specific Identifiable Object Type to which metadata may be attached

- A Report Structure which identifies the Metadata Attributes for which metadata can be reported

The Full Target Identifier specifies one or more Identifier Components, each of which identifies an Identifiable Object Type, and links this with one of:

- a Code List
- a Category Scheme
- a Concept Scheme
- an Organisation Scheme
- an Object Type Scheme

that defines the valid values for the Identifiable Object Type within the context of this Metadata Structure Definition.

The Partial Target Identifier comprises a sub set of the Identifier Components from the Full Target Identifier.

The Report Structure specifies either a flat or a hierarchical structure of one or more Metadata Attributes, each of which is either a Coded Metadata Attribute or an Uncoded Metadata Attribute

A Coded Metadata Attribute identifies a Concept in a Concept Scheme which defines the semantic and the core Representation and Type to which the Metadata Attribute must conform when metadata is reported in a Metadata Set. It can also optionally identify a Code List which contains the valid values that the Metadata Attribute can take when metadata is reported in a Metadata Set. If such a Code List is identified it overrides any core Representation that may have been defined for the Concept.

An Uncoded Metadata Attribute identifies a Concept in a Concept Scheme which defines the semantic and the core Representation and Type to which the Metadata Attribute must conform when metadata is reported in a Metadata Set. It can also optionally identify an alternative non coded Representation and Type to which the Metadata Attribute must conform when metadata is reported in a Metadata Set If such an alternative Representation and Type is identified it overrides any core Representation and Type that may have been defined for the Concept.

The Metadata Attribute is specified as being linked to one or more of the Full Target Identifier and the Partial Target Identifier. This specification defines the Identifiable Object Types for which the Metadata Attribute can be reported in a Metadata Set.

The Metadata Attribute can have additional Properties specified which define by means of a name and data type additional metadata that may be reported for the Metadata Attribute when it is reported in a Metadata Set.

3.2.5.5 Metadata Set

The Metadata Set contains metadata whose content conforms to the specification of a Metadata Structure Definition.

The Metadata Set comprises:

- One or more Attachment Keys, each of which defines the key of an object for which metadata is reported
- One or more Metadata Attribute Values, and optionally additional Attribute Property Values, which are reported for a specific object as identified by an Attachment Key

An Attachment Key is one of Full Target Key or Partial Target Key.

A Full Target Key comprises one or more Identifier Component Values, each of which is a value for the relevant Identifier Component specified in the Full Target Identifier of the Metadata Structure Definition.

A Partial Target Key comprises one or more Identifier Component Values, each of which is a value for the relevant Identifier Component specified in the Partial Target Identifier of the Metadata Structure Definition.

A Metadata Attribute Value is one of Coded Attribute Value and Uncoded Attribute Value.

A Coded Attribute Value is the value of the metadata reported for a Coded Metadata Attribute specified in the Report Structure of the Metadata Structure Definition. Additional Attribute Property Values, can be reported if this is allowed.

An Uncoded Attribute Value is the value of the metadata reported for an Uncoded Metadata Attribute specified in the Report Structure of the Metadata Structure Definition.

3.2.5.6 Structure Set and Mapping

The purpose of a Structure Set is to specify a relationship between the contents of a source Structure or source Item Scheme and the corresponding contents in the target Structure or target Item Scheme. A Structure can be any of Key Family (Data Structure Definition), Metadata Structure Definition, Dataflow Definition, Metadataflow Definition. The contents of these structures are the Dimension, Measures, and Data Attributes of the Data Structure Definition, and the Identifier Components and Metadata Attributes of the Metadata Structure Definition. An Item Scheme is one of CodeList, Category Scheme, Concept Scheme, and the Items are one of Code, Category, Concept.

The Structure Set comprises one or more of:

- A Structure Map: this identifies the source and target Structures and comprises multiple Component Maps, each of which identifies a source and target Component. Each of these Component Maps may be linked to a Code List Map in order to map the Codes if the Code List used is different for the source and target Components. The Code List Map identifies the source and target Code Lists and each Code List Map has multiple Code Maps, each of which identifies a source code and a target Code

- A Code List Map: this identifies the source and target Code Lists and each Code List Map has multiple Code Maps, each of which identifies a source code and a target Code

- A Category Scheme Map: this identifies the source and target Category Schemes and each Category Scheme Map has multiple Category Maps, each of which identifies a source code and a target Category

- A Concept Scheme Map: this identifies the source and target Concept Schemes and each Concept Scheme Map has multiple Concept Maps, each of which identifies a source code and a target Concept

Each of the various maps may have additional metadata describing the mapping by means of one or more Property, each of which defines a name, a type (type of data) and value.

3.2.5.7 Data and metadata provisioning

The purpose of data and metadata provisioning is to specify both the access to and the content of a Data Source. In the description below the term Data Source applies equally to metadata source and the term Data Provider applies equally to metadata provider.

The Dataflow Definition defines metadata and has links to objects that enable Data Sets to be registered, discovered, and processed. The Dataflow Definition has a link to a Key Family which defines the structure of any datasets published according to the Dataflow Definition.

In order to support data and metadata discovery, the Dataflow Definition and the Metadataflow Definition may also be linked to one or more Category in one or more Category Schemes such as a scheme of subject matter domains.

Data Sets and Metadata Sets are published by Data Providers. In order to manage this process the Data Provider is linked with the Dataflow Definition or Metadataflow Definition by means of a Provision Agreement.

Content Constraints can be specified for a Dataflow Definition, Metadataflow Definition, and a Provision Agreement. A Content Constraint constrains the use of a Key Family for a Data Set linked to the Dataflow Definition or Provision Agreement, or constrains the use of a Metadata Structure Definition for a Metadata Set linked to a Metadatafow Definition or Provision Agreement. A Key Family and Metadata Structure Definition may be constrained by:

1. Specifying one or more Key Sets, each of which comprises a set of Keys which, together, may be included in or excluded from the complete set of Keys that are valid for the Key Family or Metadata Structure Definition when used in a Data Set linked to the Dataflow Definition or reported for a Provision Agreement, or when used in Metadata Set when used in a Metadata Set linked to the Metadataflow Definition or reported for a Provision Agreement. Note that the complete set of Keys that are valid for a Key Family can be computed by combining all possible combinations of values

of the Dimensions, and the complete set of Keys for a Metadata Structure Definition can be computed by combining all possible combinations of values of the Identifier Components.

2. Specifying one or more Cube Region, each of which comprises a set of Member Selection which, together, may be included in or excluded from the set of values that are valid for a Dimension, Attribute, or Measure when used in the Dataflow Definition or Provision Agreement, or for an Identifier Component when used in the Metadataflow Definition or Provision Agreement. Each Member Selection specifies the constrained list of values that are valid. This list must be a complete set or a sub set of the full Representation specified for the relevant Dimension, Attribute, or Measure in the Key Family, or a complete set or a sub set of the list of valid values (Item List) specified for an Identifier Component.

In a registry based scenario, where a metadata registry is used to manage data and metadata reporting or to publish the existence of data and metadata in order to support search and discovery, the Provision Agreement and a Data Provider may also be linked to a Query Datasource. This specifies the URL of the data source, the means by which it can be accessed (a REST interface or a Web Services Interface), and a Content Constraint describing the contents of the Query Datasource.

If the Query Datasource is linked directly to a Data Provider then the resource at the URL supports data or metadata queries for all Dataflow Agreements and Metadataflow Agreements linked to the Data Provider (i.e. for all Provision Agreements linked to the Data Provider). In this scenario the Content Constraint is limited to a definition of the reference period of the data or metadata.

If the Query Datasource is linked to a specific Provision Agreement then it can support a query for data or metadata for only those Dataflow Agreements and Metadataflow Agreements linked to it. In this scenario the Content Constraint defines the contents of the queryable datasource in terms of one or both of Key Set and Cube Region, in addition to the reference period of the data or metadata.

Also in a registry based scenario a specific Data Set or Metadata Set may be registered for a Provision Agreement. This registration results in a Simple Datasource which has a URL which points to the web location of an SDMX-ML formatted file containing the data or metadata, and a Content Constraint describing the contents of the Simple Datasource (Data Set or Metadata Set) as one or both of Key Set and Cube Region, in addition to the reference period of the data or metadata.

3.2.5.8 Hierarchical Code Scheme

A Hierarchical Code Scheme describes the hierarchical relationship between Codes, where the Codes may be built into a number of different Hierarchies each serving a different purpose but based on a common theme. An example of such a Hierarchical Code Scheme is a list of Codes that support the theme of geographical location. Such a Hierarchical Code Scheme could be viewed according to many different Hierarchies. A political hierarchy would comprise an administrative regional breakdown within a country, a geographical breakdown would comprise a placing the countries in continents, and an economic breakdown might place the countries in one or more economic communities (e.g. many of the countries in “Europe” could be both a part of the EU and the OECD communities, the USA, Canada, and Mexico would be a part of the NAFTA community, many middle eastern countries would be in the OPEC community).

A Hierarchical Code Scheme comprises:

1. A set of Code Compositions each of which associates a set of Codes, which may be derived from a variety of Code Lists, to a common parent Code, itself derived from a Code List.

2. A set of Hierarchies that specify the associations between the Codes comprising the Hierarchy.

The structural rules for the Codes in the Hierarchical Code Scheme are:

3. A child Code can have more than one parent Code, but not within the same Hierarchy.

4. There can be more than one Code that has no parent Code (i.e. more than one “root node”).

5. The association between a parent Code and child Code is known as a Code Association.

6. Item Properties can be defined for the Code Association to define additional metadata concerning the association.

7. A set of Code Associations that have a common parent Code is known as a Code Composition.

8. There may be many Hierarchies (or “views”) defined, in terms of the associations between the Code Compositions comprising the Hierarchy.

9. A Hierarchy is either a Level Based Hierarchy or a Value Based Hierarchy.

10. A Value Based Hierarchy comprises a set of Code Compositions (any combination is allowable in principle).

11. A Level Based Hierarchy supports the need where formal levels need to be defined. Each Level comprises a set of Code Composition. The constraint of a Level Based Hierarchy is that each Code in a Level has one and only one parent in the superior Level.

3.2.5.9 Process and Transitions

In any system that processes data and metadata the system itself is a series of processes and in each of these processes the data or metadata may undergo a series of transitions. This is particularly true of its path from raw data to published data and metadata.

A Process scheme comprises:

1. A set of Process Steps each of which can be broken down in a hierarchy to further Process Steps.

2. A set of Transitions, each of which links precisely one input Process Step to precisely one output Process Step.

3. Each ProcessStep can reference zero or more objects, such as Dataflow Definitions, Hierarchy and Code Lists, as input, and zero or more similar (to the input process) objects as output.

4. The computation performed by a ProcessStep is optionally described by an ExpressionNode, which can represent an arbitrary expression involving any identifiable objects.

5. The ProcessStep could also be described textually in multiple languages.

6. The Transition controls the execution of ProcessSteps from source ProcessStep to target ProcessStep based on the evaluation of a condition defined in an ExpressionNode in a Transformation Scheme.

7. The Transition can be used for looping and conditional execution of ProcessSteps.

3.2.5.10 Transformations and Expressions

Transformations and expressions track the derivation of data. It is similar in concept to lineage in data warehousing – i.e. how data is acquired or derived.

A Transformation allows the identification and documentation of the functions performed (these will normally be automated, program functions), as well as defining structures that support a syntax neutral expression "grammar" that can specify the functions at a granular level such that a program can "read" the metadata and compose the function required in whatever computer language is appropriate.

The Transformation Scheme comprises:

1. Expression Nodes, each of which can be broken down in a hierarchy to further Expression Nodes.

2. The Expression Node must be of a particular Type which identifies the expected representation of the result of the expression.

3. The Expression Node may take the value of Constant.

4. The Expression Node may reference an object.

5. The Expression Node cannot both take the value of Constant and reference an object.

6. The Expression Node may link to a maximum of one Operator in a known scheme of Operators.

7. The Operator may have Operands each of which is a mathematical operator which, together with the Operator define the contents of an expression.

8. The sequence of the Operands defines the number and ordering of formal parameters that the Operator takes.

9. Each child Expression Node of the parent Expression Node that has a link to an Operand, corresponds to each of the formal parameters of the Operands in the correct sequence.

3.3 SDMX-ML and SDMX-EDI: Comparison of Expressive Capabilities and Function

SDMX offers several equivalent formats for describing data and structural metadata, optimized for use in different applications. Although all of these formats are derived directly from the SDM-IM, and are thus equivalent, the syntaxes used to express the model place some restrictions on their use. Also, different optimizations provide different capabilities. This section describes these differences, and provides some rules for applications which may need to support more than one SDMX format or syntax. This section is constrained to the Key Family and the Date Set.

3.3.1 Format Optimizations and Differences

The following section provides a brief overview of the differences between the various SDMX formats.

Structure Definition

- The SDMX-ML Structure Message supports the use of annotations to the structure, which is not supported by the SDMX-EDI syntax.

- The SDMX-ML Structure Message allows for the structures on which a key family depends – that is, codelists and concepts – to be either included in the message or to be referenced by the message containing the data structure definition. XML syntax is designed to leverage URIs and other Internet-based referencing mechanisms, and these are used in the SDMX-ML message. This option is not available to those using the SDMX-EDI structure message.

Validation

- SDMX-EDI – as is typical of EDIFACT syntax messages – leaves validation to dedicated applications ("validation" being the checking of syntax, datatyping, and adherence of the data message to the structure as described in the structural definition.)

- The SDMX-ML Generic Data Message also leaves validation above the XML syntax level to the application.

- The SDMX-ML Compact Data and Cross-Sectional Data Messages will allow validation of XML syntax and datatyping to be performed with a generic XML parser, and enforce agreement between the structural definition and the data to a moderate degree with the same tool.

- The SDMX-ML Utility Data Message leverages a generic XML parser to perform the most complete degree of validation at all levels. (Note that dependencies between and among coded dimension and attribute values are not captured in the structural definition, and therefore must always be validated by the application.)

Update and Delete Messages and Documentation Messages

- The SDMX-ML Utility Data Message must always provide a complete update of the data set, ands thererefore cannot be used to perform deletions. Further, it cannot be used to send documentation without the corresponding data. All other SDMX data messages allow for both delete messages and messages consisting of only data or only documentation.

Character Encodings

- All SDMX-ML messages use the UTF-8 encoding, while SDMX-EDI uses the ISO 8879-1 character encoding. There is a greater capacity with UTF-8 to express some character sets (see the “APPENDIX: MAP OF ISO 8859-1 (UNOC) CHARACTER SET (LATIN 1 OR “WESTERN” in the document “SYNTAX AND DOCUMENTATION VERSION 2.0”. Many transformation tools are available which allow XML instances with UTF-8 encodings to be expressed as ISO 8879-1-encoded characters, and to transform UTF-8 into ISO 8879-1. Such tools should be used when transforming SDMX-ML messages into SDMX-EDI messages and vice-versa.

Data Typing

The XML syntax and EDIFACT syntax have different data-typing mechanisms. The section below provides a set of conventions to be observed when support for messages in both syntaxes is required. For more information on the SDMX-ML representations of data, see below.

3.3.2 Data Types

The XML syntax has a very different mechanism for data-typing than the EDIFACT syntax, and this difference may create some difficulties for applications which support both EDIFACT-based and XML-based SDMX data formats. This section provides a set of conventions for the expression in data in all formats, to allow for clean interoperability between them.

It should be noted that this section does not address character encodings – it is assumed that conversion software will include the use of transformations which will map between the ISO 8879-1 encoding of the SDMX-EDI format and the UTF-8 encoding of the SDMX-ML formats.

Note that the following conventions may be followed for ease of interoperation between EDIFACT and XML representations of the data and metadata. For implementations in which no transformation between EDIFACT and XML syntaxes is foreseen, the restrictions below need not apply.

1. Identifiers are:

    Maximum 18 characters;

    Any of A..Z (upper case alphabetic), 0..9 (numeric), _ (underbar);

    The first character is alphabetic.

2. Names are:

   Maximum 70 characters.

   From ISO 8859-1 character set (including accented characters)

3. Descriptions are:

   Maximum 350 characters;

   From ISO 8859-1 character set.

4. Code values are:

   Maximum 18 characters;

   Any of A..Z (upper case alphabetic), 0..9 (numeric), _ (underscore), / (solidus, slash), = (equal sign), - (hyphen);

However, code values providing values to a dimension must use only the following characters:
A..Z (upper case alphabetic), 0..9 (numeric), _ (underscore)

5. Observation values are:

   Decimal numerics (signed only if they are negative);

   The maximum number of significant figures is:

   - 15 for a positive number
   - 14 for a positive decimal or a negative integer
   - 13 for a negative decimal

   Scientific notation may be used.

6. Uncoded statistical concept text values are:

   Maximum 1050 characters;

   From ISO 8859-1 character set.

7. Time series keys:

In principle, the maximum permissible length of time series keys used in a data exchange does not need to be restricted. However, for working purposes, an effort is made to limit the maximum length to 35 characters; in this length, also (for SDMX-EDI) one (separator) position is included between all successive dimension values; this means that the maximum length allowed for a pure series key (concatenation of dimension values) can be less than 35 characters. The separator character is a colon (“:”) by conventional usage.

3.4 SDMX-ML and SDMX-EDI Best Practices

3.4.1 Reporting and Dissemination Guidelines

3.4.1.1 Central Institutions and Their Role in Statistical Data Exchanges

Central institutions are the organisations to which other partner institutions "report" statistics. These statistics are used by central institutions either to compile aggregates and/or they are put together and made available in a uniform manner (e.g. on-line or on a CD-ROM or through file transfers). Therefore, central institutions receive data from other institutions and, usually, they also "disseminate" data to individual and/or institutions for end-use. Within a country, a NSI or a national central bank (NCB) plays, of course, a central institution role as it collects data from other entities and it disseminates statistical information to end users. In SDMX the role of central institution is very important: every statistical message is based on underlying structural definitions (statistical concepts, code lists, key families) which have been devised by a particular agency, usually a central institution. Such an institution plays the role of the reference "structural definitions maintenance agency" for the corresponding messages which are exchanged. Of course, two institutions could exchange data using/referring to structural information devised by a third institution.

Central institutions can play a double role:

- collecting and further disseminating statistics;
- devising structural definitions for use in data exchanges.

3.4.1.2 Defining Data Structure Definitions (Key Families)

The following guidelines are suggested for building a data structure definition. However, it is expected that these guidelines will be considered by central institutions when devising new data structure definition definitions.

- ***Avoid dimensions that are not appropriate for all the time series in the data structure definition.*** If some dimensions are not appropriate for some series then consider moving these series to a new data structure definition in which these dimensions are dropped from the key structure.

- ***Avoid composite dimensions.*** Each dimension should correspond to a single characteristic of the data, not to a combination of characteristics.

- ***Avoid creating a new code list where one already exists.*** It is highly recommended that structural definitions and code lists be consistent with internationally agreed standard methodologies, wherever they exist, e.g., System of National Accounts 1993; Balance of Payments Manual, Fifth Edition; Monetary and Financial Statistics Manual; Government Finance Statistics Manual, etc. When setting-up a new data exchange, the following order of priority is suggested when considering the use of code lists:
    - international standard code lists;

- international code lists supplemented by other international and/or regional institutions;
- standardised lists used already by international institutions;
- new code lists agreed between two international or regional institutions;
- new specific code lists.

The same code list can be used for several statistical concepts, within a data structure definition or across key families.

- **Data Structure Definition.** The following items have to be specified by a structural definitions maintenance agency when defining a new data structure definition:

- Data structure definition identification:
  - data structure definition identifier
  - data structure definition name

- A list of metadata concepts assigned as dimensions of the data structure definition. For each:
  - metadata concept identifier
  - metadata concept name
  - ordinal number of the dimension in the key structure
  - code list identifier if the representation is coded
  - A list of statistical concepts assigned as attributes for the data structure definition. For each:
  - statistical concept identifier
  - statistical concept name
  - code list identifier if the concept is coded
  - assignment status: mandatory or conditional
  - attachment level
  - maximum text length for the uncoded concepts
  - maximum code length for the coded concepts

- A list of the code lists used in the data structure definition. For each:

  - code list identifier
  - code list name
  - code values and descriptions

- Definition of data flow definitions. Two (or more) partners performing data exchanges in a certain context need to agree on:
  - the list of data set identifiers they will be using;
  - for each data flow:
    - its content and description
    - the relevant key family definition

- **Mandatory attributes**. Once the key structure of a data structure definition has been decided, then the set of mandatory attributes of this data structure definition has to be defined. In general, some statistical concepts are necessary across all key families to qualify the contained information. Examples of these are:
  - Reference area
  - Frequency (always a dimension)
  - A descriptive title (see also comment below)
  - Collection (e.g. end of period, averaged or summed over period)
  - Unit (e.g. currency of denomination)
  - Unit multiplier (e.g. expressed in millions)
  - Availability (which institutions can a series become available to)
  - Decimals (i.e. number of decimal digits used in a time series)
  - Observation Status (e.g. estimate, provisional, normal)

Therefore, those concepts which are not dimensions within a data structure definition have to be present in that data structure definition as mandatory attributes. Moreover, additional attributes may be considered as mandatory when a specific data structure definition is defined.

3.4.1.3 Time and Frequency

While it is not required that a data structure definition designed to provide only cross-sectional presentations have the concept of Time associated with the data it describes, this is a very unusual case. For all key families which use the Time concept, it is strongly recommended that the concept Frequency also be used in the data structure definition as a dimension. While this may not seem to be important for some publishers and disseminators of statistics, the lack of a Frequency can create difficulties with many systems in the presentation and processing of statistics.

Conventionally, Frequency is the first dimension in the key. Frequency is typically a value from the following list, although it may be necessary to make additions to this list for specific uses:

A Annual
B Business (often not supported)
D Daily
E Event (often not supported)
H Semi-annual
M Monthly
Q Quarterly
W Weekly

For reasons related to backward compatibility with existing systems, there is a corresponding concept of “TIME_FORMAT”, which is needed in the formats to describe how time is formatted. TIME_FORMAT is included in the data structure definition as a required series-level attribute. However, when the data structure definition definition is serialised as SDMX-EDI the TIME_FORMAT is declared as a dimension (which, in sequence, is placed immediately after the time dimension), and not as an attribute. In the SDMX-ML rendering it is declared as defined in the data structure definition (i.e. as a series-level attribute).

In the XML representation, the TIME_FORMAT is usually a value taken from the following list (meanings as defined in ISO 8601):

P1Y – Annual
P6M – Semi-annual
P3M – Quarterly
P1M – Monthly
P7D – Weekly
P1D – Daily
PT1M – Minutely

For SDMX-EDI, there is a syntax-specific list of relevant codes taken from the code list associated with the UN/EDIFACT TDID data element 2379 - Date or time or period format code.

Applications processing time ranges in either SDMX-EDI or SDMX-ML must know how to iterate time such as knowledge of leap years and 53 week years. For the latter the calculation of weeks is according to ISO 2017 (simply put, this states that the first week in a year is the week that contains the first Thursday of the year).

Time ranges are expressed in SDMX-ML simply by omitting the time value from the observation for all except the first observation (supported only by the CompactData message). In SDMX-EDI the time period is expressed as a time range by declaring begin and end periods. This can be used to validate whether all the observations are present for the time series. As the SDMX-ML declares only the beginning period, it is recommended that the time period is also present in the last observation, so that a similar validation can be performed.

Additional attributes may be necessary to specify such items as whether the time period specified is the beginning or end of period, etc. For example, a monthly series may contain observations taken at the beginning, or the middle, or end of the month, and it may be important for this metadata to be attached as an attribute.

If a data structure definition which uses time does not use the concept of Frequency, then it cannot use certain specific features of the formats (such as expressing time ranges in SDMX-EDI and the CompactData message in SDMX-ML.)

3.4.1.4 Exchanging Attributes

3.4.1.4.1 Attributes on series, sibling and data set level

- *Static properties*.
    - Upon creation of a series the sender has to provide to the receiver values for all mandatory attributes . In case they are available, values for conditional attributes should also be provided. Whereas initially this information may be provided by means other than SDMX-ML or SDMX-EDI messages (e.g. paper, telephone) it is expected that partner institutions will be in a position to provide this information in SDMX-ML or SDMX-EDI format over time.
    - A centre may agree with its data exchange partners special procedures for authorising the setting of attributes' initial values.
    - Attribute values at a data set level are set and maintained exclusively by the centre administrating the exchanged data set.
- *Communication of changes* to the centre.
    - Following the creation of a series, the attribute values do not have to be reported again by senders, as long as they do not change.
    - Whenever changes in attribute values for a series (or sibling group) occur, the reporting institutions should report either all attribute values again (this is the recommended option) or only the attribute values which have changed. This applies both to the mandatory and the conditional attributes . For example, if a previously reported value for a conditional attribute is no longer valid, this has to be reported to the centre.
    - A centre may agree with its data exchange partners special procedures for authorising modifications in the attribute values.
- Communication of observation level attributes “observation status” ”, "observation confidentiality", "observation pre-break"
    - In SDMX-EDI, the observation level attribute “observation status” is part of the fixed syntax of the ARR segment used for observation

reporting. Whenever an observation is exchanged, the corresponding observation status must also be exchanged attached to the observation, regardless of whether it has changed or not since the previous data exchange. This rule also applies to the use of the SDMX-ML formats, although the syntax does not necessarily require this.

- o If the “observation status” changes and the observation remains unchanged, both components would have to be reported.

- o For key families having also the observation level attributes “observation confidentiality” and "observation pre-break" defined, this rule applies to these attribute as well: if an institution receives from another institution an observation with an observation status attribute only attached, this means that the associated observation confidentiality and pre-break observation attributes either never existed or from now they do not have a value for this observation.

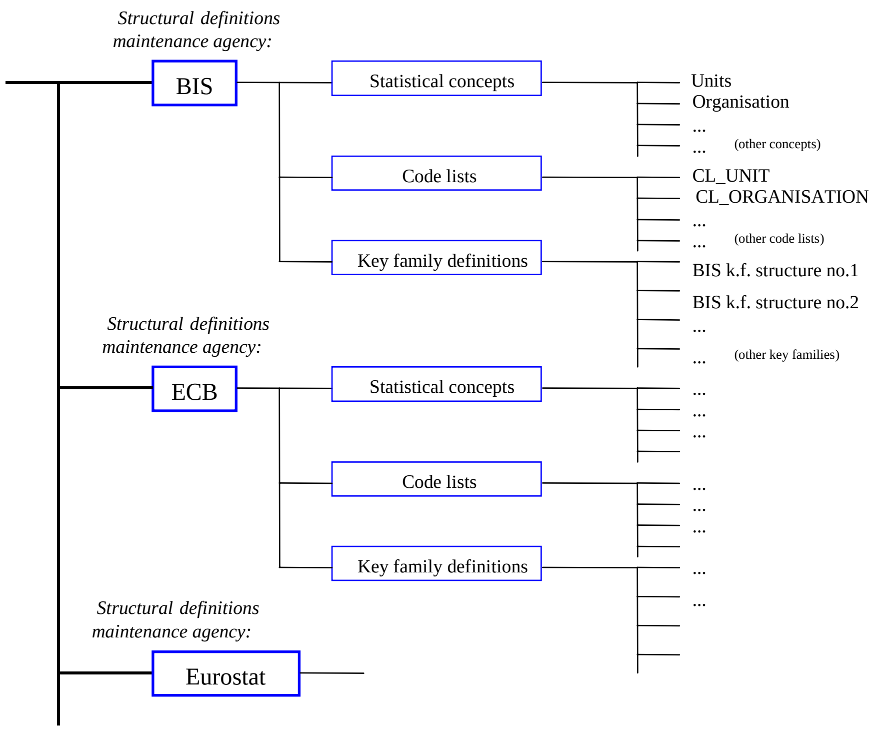
3.4.2 Best Practices for Batch Data Exchange

Batch data exchange – the exchange and maintenance of entire databases between counterparties – is an activity that often employs SDMX-EDI formats, and might also use the SDMX-ML CompactDataMessage. The following points apply equally to both formats.

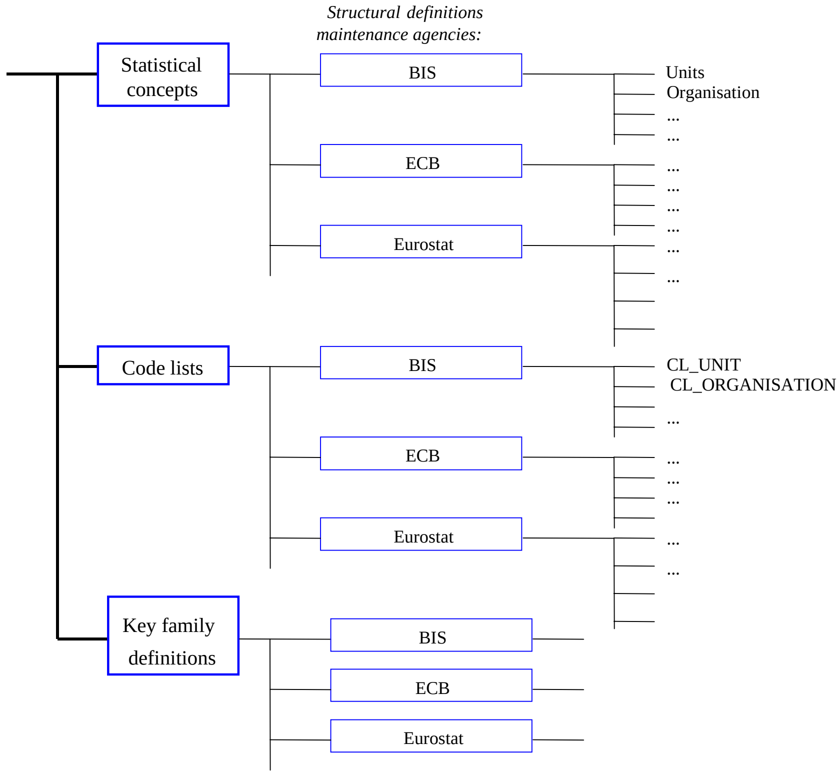
3.4.2.1 More Than One Central Institution Involved in a Data Exchange

In the paragraph discussing the role of central institutions, it was mentioned that though, usually, a central institution administrates the exchange of data sets based on the structural definitions it devises. There may be other cases in which a third institution's structural definitions could be used in a data exchange. In this case, the central institution administrating this data set(s) should take care (possibly in co-operation with the corresponding structural definitions maintenance agency) that the necessary structural definitions become known to all data exchange partners involved and that the corresponding SDMX structural definition messages are properly maintained and, if necessary, appropriately updated.

SDMX gives the possibility to partner institutions to design generic data exchange systems which can take into account the role of central institutions in devising structural definitions. In principle, each institution should design its system in such a way that could cope with an environment in which more than one structural definitions maintenance agency could exist. For example, the following figures describe alternative ways to organise structural definitions assuming the existence of three central institutions (e.g. BIS, ECB, Eurostat). In practice, more central institutions could be envisaged and, therefore, more central branches in the tree; including possibly even the home institution (e.g. a central bank or statistical institute) if the home institution plays a role in "devising" structural definitions within a user community.



1198 **Figure 1: Schematic of structural definitions maintenance agencies**



1199 **Figure 2: Alternative schematic of structural definitions maintenance agencies**

3.4.2.2 Positioning of the Dimension "Frequency"

The position of the “frequency” dimension is unambiguously identified in the data structure definition definition. Moreover, most central institutions devising structural definitions have decided to assign to this dimension the first position in the key structure. This facilitates the easy identification of this dimension, something that it is necessary to frequency's crucial role in several database systems and in attaching attributes at the sibling group level.

3.4.2.3 Identification of Data Structure Definitions (Key Families)

In order to facilitate the easy and immediate recognition of the structural definition maintenance agency that defined a data structure definition, most central institutions devising structural definitions use the first characters of the data structure definition identifiers to identify their institution: e.g. BIS_MACRO, EUROSTAT_BOP_01, ECB_BOP1, etc.

3.4.2.4 Identification of the Data Flows

In order to facilitate the easy and immediate recognition of the institution administrating a data flow definitions, many central institutions prefer to use the first characters of the data flow definition identifiers to identify their institution: e.g. BIS_MACRO, ECB_BOP1, ECB_BOP1T, etc. Note that in GESMES/TS the Data Set plays the role of the data flow definition (see The words in Italics refer to classes in the SDMX-IM.

The statistical information in SDMX is broken down into two fundamental parts - structural metadata (comprising the *KeyFamily*, and associated *Concept*s and *Code Lists*) – see Framework for Standards -, and observational data (The *DataSet*). This is an important distinction, with specific terminology associated with each part. Data - which is typically a set of numeric observations at specific points in time - is organized into data sets (*DataSet*) These data sets are structured according to a specific Data Structure Definition (*KeyFamily*), and are described in the data flow definition (*DataflowDefinition*) The Data Structure Definition describes the metadata that allows an understanding of what is expressed in the data set, whilst the data flow definition provides the identifier and other important information (such as the periodicity of reporting) that is common to all of its component data sets.

Note that the role of the Data Flow (called *DataflowDefintion* in the model) and Data Set is very specific in the model, and the terminology used may not be the same as used in all organisations, and specifically the term Data Set is used differently in SDMX than in GESMES/TS. Essentially the GESMES/TS term “Data Set” is, in SDMX, the “Dataflow Definition” whist the term “Data Set” in SDMX is used to describe the “container” for an instance of the data.
Data Set0).

3.4.2.5 Special Issues

3.4.2.5.1 "Frequency" related issues

- ***Special frequencies.*** The issue of data collected at special (regular or irregular) intervals at a lower than daily frequency (e.g. 24 or 36 or 48 observations per year, on irregular days during the year) is not extensively discussed here. However, for data exchange purposes:

- such data can be mapped into a series with daily frequency; this daily series will only hold observations for those days on which the measured event takes place;

- if the collection intervals are regular, additional values to the existing frequency code list(s) could be added in the future.

- ***Tick data.*** The issue of data collected at irregular intervals at a higher than daily frequency (e.g. tick-by-tick data) is not discussed here either. However, for data exchange purposes, such series can already be exchanged in the SDMX-EDI format by using the option to send observations with the associated time stamp.

4 GENERAL NOTES FOR IMPLEMENTORS

This section discusses a number of topics other than the exchange of data sets in SDMX-ML and SDMX-EDI. Supported only in SDMX-ML, these topics include the use of the reference metadata mechanism in SDMX, the use of Structure Sets and Reporting Taxonomies, the use of Processes, a discussion of time and data-typing, and some of the conventional mechanisms within the SDMX-ML Structure message regarding versioning and external referencing.

This section does not go into great detail on these topics, but provides a useful overview of these features to assist implementors in further use of the parts of the specification which are relevant to them.

4.1 Reference Metadata Reporting

Reference metadata is, in technical terms, that metadata which is transmitted as the primary content of a report. In fact, it may duplicate some of the attributes which are associated with the SDMX data formats. The difference is, of course, that the reference metadata is reported independent of the data, in a set of formats which are distinct from the data formats.

The reference metadata mechanism in SDMX-ML is parallel to the data mechanism: on the data side, a key family (data structure definition) which can be used to structure data in a generic data format, or in one of a set of three different key-family-specific data formats (Utility, Compact, and Cross-Sectional). Each format is represented with an XML Schema which is created according to the SDMX standard.

On the reference metadata side, there is a parallel mechanism: a metadata structure definition is created, which can be used to structure metadata reports. There are only two types of metadata reports in SDMX-ML: a generic report (GenericMetadata.xsd), which can be used for any metadata report, according to the contents of a metadata structure definition, and a metadata-structure-definition-specific metadata report, which is derived from the metadata structure definition according to rules specified in the SDMX Standard.

Metadata structure definitions are structured in a fairly simple fashion – because reference metadata is less structured than aggregate statistical data, there is a simpler structure for it, either a flat list or a hierarchy. A metadata structure definition takes a set of concepts from a concept scheme, organizes them into a hierarchy, states whether their reporting is required or optional, and assigns representations to them. It also performs one other function: it describes the subject or subjects of the reported metadata.

In the SDMX-ML Structure Message, the XML structures are organized as follows:

- Concepts are expressed using the ConceptScheme element
- Target identifiers (the subjects of the metadata report) are described in the first section of the metadata report structure
- The concepts are arranged into a flat list or hierarchy in the Report Structure, and associated with the target identifiers

There are a few subtleties in this mechanism. Target identifiers are similar to the keys in data structure definitions – they are composite identifiers which explain which specific set of metadata is being reported. The difference is that where data keys are composed of concepts ("dimensions"), the reference metadata target identifiers are composed of object types in the model. This is non-obvious until you see an example. Let's say we want to report metadata about an organization that disseminates data. In the SDMX Information Model, that organization would be described as a "data provider". Thus, we could have a single field in our target identifier which contained the ID of a data provider, telling us which data provider was the subject of a given metadata report. A list of all the objects in SDMX-ML which can be used to compose target identifiers is found in the `structure:ObjectIDType` definition.

Some target identifiers are composite – that is, they have more than a single object in them. Take the example where we want to report reference metadata for the data disseminated in each category of some categorization scheme by a specific data provider. In this case, we would have two object types in our target identifier: the data provider and the category within the category scheme.

If the category scheme had categories for "Population Statistics" and "Education Statistics" (as two examples), and our list of data providers has "Organisation A" and Organisation B" in it, then we have four possible target identifiers against which we can report reference metadata: Population Statistics by Organisation A, Population Statistics by Organisation B, Education Statistics by Organisation A, and Education Statistics by Organisation B.

It is also possible to report reference metadata for a subset of the full set of objects in the target identifier – thus, we have Full Target Identifiers (with a value for all objects) and Partial Target Identifiers (with values for a subset of the objects). This is similar to the idea of Groups and Partial Keys on the data side.

The contents of reference metadata reports are based on concepts, reported as attributes. This is very similar to the attributes in the SDMX-ML data mechanism. Each attribute is associated with a concept, and a representation is given for that concept, along with an indication of whether it is required or optional. The concepts can be assigned default representations in the concept scheme – these can be used or over-written as desired in the metadata structure definition. Representations are the same as those on the data side, which are described more completely below. Typically, however, reference metadata consists of textual information (that is, it is represented as a fairly long string).

Concepts can be arranged in a presentational hierarchy, or can be reported as a flat list. Thus, if I have a concept of CONTACT which consists of the sub-concepts NAME, E-MAIL, ADDRESS, and PHONE_NUMBER, I can arrange these in a simple hierarchy in which CONTACT is made up of the other subordinate concepts.

Note that the metadata attributes can be *any concepts whatsoever*. This makes the reference metadata mechanism in SDMX-ML extremely powerful as a generic way to express many sorts of metadata which is not otherwise formally described in SDMX-ML.

4.2 Reporting Taxonomies and Structure Sets

There are two constructs in SDMX-ML which serve as grouping mechanisms above the level of the dataset/dataflow and metadata set/metadata flow. These are the Reporting Taxonomy and the Structure Set.

Reporting Taxonomies are a specialized type of category scheme which is used to organize related data flows and metadata flows into coherent “reports”. In the real world, many types of “reports” are made up of data and metadata from a variety of data sets and metadata sets. The Reporting Taxonomy is the way in which these groups are formally organized in SDMX.

Reporting Taxonomies are simple – they are just a recursive set of Categories, which, in SDMX, contain references to data flows and metadata flows. All of the references to flows needed for the Reporting Taxonomy may be immediate children of the Reporting Taxonomy, or they may be organized into a flat grouping or hierarchy using the subordinate Categories of the Reporting Taxonomy.

Because it is quite common for only specific subsets of an entire dataset or metadata set to be used in a Reporting Taxonomy, it may be necessary to create dedicated data flows and metadata flows, whose constraints describe exactly the contents of each dataset or metadata set in the flow that are needed.

The Reporting Taxonomy is thus a description of the structure of a group of data and metadata, brought together to form the contents of any kind of report which is issued on a regular basis.

Structure Sets serve a more complex function. They are able to express mappings between codelists (including hierarchical codelists), key families (data structure definitions) and metadata structure definitions, concept schemes, category schemes, and organization schemes. Because they have all of these capabilities, it is possible to use Structure Sets to describe the relationships within some types of data cubes, especially where these have data which do not all share the same dimensionality.

In the simplest case, Structure Sets can be used to express a simple mapping between two codelists. All of the mappings in the Structure Set are pair-wise mappings – that is, if A and B and C are all equal, A is equated to B, and B is equated to C. Logically, A is equal to C, but that need never be explicitly stated. To help organize the mappings, each mapping at the most granular level (that is, of two codes, or two components, or two concepts, or two categories) can be given an “Alias”. The Alias becomes a single name which can be used to refer to all of the equal, granular members. Thus, if I have a code “MX” which is equal to a code “Mex” which is equal to a code “MXCO”, I can assign an alias to each pair-wise mapping: “MX” = “Mex” (Alias is “MEX”); “Mex” = “MXCO” (Alias is “MEX”). Thus, whenever I want to refer to any one of the set of equal codes, I can use the alias to do so.

Structure Sets can also be used to indicate relationships between the structure definitions for both data and metadata. This feature can be used in different ways: as a means of describing the similarities and differences between a group of key families (data structure definitions) or metadata structure definitions, including inheritance relationships; and as a way of capturing the transformation between a key family and a metadata structure definition and the data/metadata they structure.

In the first case, it is possible to describe how key families relate by mapping their components to each other. If we have two key families which are identical except that they may have slightly different names, and one of them has an additional dimension, I can express this either by having maps which assert the relationships between each equivalent dimension and attribute, or I can create an “abstract” key family which has all of the commonalities in it, and is then inherited into two “concrete” key families which differ in certain respects. I also have the ability to map the corresponding representations (codelists) to each other for similar dimensions or attributes, and to reference this mapping from the component map. This mechanism can be used to describe the common dimensionality of the various sets of data found in a data cube, where the dimensionality is not the same across all cube regions.

In the second case, I can take a key family and map its components against the metadata attributes reported in a reference metadata structure definition (or vice versa). If I have an attribute at the series level of my key family, I can map that to any of the reported concepts in my metadata report which correspond to it. There is even the ability to express how a coded value should be transformed – as the code, as its name, or as its description. While this facility is somewhat limited in terms of all possible representations, it does provide a way of cross-walking attribute values between data sets and metadata sets, where they have attributes in common.

Structure Sets are a powerful way of expressing the relationships between the structures in the SDMX Information Model in some useful ways. For full details, please read the documentation in the SDMX-ML Structure namespace module.

4.3 Processes

Processes in SDMX-ML are a way of documenting the steps of a statistical process. This is a very generic mechanism, and it is not extremely detailed. It is not intended to support process automation in the same way as some other process description and workflow standards. Instead, it is meant to allow for the description and documentation of processes at a level suitable for the attachment of reference metadata.

Thus, if I have several steps in my process, I can create a process step for each one, and then report specific metadata concepts against each, as appropriate. Because the process step is an object in the SDMX model, it can serve as a target identifier for reference metadata reports.

4.4 Time and Representations

There are several different representations in SDMX-ML, taken from XML Schemas and common programming languages. The table below describes the various representations which are found in SDMX-ML, and their equivalents.

SDMX-ML Data Type	XML Schema Data Type	.NET Framework Type	Java Data Type
`String`	`xsd:string`	`System.String`	`java.lang.String`
`Big Integer`	`xsd:integer`	`System.Decimal`	`java.math.BigInteger`
`Integer`	`xsd:int`	`System.Int32`	`int`
`Long`	`xsd.long`	`System.Int64`	`long`
`Short`	`xsd:short`	`System.Int16`	`short`
`Decimal`	`xsd:decimal`	`System.Decimal`	`java.math.BigDecimal`
`Float`	`xsd:float`	`System.Single`	`float`

SDMX-ML Data Type	XML Schema Data Type	.NET Framework Type	Java Data Type
Double	xsd:double	System.Double	double
Boolean	xsd:boolean	System.Boolean	boolean
DateTime	xsd:dateTime	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Time	xsd:time	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Date	xsd:date	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Year, Month, Day, MonthDay, YearMonth	xsd:g*	System.DateTime	javax.xml.datatype.XMLGregorianCalendar
Duration	xsd:duration	System.TimeSpan	javax.xml.datatype.Duration
URI	xsd:anyURI	System.Uri	Java.net.URI or java.lang.String

There are also a number of SDMX-ML data types which do not have these direct correspondences, often because they are composite representations:

- Timespan (start DateTime + Duration)
- Count (Integer with an interval of “1”)
- InclusiveValueRange (startValue and endValue)
- ExclusiveValueRange (startValue and endValue)
- Incremental (Double with a specified interval)
- ObservationalTimePeriod (a union type of Date, Time, DateTime, and a set of codes for common periods – see below).

Because ObservationalTimePeriod has a very specific use in SDMX-ML, it is a union type which departs from ISO 8601 in one particular – it includes some coded representations for specific statistical conventions. It is assumed that ObservationalTimePeriod is used in conjunction with a frequency, and so it relies on additional information in the definition of the coded values for frequency (such as when a quarter begins and ends). The format can be any of the basic 8601-compliant formats, but also provides the additional short list:

Quarters:
YYYY-Q1
YYYY-Q2
YYYY-Q3
YYYY-Q4

Bi-annual (Semi-annual):

YYYY-B1
YYYY-B2

Tri-annual:

YYYY-T1
YYYY-T2
YYYY-T3

Weekly :

YYYY-W1 to YYYY-W52

Other periods are conventionally specified using their starting period (that is, a time period referring to a decade would have its start year given in ISO 8601 format, with the Frequency indicating that the period is a decade.)

Data types also have a set of facets:

isSequence = true | false (indicates a sequentially increasing value)
minLength = Integer (# of digits)
maxLength = Integer (# of digits)
startValue = Value (of specified type)
endValue = Value (of specified type)
interval = Double
timeInterval = Duration
decimal = Integer (# of digits to right of decimal point)
pattern = (a regular expression, as per W3C XML Schema)

Note that codelists may also have textual representations assigned to them, in addition to their enumeration of codes.

4.5 Versioning and External Referencing

Within the SDMX-ML Structure Message, there is a pattern for versioning and external referencing which should be pointed out. The identifiers are qualified by their version numbers – that is, an object with an Agency of "A", and ID of "X" and a version of "1.0" is a different object than one with an Agency of "A', an ID of "X", and a version of "1.1".

The production versions of identifiable objects/resources are assumed to be static – that is, they have their isFinal attribute set to 'true". Once in production, and object cannot change in any way, or it must be versioned. For cases where an object is not static, the isFinal attribute must have a value of "false", but non-final objects should not be used outside of a specific system designed to accommodate them. For most purposes, all objects should be declared final before use in production.

This mechanism is an "early binding" one – everything with a versioned identity is a known quantity, and will not change. It is worth pointing out that in some cases relationships are essentially one-way references: an illustrative case is that of Categories. While a Category may be referenced by many dataflows and metadata flows, the addition of more references from flow objects does not version the Category. This is because the flows are not properties of the Categories – they merely make references to it. If the name of a Category changed, or its sub-Categories changed, then versioning would be necessary.

Versioning operates at the level of versionable and maintainable objects in the SDMX information model. If any of the children of objects at these levels change, then the objects themselves are versioned.

One area which is much impacted by this versioning scheme is the ability to reference external objects. With the many dependencies within the various structural objects in SDMX, it is useful to have a scheme for external referencing. This is done at the level of maintainable objects (key families, codelists, concept schemes, etc.) In an SDMX-ML Structure Message, whenever an “isExternalReference” attribute is set to true, then the application must resolve the address provided in the associated “uri” attribute and use the SDMX-ML Structure Message stored at that location for the full definition of the object in question. Alternately, if a registry “urn” attribute has been provided, the registry can be used to supply the full details of the object.

Because the version number is part of the identifier for an object, versions are a necessary part of determining that a given resource is the one which was called for. It should be noted that whenever a version number is not supplied, it is assumed to be “1.0”. (The “x.x” versioning notation is conventional in practice with SDMX, but not required.)

5 THE REGISTRY/REPOSITORY

5.1 Registry Functionality

The purpose of the registry/repository is twofold:

1. It acts as a repository for structural metadata – these metadata can be stored and retrieved from the repository.

2. It acts as a source for metadata describing data and metadata sets to aid the discovery of data and metadata.

3. Whilst the registry interfaces and the conceptual model, explanation and examples are documented elsewhere, this section describes in brief the support that the registry provides to a user of the SDMX standards.

4. A user may undertake one or more of the following:

5. Submit structural metadata (e.g. data structure definition and metadata structure definitions) for storage in the repository.

6. Submit data and metadata provisioning information for storage in the repository (e.g. lists of data and metadata flows, and data providers).

7. Submit provisioning agreements for storage in the repository. These define which organisations provide data, to whom, and when – these support the “pull” scenario.

8. Register data and metadata sets against these provision agreements – this registration process causes the registry to index relevant parts of the data and metadata so that data and metadata can be discovered by query applications. It also causes a notification to be sent to organisations that have requested to be informed (using the subscription service) of newly registered data or metadata, if these data or metadata meet the subscription criteria of the organisation.

9. Query for data and/or metadata – the query process returns, amongst other metadata, the URL where the data and metadata are located and the type of service

offered by the URL (e.g. a query service on a database, a file containing the SDMX-ML rendering of the data or metadata).

10. Query for structural metadata.

11. Submit subscriptions to registry objects, so that any update on that registry object will generate a notification, sent to the subscriber as via HTTP, HTTPS, or as an e-mail. The contents of the notification are in the form of an SDMX-ML Notification message.

(1) and (2) and (6) above support any type of reporting scenario. Organisations that use the "pull" reporting scenario need the functionality described in (3)-(5) and (7) above. The functionality supporting the "pull" scenario is also relevant to applications that offer general data and metadata search and retrieve functions over the web.

5.2 Deployment of Registries

Although a technical specification such as this one refers to "the" registry, this is not an indication that there is a single, centralized registry. In terms of the specification, there is a standard set of functions and interfaces which any conformant registry must support, but the specification says nothing about how a registry might be used.

Below are some anticipated registry-based scenarios, indicating how the registry specification could be usefully implemented.

To Support Application Integration within an Organisation:

In this deployment scenario, the registry functions on an intranet (or possibly a secure extranet) in support of a data warehouse, metadata repository, or similar application. It provides a central repository full of standardized metadata, which allows for the loose coupling of applications. The standard technical problem in this type of integration is the wide range of formats and protocols employed by different parts of a large organization. The use of a registry and generalized tools working off of the SDMX Technical standards can materially ease the challenges of data and metadata integration.

To Support a Statistical Domain or Community of Interest:

This scenario involves a single, possibly inter-organisation group which operates an SDMX registry to facilitate the sharing of data and metadata. In this case, the registry would typically contain public data, and operate on the Internet, or could be operated on a secure extranet. If it is the case that a statistical domain wishes to produce standard data and/or metadata structure definitions, then having a central repository to act as the official distribution point makes some sense. Further, it provides for immediate visibility into the data sets and metadata sets available across the domain, acting as a "domain portal".

To Support Data Collection and Reporting Activities:

This scenario involves the operation of a registry to support the collection of statistics from a large number of reporters, and could support collection of public data on the Internet, but would probably be on a secure extranet. The collecting organization would operate the registry for the benefit of those reporting to it. This makes most sense if the data collected are aggregated and then re-distributed to the reporters in their aggregate form.

Federated Registries:

Rather than having a single, huge registry for all statistical data and metadata – which would involve some immense challenges of scale and operations – the typically envisaged scenario is to have many domain registries which are federated. Many of the existing registry implementations – notably those compliant with ISO 15000 (ebXML), parts 3 and 4 –support the idea of “federation”. This is essentially a network of registries which provide visibility into each other’s contents. A user can choose to query only the local registry, or to query the network. SDMX has not yet specified a specific mechanism of federation, but because these standards already exist, it is easily conceivable that a federation of statistical registries could be created, providing visibility into all of the statistical data and metadata from participating domains and organizations.

6 OVERVIEW OF THE INFORMATION MODEL

6.1 Fundamental Aspects

6.1.1 Scope of the Model

It is important to understand that the SDMX-IM is a technical artefact. Whilst being a conceptual model it is used as the basis for the various syntax implementations (e.g. SDMX-EDI and SDMX-ML). It is also used as the basis for the SDMX registry design and the SDMX registry interfaces – these interfaces are specified in terms of XML schemas.

Therefore, the SDMX-IM is intended for use by implementers of syntax specifications and systems that need to process SDMX documents. For the technician the SDMX-IM can play the role of being a specification of the scope of SDMX.

The SDMX-IM covers the requirements for:

- Data structure definition (Key Family) including category scheme, concept scheme, and code list
- Data and related metadata reporting and dissemination
- Metadata structure definition
- Metadata reporting and dissemination
- Hierarchical code scheme
- Mapping between structures
- Process – operations performed on data in order to validate data or to derive new information according to a given set of rules defining the transitions between input and output processes

The metamodel also supports the specific data and metadata requirements of the SDMX registry, where these requirements have an impact on the data structure definition and metadata structure definition and data and metadata reporting and registration. The functions of the SDMX registry, its model, and the map to the SDMX-IM are documented separately.

6.1.2 Use Cases

The UML model documents the “use cases” which are the user/application functionality that the model supports. The fundamental requirement of the model is to support data and metadata exchange, whether this be in a “push” scenario where the data provider sends the data or metadata to the data consumer, or in a “pull” scenario where the data consumer “discovers” or is informed of the availability of data and where it is located, and retrieves it directly from this location.

In order to achieve this, the model supports the definition of the components used to:

- structure the data and metadata, related concepts and code lists
- contain the data and metadata
- manage the process of exchange
- register data and metadata
- discover data and metadata

The user of the SDMX standards will use them to:

1. Develop and publish data structure definition and metadata structure definitions, together with related concepts and code lists.
2. Publish data and metadata.
3. Consume (receive) data and metadata.
4. Register the existence of the data and metadata.
5. Query for data and metadata.

The first three points above are common to both “push” and “pull” scenarios, the last two points are specific to the “pull” scenario.

6.1.3 Package Structure

The SDMX-IM has been structured into a number of packages to aid the understanding, re-use and maintenance of the model. These packages act as convenient compartments for the various sub models in the SDMX-IM. The diagram below shows the sub models of the SDMX-IM that are included in the version 2.0 specification.

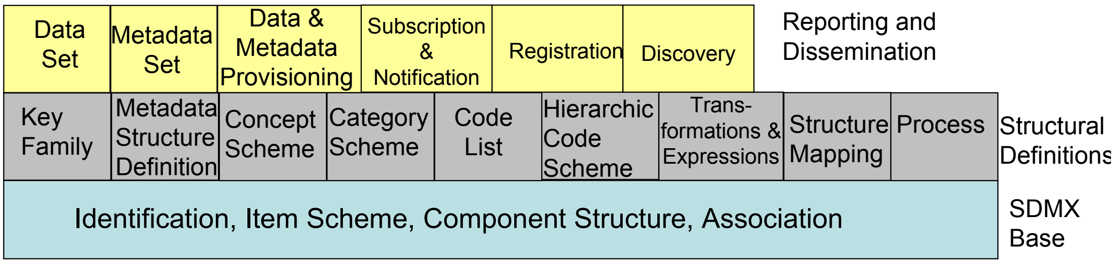


Figure 3: SDMX Information Model Version 2.0 package structure including the registry

Although any package can make use of any construct in another package, in conceptual terms the packages are shown in three layers:

- the SDMX Base layer comprises fundamental building blocks which are used by the Structural Definitions layer and the Reporting and Dissemination layer
- the Structural Definitions layer comprises the definition of the structural artefacts needed to support data and metadata reporting and dissemination
- the Reporting and Dissemination layer comprises the definition of the data and metadata containers used for reporting and dissemination, and the metadata concerned with the reporting and provisioning of the data and metadata

Note that the following packages are specific to a registry based scenario:

- Subscription and Notification
- Registration
- Discovery

Note also that the data and metadata required for registry functions are not confined to these three packages, and the registry also makes use of the other packages in the Information Model (such as structural definitions).

6.1.4 Model Inheritance

Inheritance is a very powerful feature of object oriented modelling and implementations. A good example of the power of inheritance is the identification mechanism in the SDMX-IM. The class diagram of the identification mechanism is shown below.

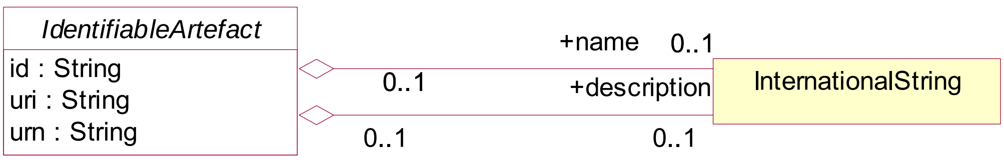


Figure 4: Identification in the SDMX-IM

[UML Note: A convention often used in UML modeling is to represent all abstract classes with no colour (shown in white above), and all concrete classes in a colour (shown in yellow above). An abstract class is one that not “used” directly in the model, but is only used as one of its concrete sub classes. In the diagram above the Identifiable Artefact is used to identify objects.]

There are three attributes in the Identifiable Artefact: the id, the universal resource identifier (uri), and universal resource name (urn). There are no description or name attributes in the Identifiable Artefact, but rather these data are provided by an International String which allows multi-lingual representation of text. Any Identifiable Artefact can have one name and one description, each having multi-lingual representations.

Sometimes, there may be more than one level of abstract class: for instance, the SDMX-IM has other abstract classes inheriting from Identifiable Artefact, with concrete classes inheriting from the lower level abstract class. Examples are Key Descriptor (which is a part of the Data structure definition whose function is to group the Dimensions that comprise the structure of the full series key), and Dimension (this is a component of the Key Descriptor).

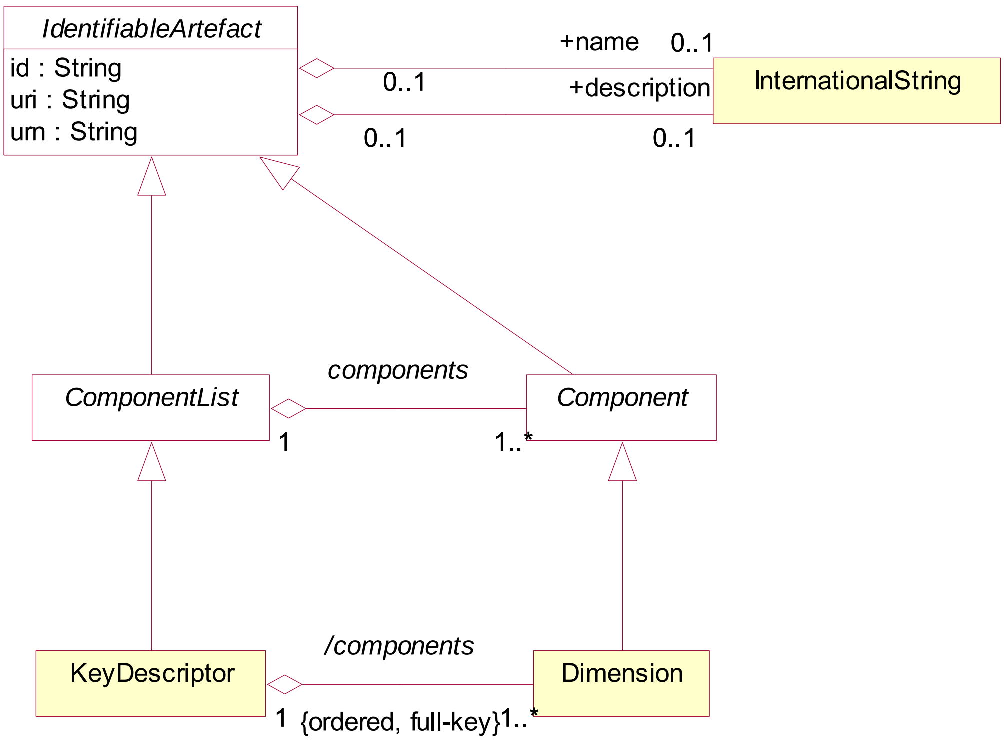


Figure 5: Example of multi-level inheritance

The Key Descriptor inherits all the facets of both Component List and Identifiable Artefact. Therefore, a Key Descriptor has an id, uri, and urn and can have a multi-lingual name, and multi-lingual description. In addition it can have Components, or, more precisely, as Component is an abstract class, an artefact that inherits from Component. In the diagram above the Key Descriptor actually has one or more Dimensions, and the association between the Key Descriptor and Dimension (marked as "/components" in the diagram) is inherited from the association between Component List and Component (marked as "components" in the diagram).

[UML Note:The UML convention is to put the "/" in front of the name of an inherited association.]

The reason to model an inherited association is twofold:

1. It makes the diagram more understandable, especially if the abstract classes are not shown on the diagram.

2. The "business rules" or constraints for the inherited association can be different: in the example above there is a constraint that the Dimensions must be ordered and contain the full key.

The major reason that abstract classes are modeled is to be able to share features that are common amongst several classes. For example, many artefacts in the model are either identifiable, versionable, or maintainable and it is useful to have a common way of applying this. It is also quite useful to identify and model "patterns" (i.e. groups of classes which fulfill a certain function that can be shared). An example is the Item Scheme from which are derived the Code List, the Concept Scheme, and the Category Scheme. The advantage in modeling these patterns is that the implementation can be common, either in syntactical terms, or in programming terms. For instance, a java programmer would be able to implement all three types of scheme with much common code: the only reason for writing code specific to just one of the sub classes is when there is something different about the sub class, such as the need to access an attribute that is only in the sub class.

6.2 The SDMX Base Layer

6.2.1 Introduction

The intention of the Base package is to model all the things that are common and that are shared throughout the specific parts of the model. Common things are:

- Identification, versioning, and maintenance
- a scheme of items that can be related together or related with items from other schemes (e.g. in concrete terms a code list can have codes, and each code can have child codes)
- a structure of components that can have associations to a scheme of items (the data structure definition and metadata structure definition are concrete examples of two such component structures)
- Organisations and organization roles (maintenance agencies and data providers are concrete examples of the organisation pattern)

6.2.2 Identification, Versioning, and Maintenance

6.2.2.1 Overview

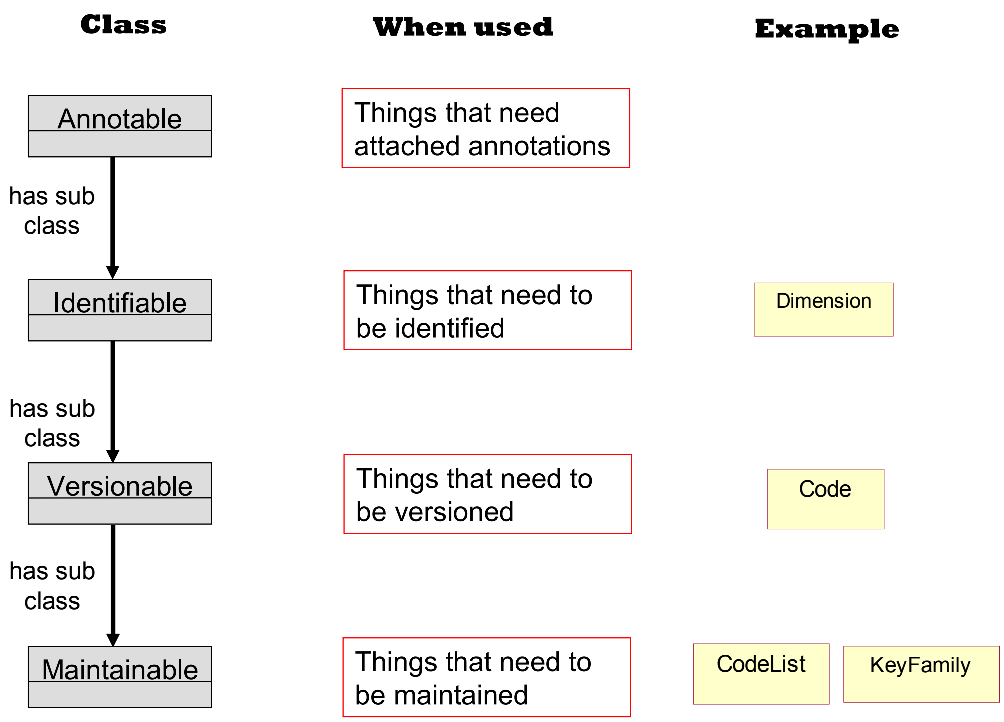


Figure 6: Identification, versioning and maintenance

The diagram above shows an inheritance tree, with each lower level class inheriting from each of the higher level classes (called super classes).

Therefore, the Code List and Key Family have annotation, identification, versioning and a link to a maintenance agency, whereas the Code has annotation, identification and versioning. It follows from this that whilst a Code can be versioned within a Code List, all Codes in a Code List must have the same maintenance agency, as this is the maintenance agency of the Code List. On the other hand, a Dimension can be identified but cannot be versioned and cannot have its own maintenance agency – it is maintained by the maintenance agency of the Key Family to which it belongs and if the specification of the Dimension changes then there will be a new version of the Key Family.

Note that all the example classes can have annotations. There is no class in the SDMX-IM that has just annotations.

6.2.2.2 Class Diagram

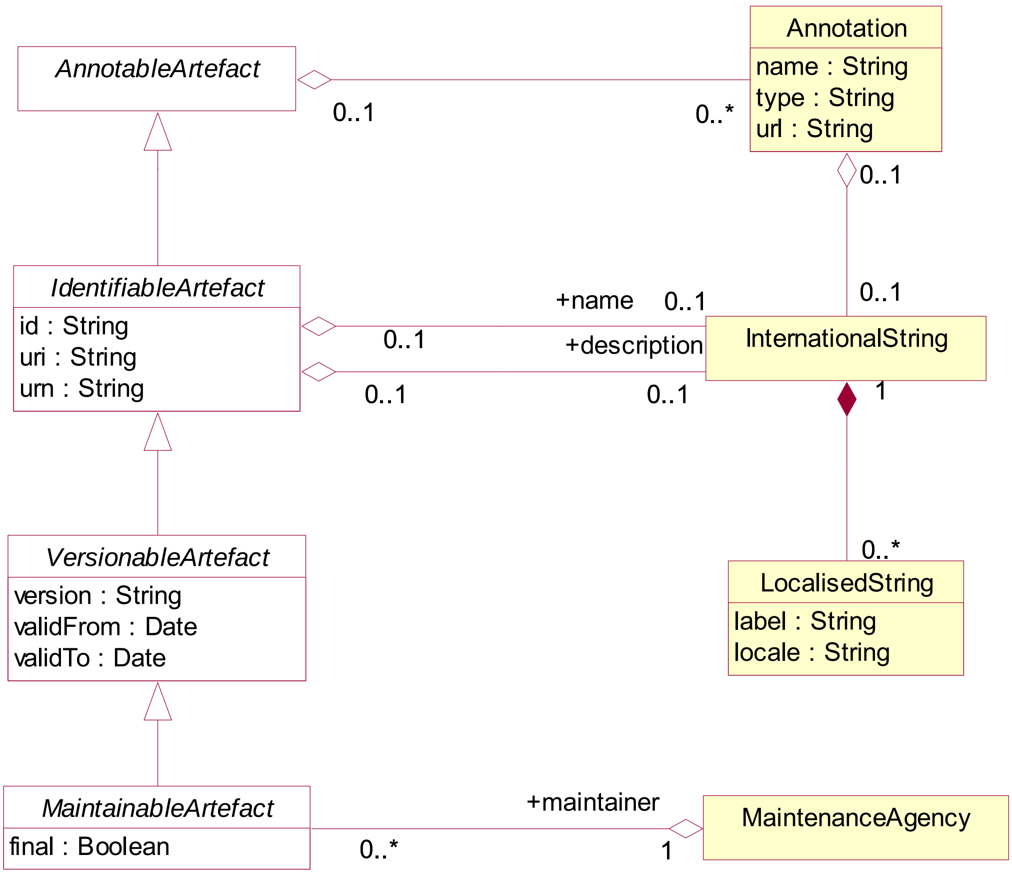


Figure 7: Class diagram of identification, versioning and maintenance

Annotations have a name, are identified by a type, can have a link to a url, and can have a link to a multi-lingual description.

Identifiable Artefacts can have both a multi-lingual name and a multi-lingual description. This is supported by the International String which is, in effect, a way of grouping multiple language variants of the same name: this is called the locale as the locale supports not just language, but language variants such as American English or Canadian French. The label is the text in the language identified by the locale.

For things that are Versionable there is, in addition to the identifying facets, a version and validity start and stop date. Maintainable Artefacts are the same as Versionable Artefacts except they also have a Maintenance Agency, and an additional attribute (final) which denoted whether the object is draft or final.

An example of the concrete use of these classes is shown below for a Code List.

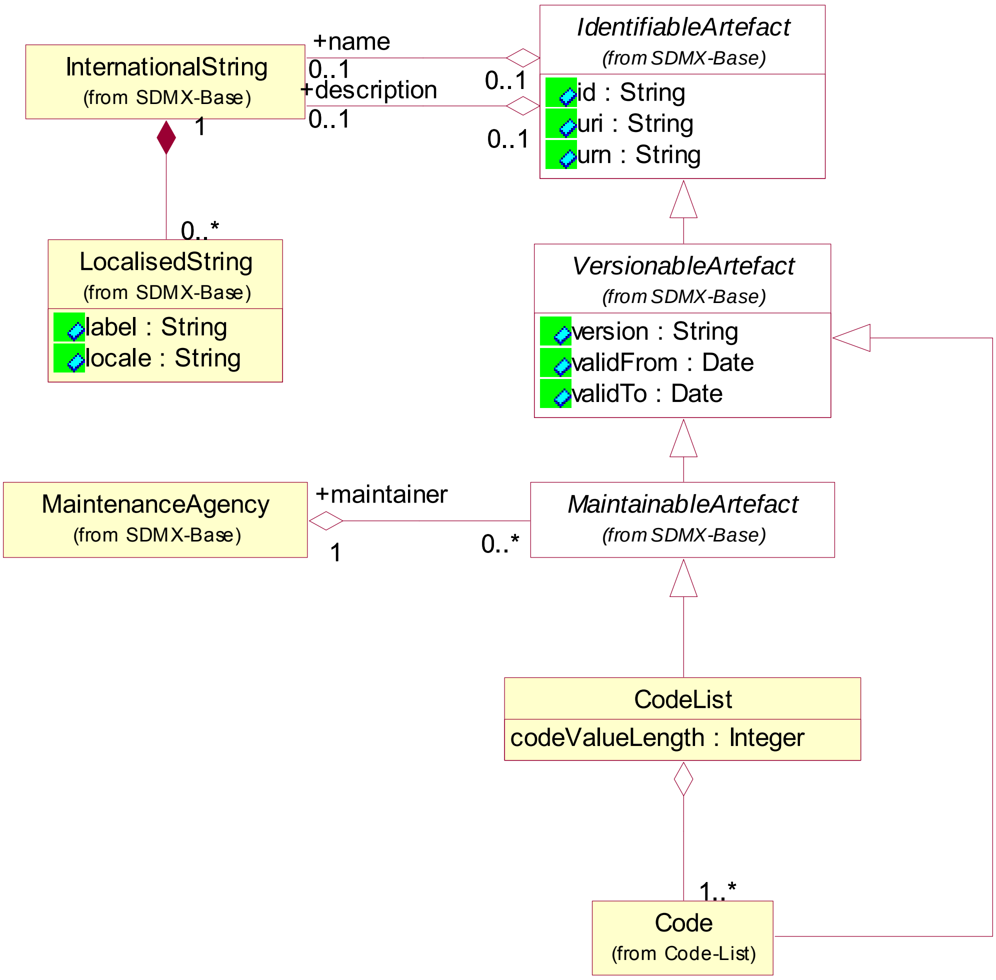


Figure 8: The simple Code List model showing inheritance from the Base

[note that for reasons of clarity some abstract classes and associations are not shown on this diagram – the full diagram is shown later].

The Code List inherits from Maintainable Artefact and therefore is identifiable, versionable, and maintainable. The Code inherits from Versionable Artefact and is therefore identifiable and versionable. The Code List can have one or more Codes as shown by the association “items”.

Essentially, the diagram is stating the following business rules:

- both a Code List and a Code have the attributes
  - `Id`
  - `Uri`
  - `Urn`

- o `Version`
- o `validFrom`
- o `validTo`

- furthermore, they both have a name and a description that can be represented in multiples languages (locale).
- the Code List has Maintenance Agency and can be given a status of final
- the Code List can have one or more Codes

6.2.3 The Item Scheme

6.2.3.1 Simple Classification Schemes

The Item Scheme can be used to structure a simple classification scheme which can be hierarchic but where each child Item can have only one parent Item.

Note that the term "classification" is used here to describe things that can classify objects. Examples are concept schemes, category schemes and code lists. The terms "classification" is not used here to denote the full semantic of a statistical classification.

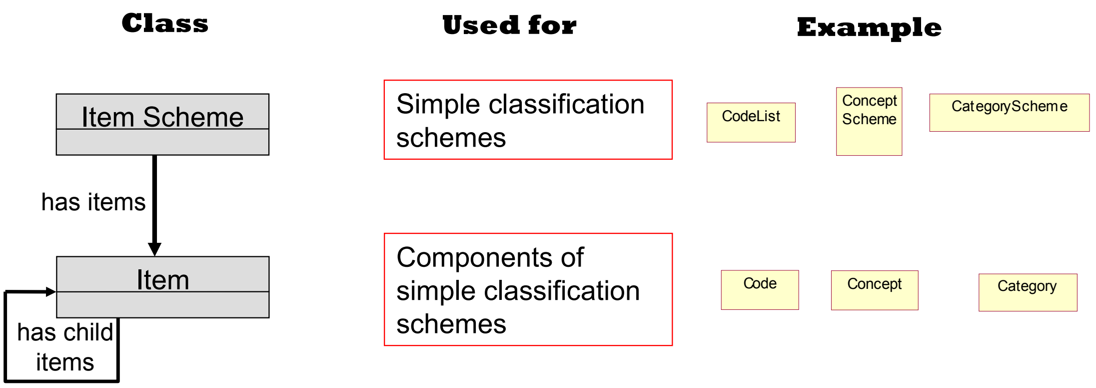


Figure 9: Schematic of the simple item scheme

The Item Scheme is used to define a list of Items. The Items in the scheme can have child items. The restriction of this simple hierarchy is that each child Item has only one parent Item. More complex parent-child relationships can be specified using the Hierarchical Code Scheme which (see section 6.3.8).

Examples of an Item Scheme are a Code List which comprises Codes, a Concept Scheme which comprises Concepts, and a Category Scheme which comprises Categories such as subject matter domains or reporting categories.

6.2.3.2 Class Diagram of the Item Scheme

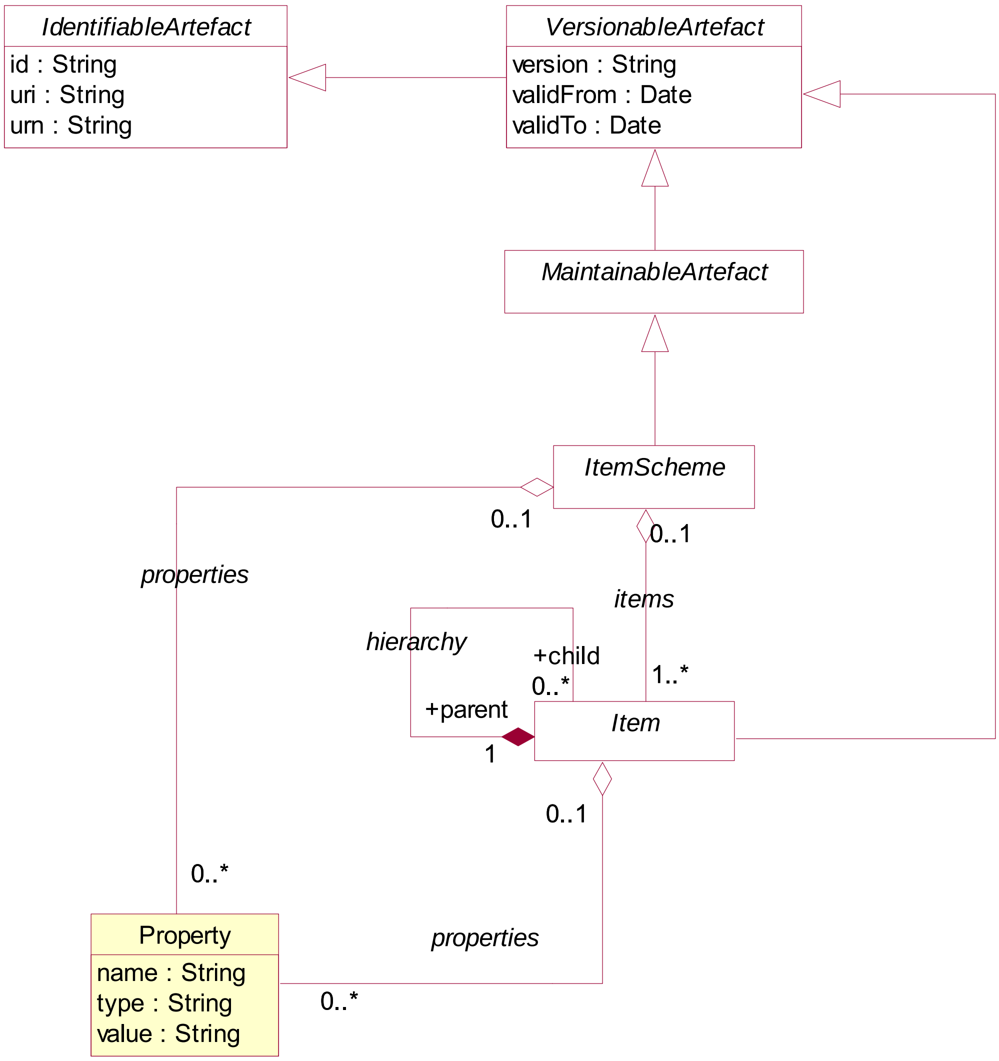


Figure 10: Class diagram of the Item Scheme

The Item Scheme is maintainable and therefore will have a maintenance agency. The Item is versionable, but not independently maintained i.e. it is maintained within the structure in which it is contained, such as an Item Scheme.

Both the Item Scheme and the Item can have Properties, which contain a specific value for a particular type of Property. For instance, in a mapping table (see later) which maps a source object to a target object (such as code in one code list to a code in another code list) the value in the Item Property could be the weighting factor to be applied to the value of the source code when analyzing or aggregating the data according to the target code. Another use of the Property is in a complex hierarchical scheme where the sequence of the set of child items is important – the sequence number can be held in the Property.

6.2.4 The Structure Scheme

6.2.4.1 Overview

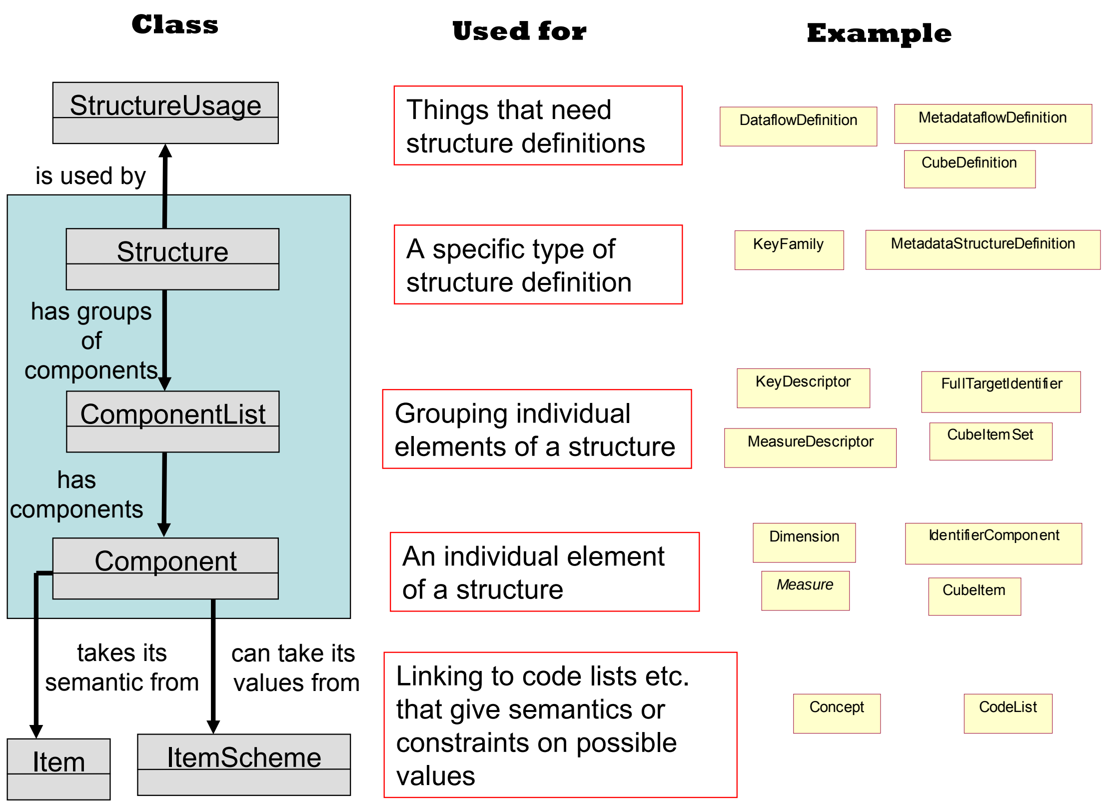


Figure 11: The Structure Scheme

The Structure Scheme is quite simple yet very powerful. The Structure comprises groups (Component List) of Components. A Component must take its semantic from an Item (in an Item Scheme) - for instance this could be a Concept in a Concept Scheme. The possible values and format for a Component when data or metadata are reported can be specified with the Concept as the “core” Representation. This Representation can be overridden with a “local” Representation within the concrete use of a Structure (such as a Key Family (Data Structure Definition)). An example of a Representation is a Code List. Note that the Structure is independent of its usage i.e. it is shareable and can be used by many artefacts. For instance, a Key Family may be used by many Dataflow Definitions and Metadata Structure Definition may be used by many Metadataflow Definitions.

A concrete example of a something that needs to be defined by a Structure is a Dataflow Definition. Data reported for a Dataflow Definition must conform to a Key Family. A Key Family is a structure definition for data and related metadata and has three basic groups of Components: one to define the key components (Key Descriptor) which comprises Dimensions; one to define the measure components (Measure Descriptor); one to define the attribute components (Attribute Descriptor). Each of these Components take their semantic from a Concept in a Concept Scheme, which can be linked to a Code List which defines the valid values that the Component can take in when data or metadata is reported in a data set.

6.2.4.2 The Structure Pattern

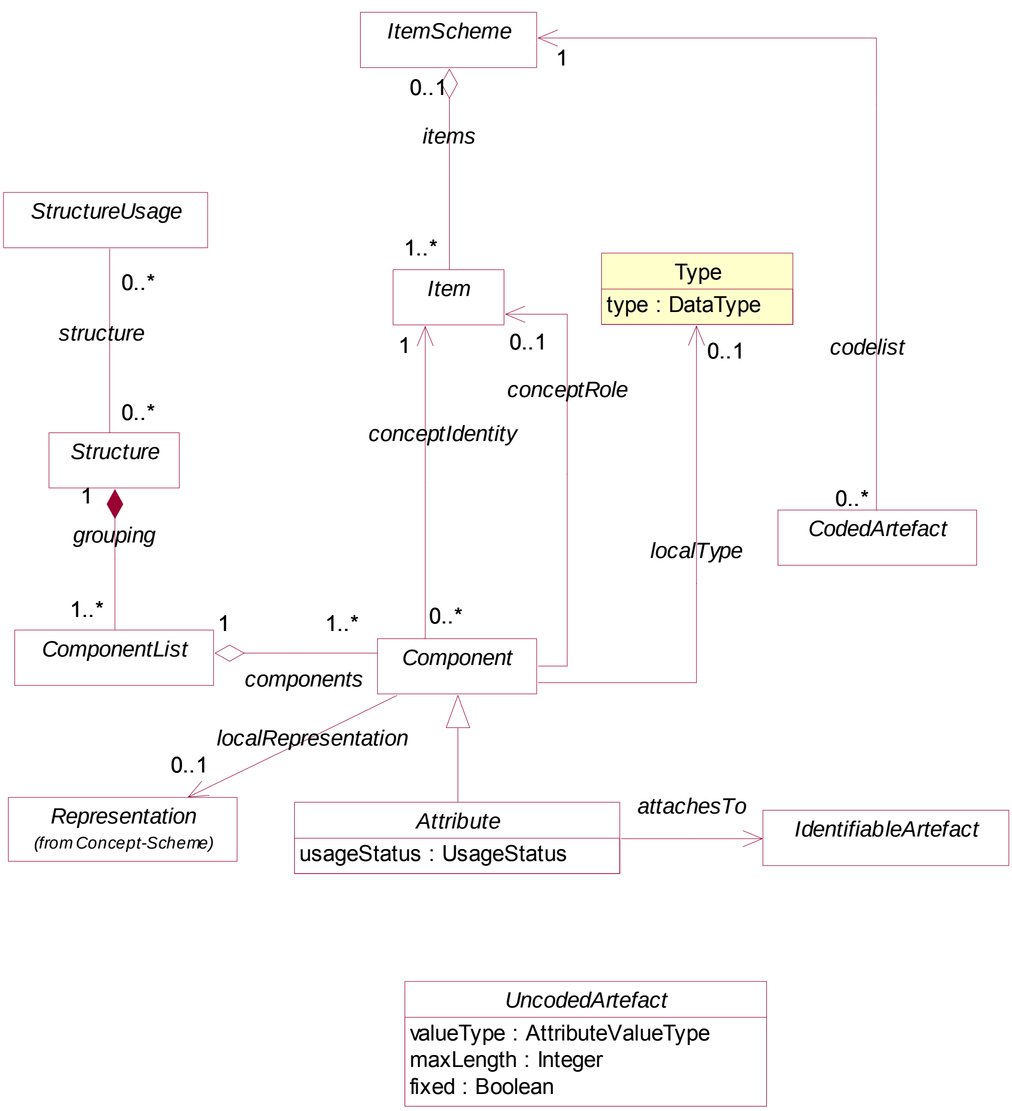


Figure 12: The Structure pattern in the SDMX Information Model

This pattern is used by both the Data Structure Definition (see 6.3.5) and the Metadata Structure Definition (see 6.3.7), which means both the data structures and the metadata structures are defined using essentially the same underlying UML structures. This is important when building software systems to support SDMX, as common software components can be developed to support both types of structural definition.

The Coded and Uncoded Artefact are used in both the Data and Metadata Structure Definitions, where concrete data and metadata classes inherit from them.

6.2.5 Relating Structures and Item Schemes

6.2.5.1 Overview

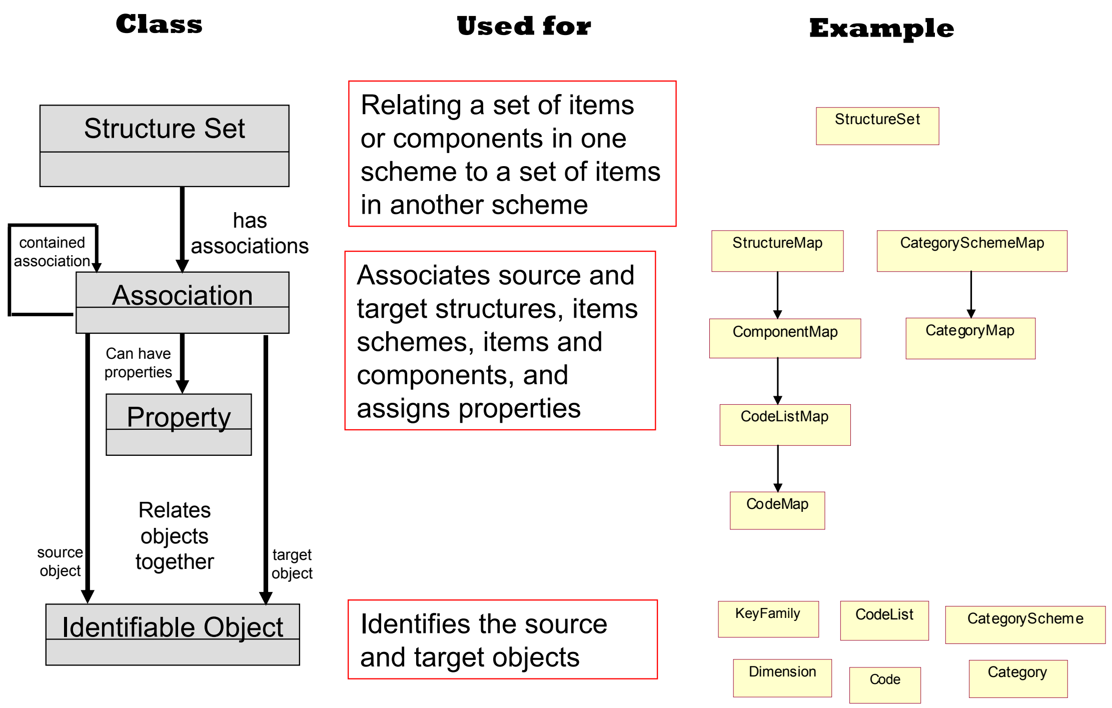


Figure 13: Schematic of the Structure Set

The SDMX-IM has a very powerful mechanism for relating two Item Schemes, such as two Code List or two Category Schemes, or two Structures such as two Data Structure Definitions or two Metadata Structure Definitions. This is achieved by the Association. The Association has the following structure:

- it relates a source object to a target object
- it can have properties that define aspects of the association
- it can have a tree structure of sub associations (contained association)

An example of a simple association is a Code List Map which relates together two Code Lists. The contained association of the Code List Map is the Code Map which relates together two Codes. A more complex association is that between two Data Structure Definitions where it may be necessary to link not only the components of the definitions (such as the Dimensions, Attributes etc.), but also the Code Lists used by the components in the Data Structure Definition. Here there is the possibility of a four level tree:

Structure Map – Component Map – Code List Map – Code Map

6.2.5.2 The Association Pattern

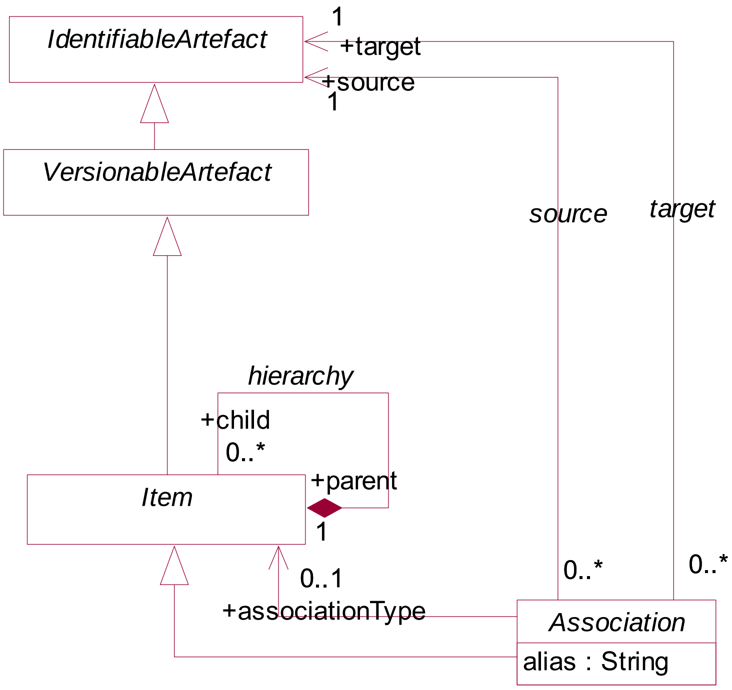


Figure 14: The Association pattern in the SDMX Information Model

The Association is an Item and so can have properties and can have child Items. The Item is versionable and so the Association can be versioned. The Association links one object (source Identifiable Artefact) to another (target Identifiable Artefact). In the SDMX-IM for the Structure Set these Identifiable Artefacts are restricted to:

- Structures (Data Structure Definition and Metadata Structure Definition)
- Component (Dimension, Measure, Data Attribute, Identifier Component, Metadata Attribute)
- Structure Usage (Dataflow Definition, Metadataflow Definition)
- Item Scheme and Items (Code List-Code, Category Scheme – Category, Concept Scheme – Concept)

Note that the Association can be contained in any structure that can be maintained. For mapping tables this structure is a Structure Map.

6.2.6 Organisations

An organisation can play many roles. In the SDMX-IM three roles are identified:

- Maintenance agency
- Data (and metadata) provider
- Data consumer

Lists and identities of organizations are maintained in an Organisation Scheme. The Organisation Scheme is a specialised form of Item Scheme. Therefore, from a maintenance point of view, the Organisation Scheme is just like any other Item Scheme.

6.2.7 Summary of the SDMX Base

The SDMX Base comprises, on the whole, abstract classes arranged in core structures that are used in the rest of the SDMX-IM. Most classes in the sub models that define structure (Structural Definitions packages) inherit from one of the patterns described above: in fact in these structural sub models there are few classes that do not inherit from one of the classes in these patterns - clearly there are some additional classes that are specific to one or other of the various structural sub models, but these are few. The classes in these patterns themselves inherit identification, versioning, and maintenance as appropriate, thus all structures and their components (e.g. Key Family and related components) have appropriate identification, versioning, and maintenance.

6.3 Structural Definitions Layer

6.3.1 Introduction

The purpose of the packages in this layer is to define structural metadata. The list of structural metadata is

SDMX structural feature	**Inherits from SDMX base pattern**
Code List	Item Scheme
Concept Scheme	Item Scheme
Category Scheme	Item Scheme
Hierarchical Code Scheme	Item Scheme
Data Structure Definition (Key Family)	Structure
Metadata Structure Definition	Structure
Structure Set comprises: - Structure Map - Component Map - Code List Map - Code Map - Category Map - Concept Map - Organisation Scheme Map	Maintainable Artefact - Association - Association - Association - Association - Association - Association - Association
Process	Item Scheme
Transformation Scheme	Item Scheme

Structural definitions comprise "active metadata" and are used to define structural metadata that are used by applications to interpret the data and metadata exchanged. The data and metadata are exchanged in the Data Set and the Metadata Set, and the structure of the data and metadata in these "sets" are described in the Data Structure Definition (Key Family), and the Metadata Structure Definition. These definitions in turn comprise sub structures that relate to Code Lists, Concept Schemes, and Category Schemes in order to further define the semantic and the valid content of the data and metadata. The components in the various structures and item scheme can be mapped using the Structure Set. Processes can be described that transform data from an input process to an output process, these transformation processes can be formally defined as Expression Nodes in a Transformation Scheme.

6.3.2 Example Data Set

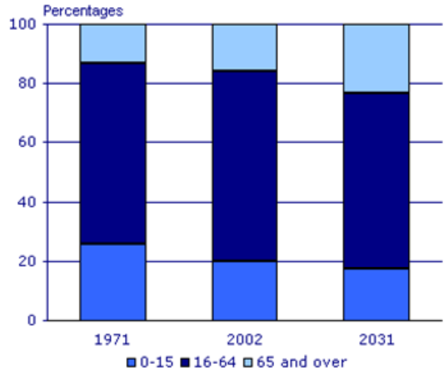
The example code lists, concept schemes, and data structure definition in this section are built from the following data table. The example for the Metadata Structure Definition is given later in this section.

national STATISTICS

People & Migration

Age structure

Average age rose to 38.2 years in 2002



Population: by age, UK

This ageing is the result of declines both in the numbers of children born and in mortality rates. This has led to a declining proportion of the population aged under 16 and an increasing proportion aged 65 and over.

Age structure: regional comparison, 2002

	0-15	16-64	65 and over
			Percentages
North East	19	64	17
North West	20	64	16
Yorkshire and the Humber	20	64	16
East Midlands	20	64	16
West Midlands	20	63	16
East	20	63	17
London	20	68	12
South East	20	64	16
South West	19	62	19
England	20	64	16
Wales	20	63	17
Scotland	19	65	16
Northern Ireland	23	63	13
United Kingdom	20	64	16

Sources:
Age structure up to 2002: population estimates, Office for National Statistics, General Register Office for Scotland, Northern Ireland Statistics and Research Agency
Age structure for 2031: population projections, Government Actuary's Department

Notes & Definitions

Published on 24 June 2004 at 9:30 am

Figure 15: Example demography data set

In this example we are primarily interested in defining the structure of the regional table: it is assumed that the graph at the top can be derived from annual figures of a regional table.

Whilst this document is not aimed at being a tutorial on data structure definitions the following sections explain the structural aspects of this regional table so as to give a better understanding of how the SDMX-IM supports this.

Note that the Data Structure Definition described here is developed by the authors for the purpose of illustrating the content and structure of a Data Structure Definition. It does not represent any real Data Structure Definition for demographic data that may have been developed by the ONS or any other organisation.

6.3.3 Concept Scheme

6.3.3.1 Concepts in the Example

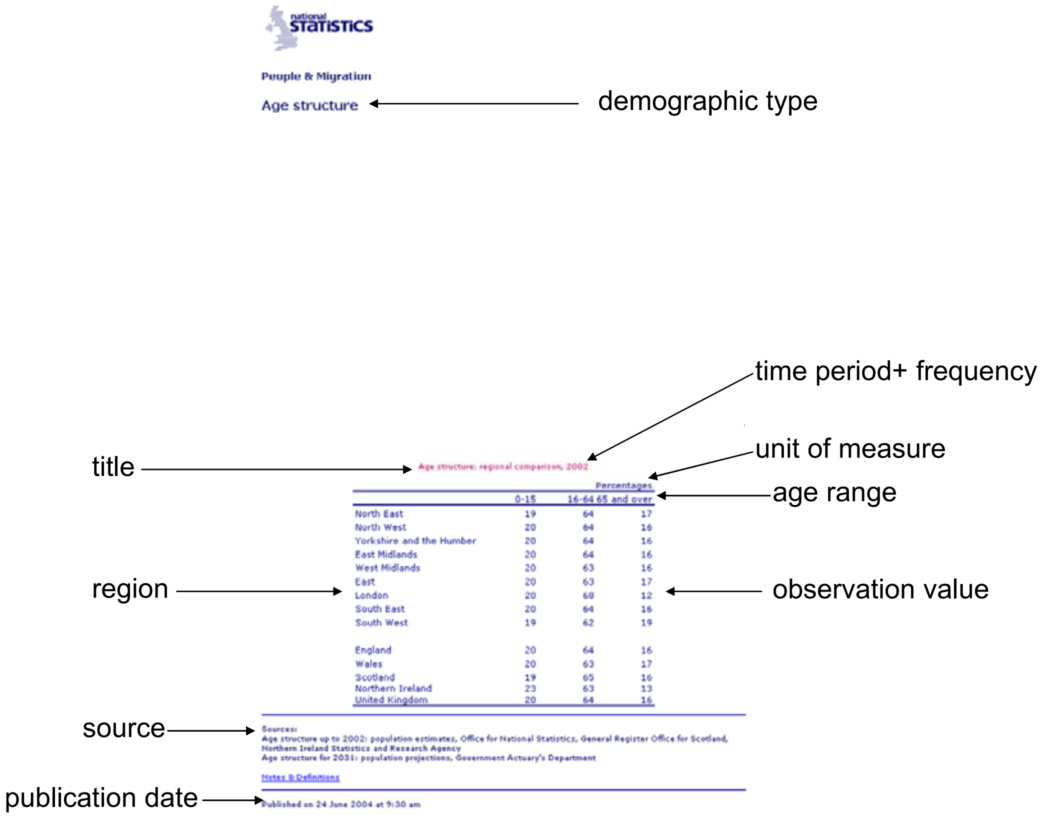


Figure 16: Identification of Concepts

Data Structure Definitions (Key Family) comprise specifying the concepts of a multi-dimensional structure, specifying the value domains (codelists), and defining what role the concepts play in the multi-dimensional structure.

The diagram above identifies the concepts. The concepts must be maintained in a Concept Scheme and they can be hierarchical (i.e. they can be defined in a semantic hierarchy). For Data Structure Definitions the concepts are a simple flat list, though they could be derived from a hierarchical scheme: i.e. the Data Structure Definition itself has no knowledge of superior and subordinate concepts.

6.3.3.2 Schematic of the Model

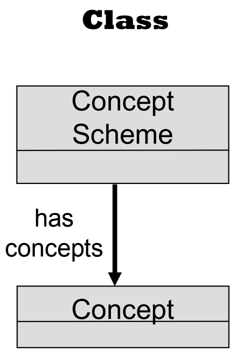


Figure 17: Schematic of the concept scheme

The Concept Scheme comprises a simple list of Concepts. Note that in this simple example the Concept is not given a “core” Representation and all Representations are specified explicitly for the Components in the Data Structure Definition. In the example the scheme would comprise, amongst other concepts, the ones identified for the demographic data structure:

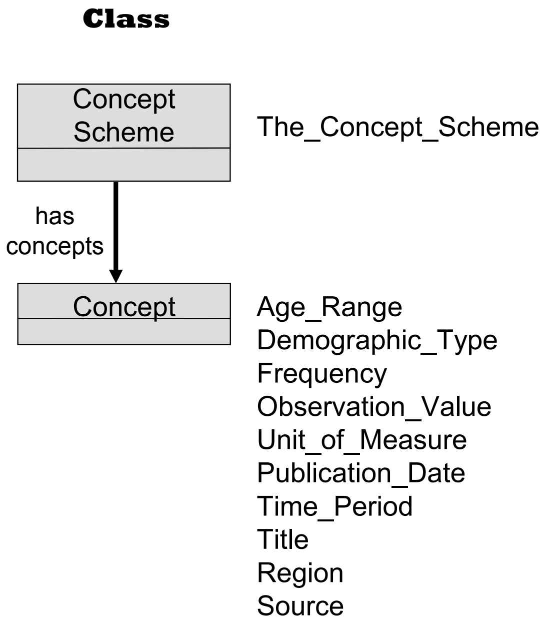


Figure 18: Example list of concepts

6.3.3.3 The Information Model

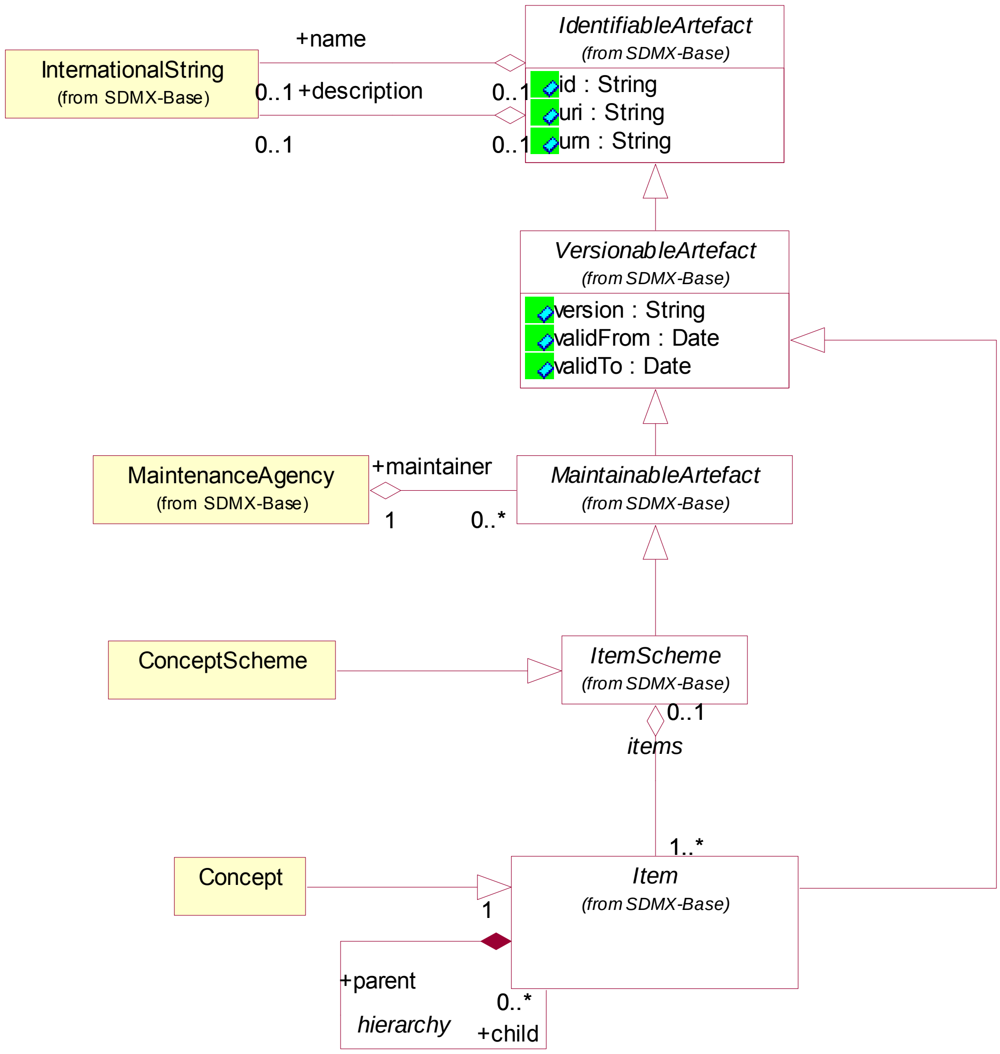


Figure 19: Concept scheme in the SDMX Information Model

Via the inheritance from the SDMX Base (see Section 6.2.3 for an explanation) the Concept Scheme has a Maintenance Agency and is Versionable and so has identification and version attributes. The Concept is Versionable.

The model allows Concepts to be related in a tree structure to other Concepts. The intent here is to relate Concepts that have a semantic association such as:

Telephone number
- Office telephone number
- Cellphone number

However, for the simple case of defining a Data Structure Definition it is only necessary to have a Concept Scheme with a flat list of Concepts. The simple structure of the Concept Scheme for this purpose is shown below in a simple form of relational diagram which shows all of the inherited attributes and associations.

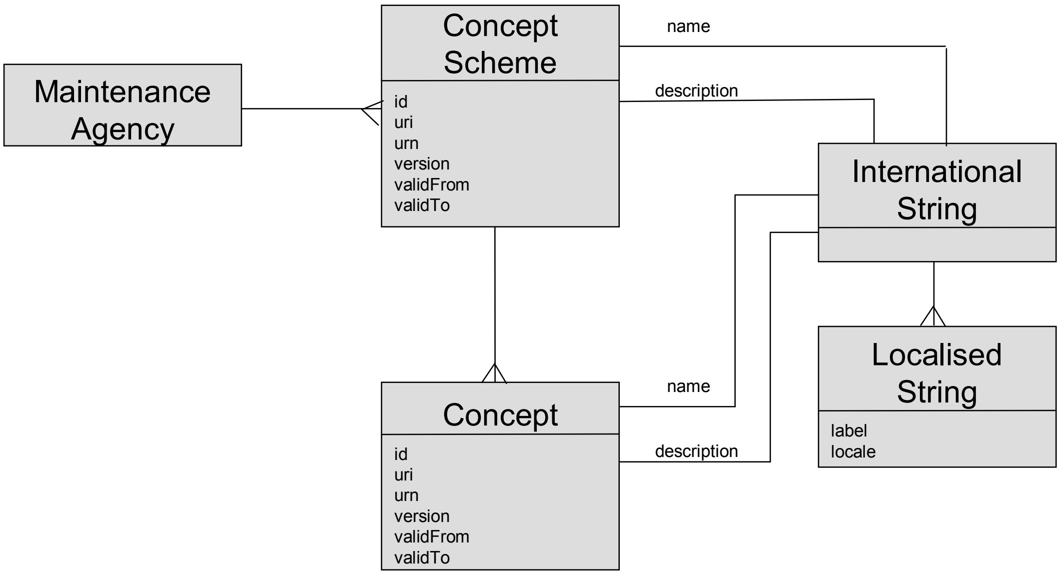


Figure 20: Concept Scheme showing inherited attributes and associations

6.3.3.4 Example Implementation of the Model

An example of a user interface of a tool which could maintain such a scheme based on the Information Model constructs is shown below.

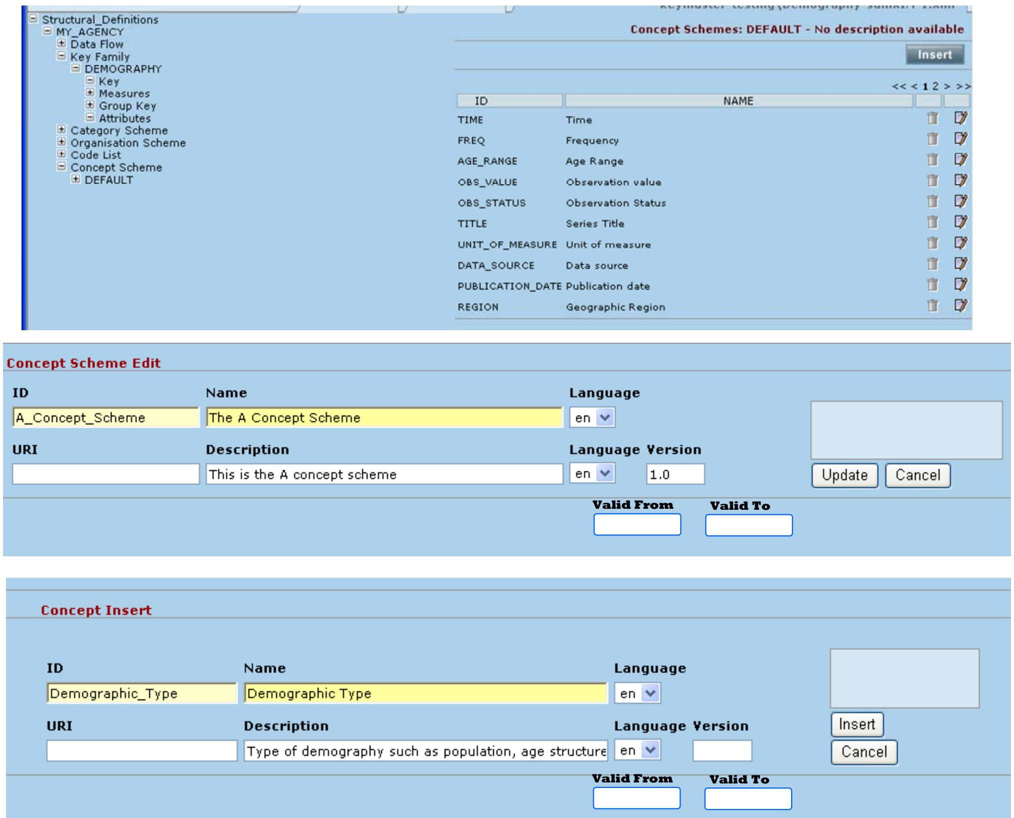


Figure 21: Example of a user interface for Concept Scheme based on the SDMX-IM

In the visual above it can be seen that:

- the Concept Scheme is “owned” by a Maintenance Agency (My_Agency)
- both the Concept Scheme and Concept have identical attributes and associations for the multi-lingual name and description (whilst the tool displays only one language variant of both name and description, multiple variants are allowed (only en:English) is shown)

6.3.4 Code List

6.3.4.1 Code lists in the Example

Much of the text shown on the example at Figure 15 is, in fact, coded information; whilst the code description is shown in the published data set on the web the actual data held is a code which is decoded from a code list.

The following information is likely to be coded:

- Age_Range
- Demographic_Type
- Frequency
- Unit_Of_Measure
- Region

6.3.4.2 Code Lists and the Model

The Code List is supported in the model in exactly the same way as a Concept Scheme. By substituting Code List for Concept Scheme and Code for Concept in the relevant diagrams at section 6.3.3, one can see easily the way the Code List is supported. The model allows a Code List to have simple hierarchy of Codes.

6.3.4.3 Example Implementation of the Model

An example of a user interface of a tool which could maintain a simple code list based on the Information Model constructs is shown below.

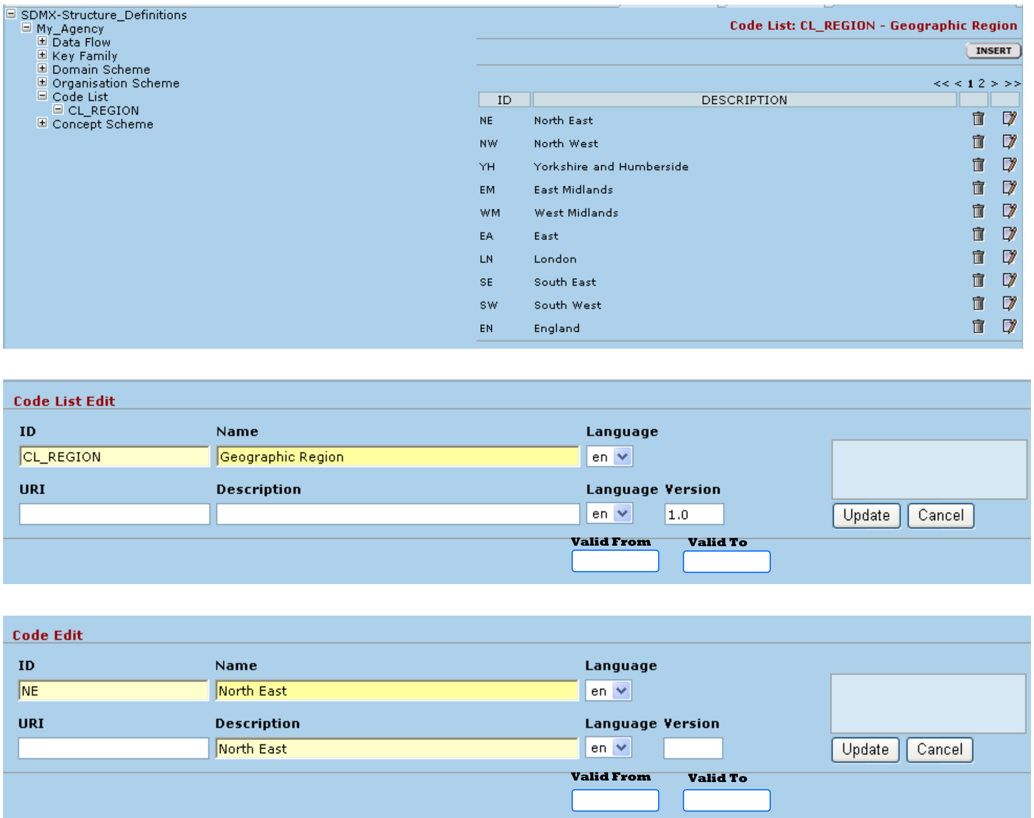


Figure 22: Example of a user interface for Code List based on the SDMX-IM

It can be seen from the example above that a code list used in a Data Structure Definition can comprise codes from various levels of a hierarchic code list (e.g. EN:England is in fact a “total” of the regions) and these are represented as a flat list. The way the model supports complex hierarchic code lists such a geographic scheme, where one code have more than one parent code, is described later (see Hierarchic Code Scheme).

6.3.5 Data Structure Definition (Key Family)

6.3.5.1 Schematic

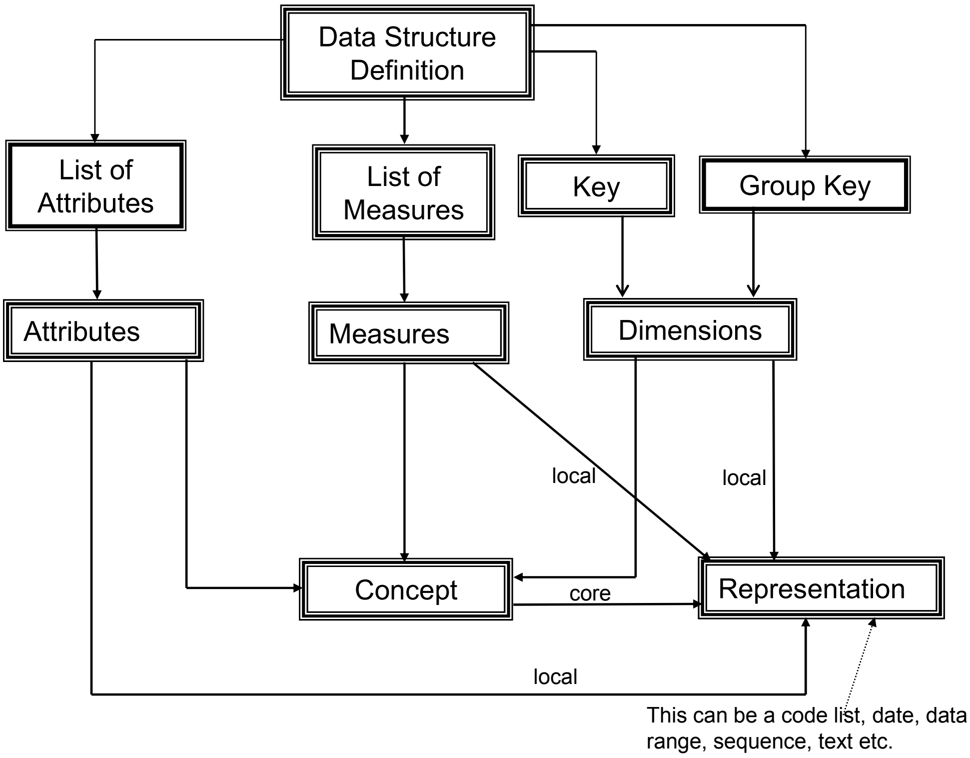


Figure 23: Schematic model of the Data Structure Definition

The Data Structure Definition comprises a number of lists of components:

- A list of Dimensions that comprise the Key or the Group Key
- A list of Measures
- A list of Attributes

The Key comprises the Dimensions or "Classificatory Variables" whose values in a Data Set, when combined, uniquely identifies the observed data values (the observation values in the Data Set that conform to the Measures). A Group Key comprises a sub set of these Dimensions (it is a partial key) and is used to link to Attributes that give some metadata about the object identified by the combined values of the Dimensions forming the partial key.

The List of Measures comprises one or more Measures, each of which is a phenomenon for which an observation is relevant or required.

The List of Attributes comprises one or more attributes that can be used to give more information (metadata) about some part of the data set. Each Attribute in the Data Structure Definition must be assigned to an identified part of the data set (in the model this is called the "attachment level"). The Data Structure Definition supports the following "attachment levels").

- The observation (this is identified by a value for each of the Dimensions of the Key plus the time value)
- A series key (this is identified by a value for each of the Dimensions of the Key)
- A group key (this is identified by a value for each of the Dimensions of the Group Key)
- A data set

Each of the components of the Data Structure Definition (Dimension, Measure, Attribute) is assigned a Type Representation. This can be a code list, a date, a date range, numeric range etc. This Representation can be taken from the Concept (core Representation) or it can be defined locally in the Data Structure Definition (local Representation). A locally defined Representation overrides the core Representation.

6.3.5.2 Mapping the Schematic to the Information Model

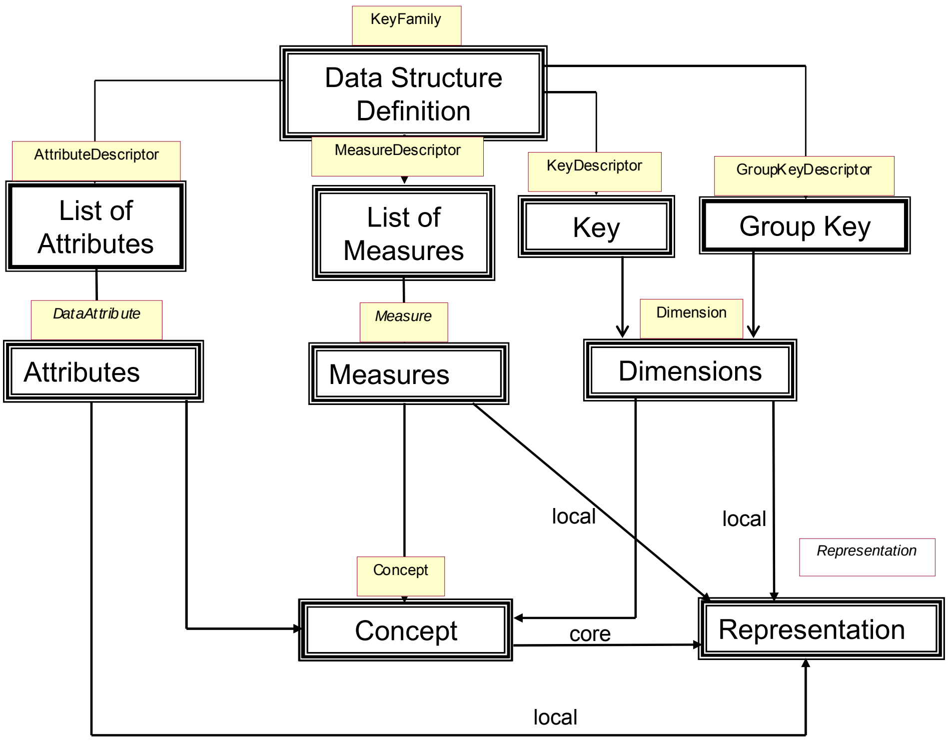


Figure 24: The map between the Data Structure schematic and the model classes

This diagram shows the model classes that represent the boxes in the schematic. The diagram below shows the actual model of the Data Structure Definition.

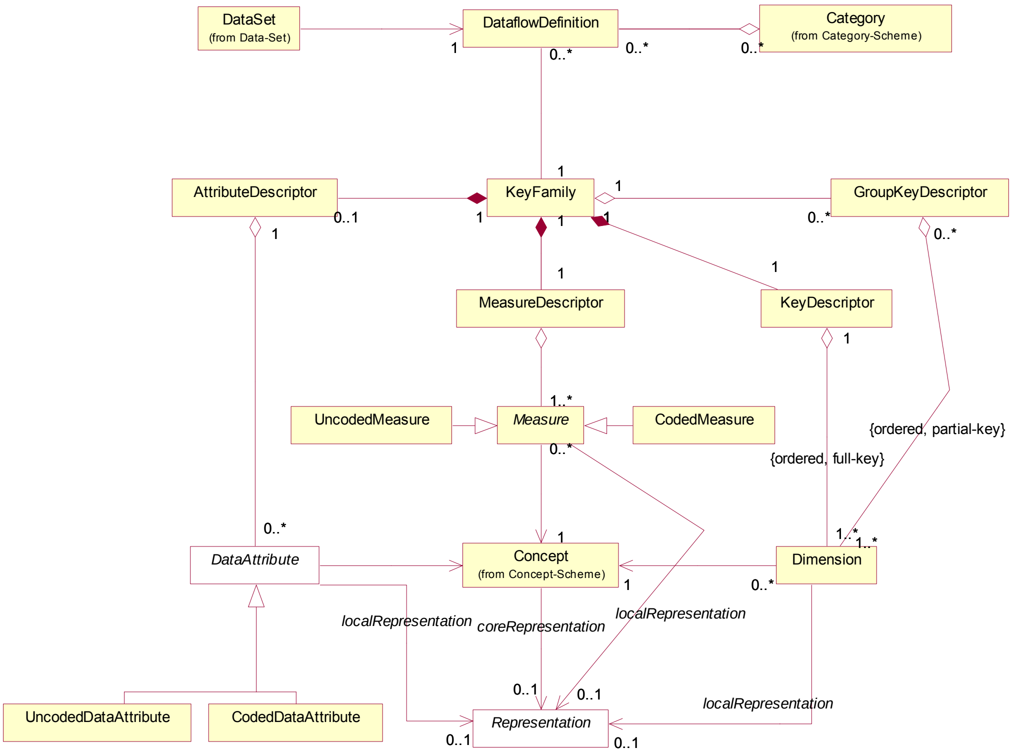


Figure 25: The model of the Key Family

Note that each of Dimension, Data Attribute, and Measure can take any type of valid Representation. The Key Family links to a Dataflow Definition. The details held for a Dataflow Definition is covered later, but essentially it defines metadata relating to a flow of data that is collected or disseminated such as periodicity of reporting and which organizations report data (the Data Set). Importantly, the Dataflow Definition can be linked to one or more "domain" Categories which "categorizes" the type of data (e.g. Balance of Payments, National Accounts, Population, Tourism, Insurance): this can be useful for supporting queries for data.

6.3.5.3 The Data Structure Definition Model and the Structure Pattern

The Data Structure Definition model is based on the Structure model from the SDMX Base: this means that nearly all of the classes are specializations (sub classes) of the (abstract) classes in the SDMX Base. The diagram below shows from which SDMX Base classes the Data Structure Definition model classes are inherited.

Note that all of Dimension, Measure, and Data Attribute have an association to a Concept Role (not shown on the diagram) which, if used, defines the role that the Concept used by the Dimension, Measure, or Data Attribute plays in this Data Structure Definition. Examples are: Frequency, Time, Primary Measure).

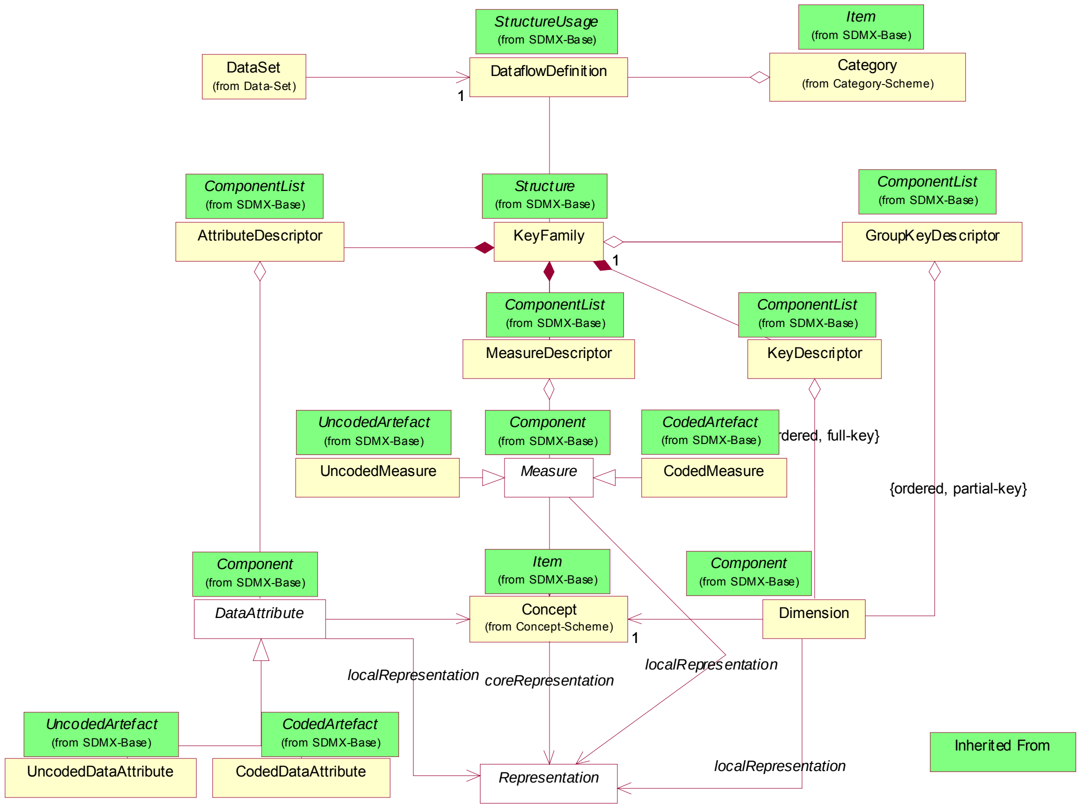


Figure 26: Derivation of the Data Structure Definition (Key Family) classes from the SDMX Base classes

Figure 12 shows the Structure pattern in the SDMX Base. The actual UML inheritance diagram of the Data Structure Definition (Key Family) classes is shown below.

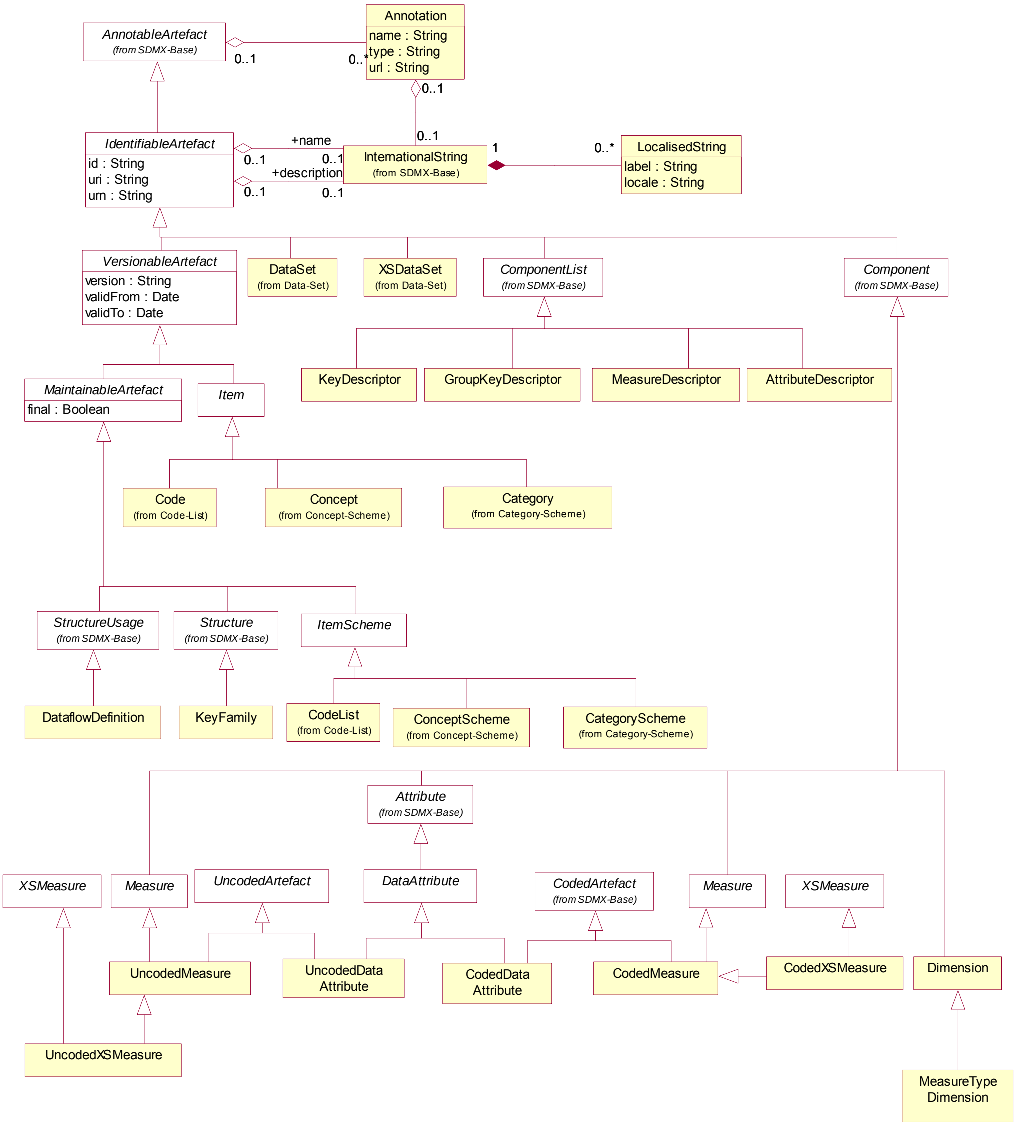


Figure 27: Inheritance of the classes in the Data Structure Definition (Key Family) model

The Code List, Concept, and Category are covered in other sections. It can be seen that the only artefact that is maintainable is the Key Family itself. Of the classes that remain, none is versionable, but all are identifiable. Note that the Measure and the Data Attribute have sub classes that inherit also from one of Coded Artefact or Uncoded Artefact, depending on whether it is coded or uncoded. A Dimension can also be coded or uncoded but specialisations of this have not been modeled.

6.3.5.4 Example Implementation of the Model

The following table shows a specification of a Data Structure Definition to support the example data set shown in Figure 15. Note that this Data Structure Definition is very simple. It is constructed purely for the purposes of this example and is not intended to be a template for key families for demographic data.

Dimensions - Key		
Type	**Concept**	**Representation/Code list**
Dimension (role is Frequency)	FREQ	Code List: CL_FREQ
Dimension	DEMOGRAPHIC_TYPE	Code List: CL_DEMOG_TYPE
Dimension	REGION	Code List: CL_REGION
Dimension	AGE_RANGE	Code List: CL_AGE_RANGE
Dimension (role is Time)	TIME	Date/Time
Measure		
OBS_VALUE (role is Primary Measure)		

Attributes			
Concept	**Assignment Status**	**Assignment Level**	**Representation/Code List**
OBS_STATUS	Mandatory	Observation	Code List: CL_OBS_STATUS
UNIT_OF_MEASURE	Mandatory	Series	Code List: CL_MEASURE_UNIT
TITLE	Mandatory	Data Set	Text
SOURCE	Conditional	Data Set	Text
PUBLICATION_DATE	Conditional	Data Set	Text

Figure 28: Table showing a Data Structure Definition

An example of a user interface of a tool which could maintain a Data Structure Definition (called Key Family in the tool) based on the Information Model constructs is shown below.

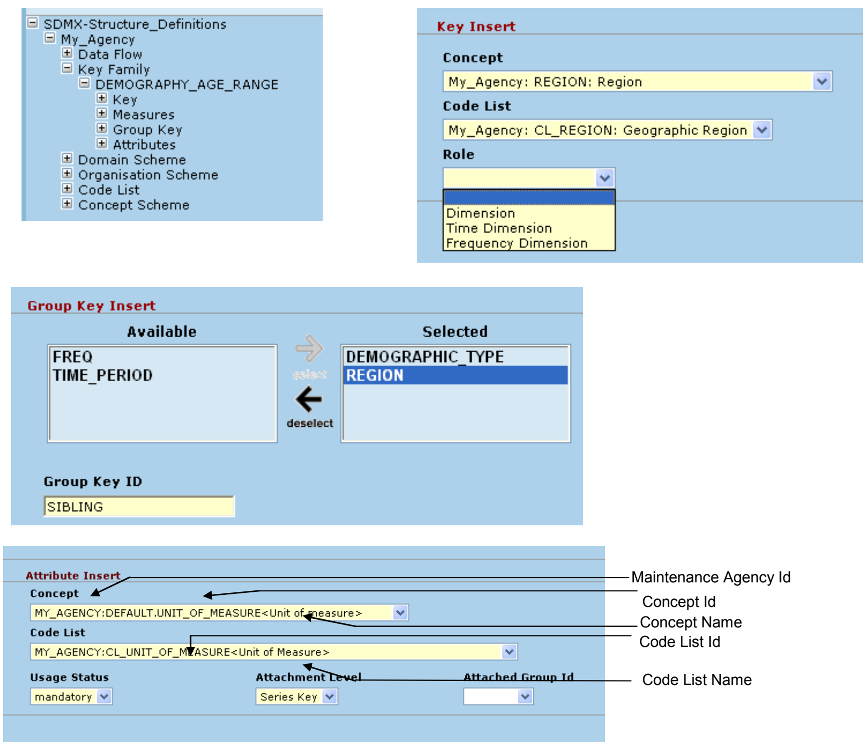


Figure 29: Example of a user interface for Key Family based on the SDMX-IM

Note the following:

- The Key Family is maintained by a maintenance agency (My_Agency)
- The Concepts used for Dimensions, Measures (not shown), and Attributes can be taken from any maintained Concept Scheme, and need not all be from the same agency/scheme.
- The Concept is identified by a unique combination of maintenance agency:concept scheme id:concept id
- The Code List is identified by a unique combination of maintenance agency:code list id
- The Group Key comprises a sub set of the Dimensions of the Key – the other Dimensions are assumed to be "wildcarded" and have the value ("all" of "not applicable")
- If the attachment level of the Attribute is "Group Key" then the Group Key Id is also necessary in order to identify the correct group.

6.3.6 Category Scheme, Data and Metadata Flow and Data Provider

6.3.6.1 Schematic

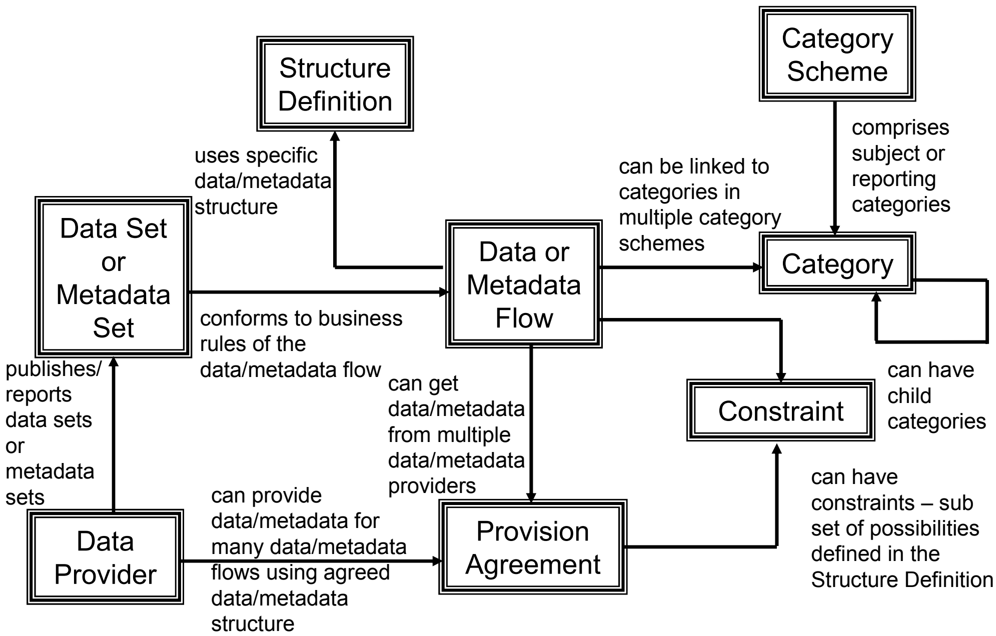


Figure 30: Schematic of data and metadata reporting

The diagram above depicts the essential characteristics supported in the model for reporting or publishing data and metadata. The pivot of this diagram is the Data or Metadata Flow. The Data or Metadata Flow is maintained by the organization that collects or “harvests” the data or metadata in order to use it or to publish it. The Data Flow is linked to a single Data Structure Definition (Structure Definition on the diagram). The Metadataflow is linked to a Metadata Structure Definition. The data or metadata for the Data or Metadata Flow may be provided by many Data Providers and any one Data Provider may report or publish data or metadata for many Data or Metadata Flows – typically a Data Provider may supply data or metadata for many topics or categories of statistical data. The Provision Agreement is not merely a convenient resolution of this many-to-many association, as the Data or Metadata Provider can apply Constraints on the scope of the data or metadata that can be supplied, in terms of key ranges or complete key sets. For instance, a Data Provider might supply data for a sub set of code values in any one of the dimensions comprising the Key: for example a typical sub set might be for only one country. This is known as “data provisioning constraints” and is not explained further in this document (interested readers will need to refer to the SDMX Information Model document for more information).

The Data or Metadata Flow may also be linked to one or more Topics in one or more Subject Matter Schemes. A Subject Matter Scheme (called Category Scheme in the model) provides a way of classifying data for collection, reporting, or publication. Typical schemes may comprise many high level categories such as financial, economic, health, tourism, transport, demography and each of these may be further segmented into lower level categories. Hence the Categories can be hierarchic. It is usual for a “drill down” search mechanism to be driven from such a scheme and so the Data and Metadata Flows that are Linked to the Category will lead to the Data and Metadata Sets and their publishers (Data Providers).

6.3.6.2 Mapping the Schematic to the Information Model

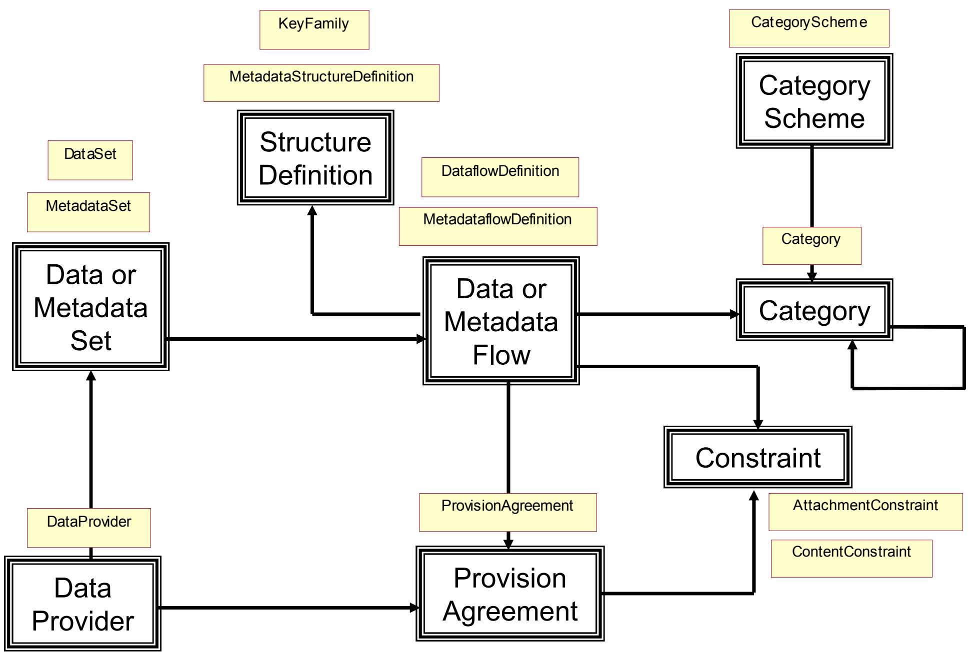


Figure 31: The map between the Data Flow and Metadata Flow schematic and the model classes

This diagram shows the model classes that represent the boxes in the schematic.

6.3.6.3 Category Scheme Model

The Category Scheme is a specialised form of Item scheme. The Items in the scheme can be hierarchic thus allowing the definition of hierarchic schemes. Figure 9 gives a simple schematic of such an Item Scheme.

6.3.6.4 Data Flow Model

The diagram below shows a simplified view of the actual model of the Data Flow Definition and associated classes. Note that many of the classes inherit from SDMX Base classes (this is shown below in the inheritance diagram) and so most of the associations shown are derived from the base associations.

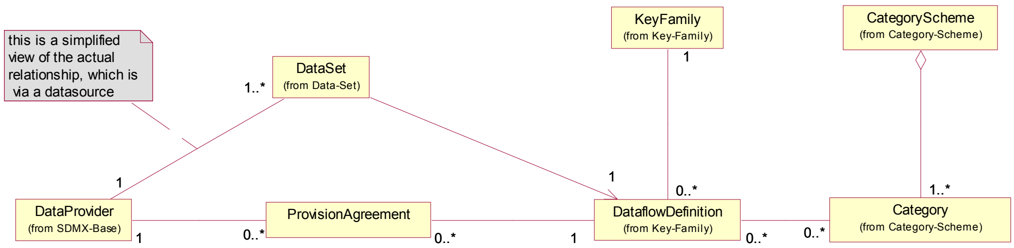


Figure 32: Simplified model of the Data Flow Definition

With the exception of the Data Set, all of the classes related to the Data Flow inherit from one of the base classes which give either versioning and identification, or just identification. The diagram below shows from which base class they are derived.

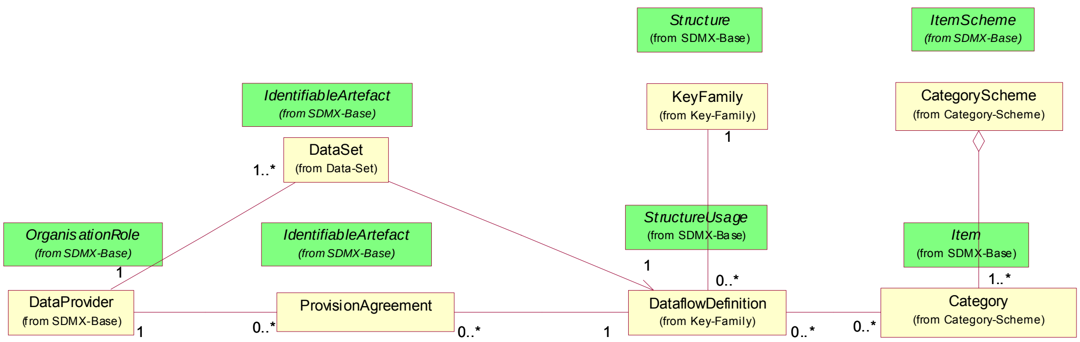


Figure 33: Schematic of the inheritance from the SDMX Base of the Data Flow classes

6.3.6.5 Metadata Flow Model

The diagram below shows a simplified view of the actual model of the Metadata Flow Definition and associated classes. Note that many of the classes inherit from SDMX Base classes (this is shown below in the inheritance diagram) and so most of the associations shown are derived from the base associations.

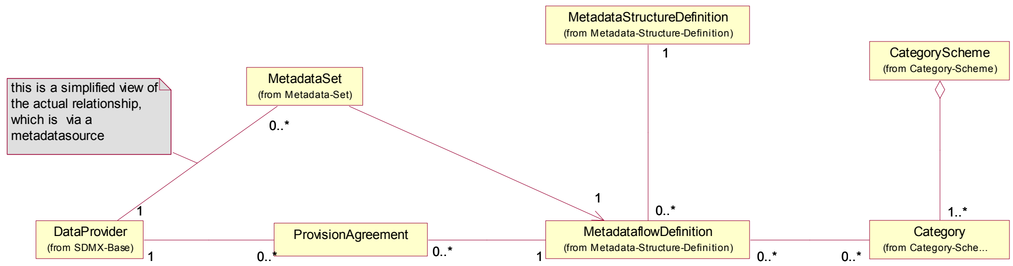


Figure 34: Simplified model of the Metadata Flow Definition

More detail on the Data Provider can be found in Section 6.2.6. The Category Scheme is explained later in this section.

With the exception of the Metadata Set, all of the classes related to the Metadata Flow inherit from one of the base classes which give either versioning and identification, or just identification. The diagram below shows from which base class they are derived.

It can be seen from this diagram that the same underlying pattern (the “Structure” pattern) is used for both data provisioning and metadata provisioning.

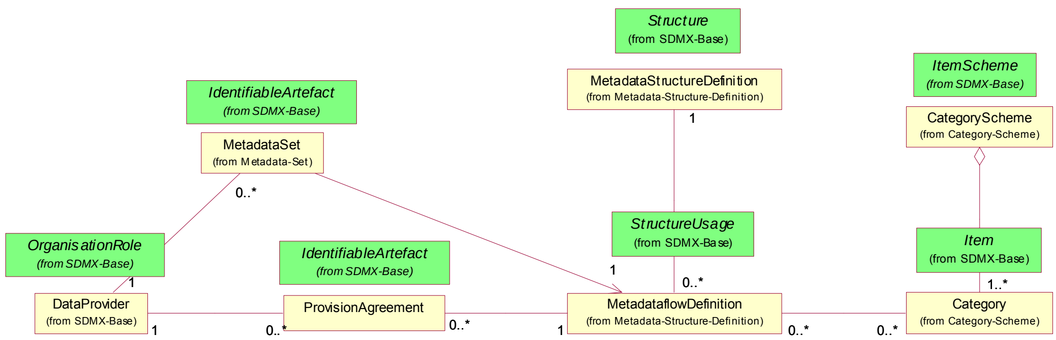


Figure 35: Schematic of the inheritance from the SDMX Base of the Metadata Flow classes

6.3.6.6 Data and Metadata Flow Inheritance

The full relationship diagram showing the inheritance from the base structure classes is shown below.

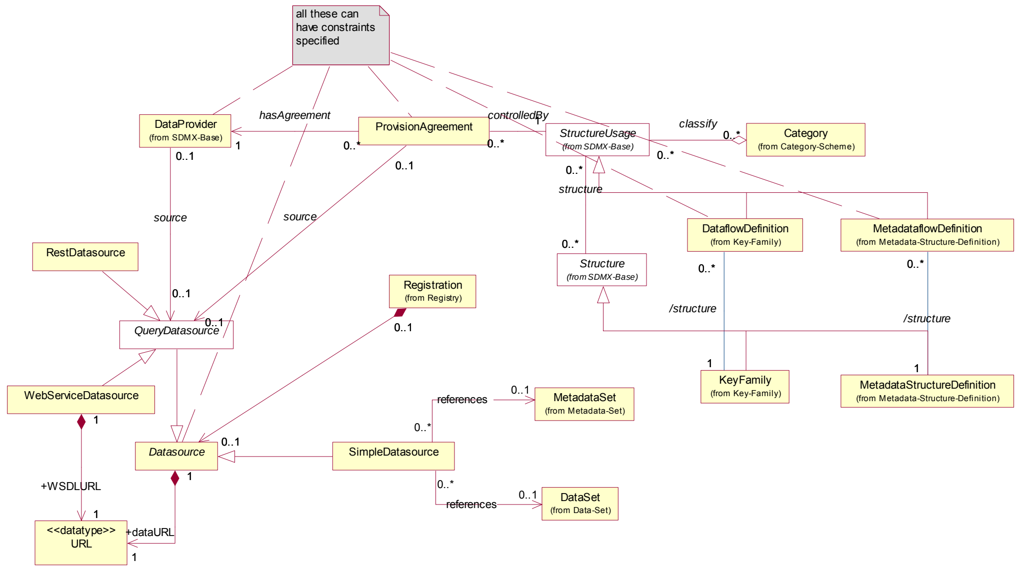


Figure 36: Detailed diagram of the Data Flow and Metadata Flow

This diagram shows also the classes that support data sources. The Datasource supports a registry centric scenario where data and metadata sets or entire databases or metadata repositories are registered. The "constraints" mechanism is used to define the content of these data sources – the Constraint is defined in terms of the applicable dates, keys, and component values, all of which can be used to aid search applications to find the relevant data and metadata. The Simple Datasource is an SDMX-ML formatted file (Data Set or Metadata Set) and which can be found at a web address. The Query Datasource is a service available over the web which can accept an SDMX-ML query and respond with an SDMX-ML Data Set or Metadata Set. The Query Datasource can use a simple REST interface or can use web services technology (Web Service Datasource).

6.3.7 Metadata Structure Definition

6.3.7.1 Overview

The Metadata Structure Definition identifies the structures to which metadata can be attached and defines the allowable content of that metadata. It is common for such metadata to be shareable amongst many artifacts to which it relates and it is often stored in a separate metadata repository and is referenced from the artifact to which it relates. This metadata is called "Reference Metadata" in SDMX. Reference Metadata is content metadata that gives more information about the artifact so as to make its interpretation more meaningful.

The SDM-IM allows reference metadata:

1. To be exchanged without the need to embed it within the object that it is describing.

2. To be stored separately from the object that it describes, yet be linked to it (example: an organization has a metadata repository which supports the dissemination of metadata resulting from metadata requests generated by systems or services that have access to the object for which the metadata pertains).

3. To be indexed to aid searching (example: a registry service can process a metadata report and extract structural information that allows it to catalogue the metadata in a way that will enable users to query for it).

In order to achieve this, the Metadata Structure Definition must have a mechanism for:

1. Identifying precisely what metadata is relevant (i.e. in terms of concepts) and rules for its usage (e.g. Mandatory/conditional), and value domain (e.g. code lists).

2. Defining precisely the context of the metadata in terms of the object type to which it is relevant, and the precise identity of the object.

Note that (1) is supported by a maintained Concept Scheme, and that (2) must also support the identification of lower level objects within a higher level structure (e.g. a Dimension in a Data structure definition, A Code in a Code List).

6.3.7.2 Schematic

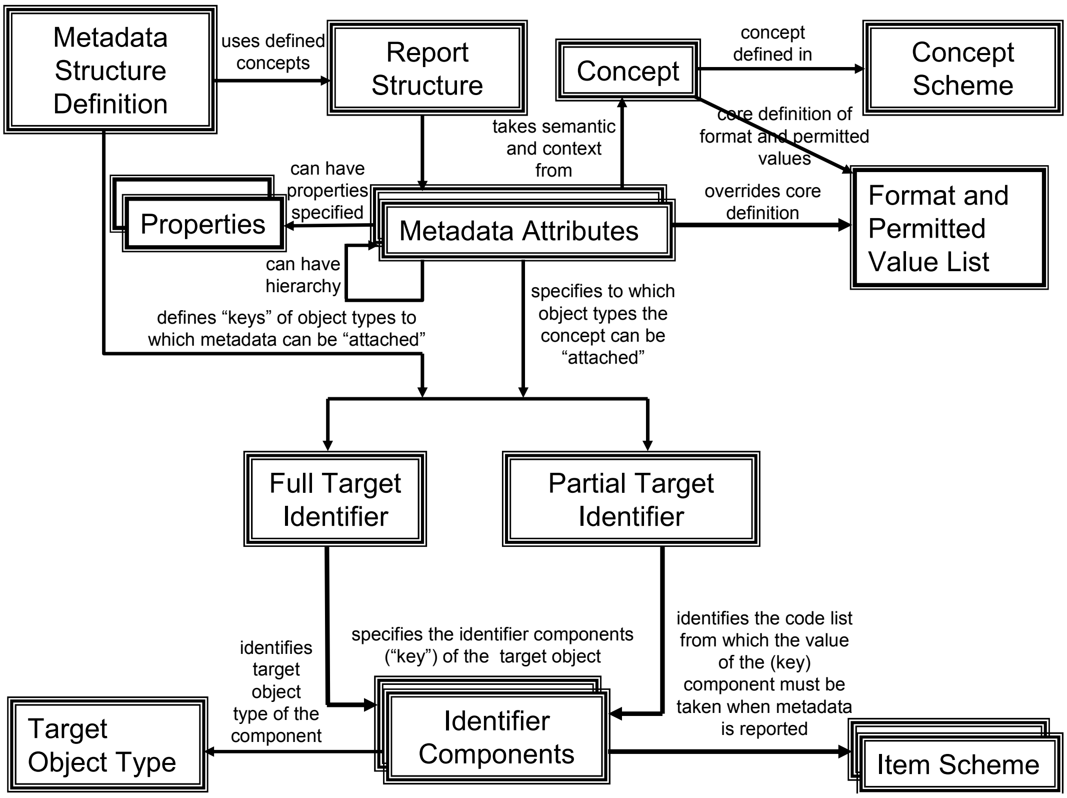


Figure 37: Schematic of the Metadata Structure Definition

Metadata Concepts are administered in a Concept Scheme. An example is the IMF Data Quality Assessment Framework.

In order to define a metadata reporting or dissemination environment it is necessary to define one or more Metadata Structure Definitions. A Metadata Structure Definition is similar in structure and intent to a Data Structure Definition: however, whereas the Data structure definition defines the structure of a data set, the Metadata Structure Definition defines the structure of a metadata set.

A Metadata Structure Definition comprises two fundamental parts:

The Object Type(s) to which metadata can be attached

A Metadata Structure Definition defines the components of the "key" to which metadata can be attached. The full key of the object is defined (Full Target Object Identifier), and optionally any number of partial keys can be defined (Partial Target Object Identifier). Each such partial key definition must be given a name so that it can be identified. Each key and partial key also identifies the object type that it is identifying (e.g. a dataflow, a data structure definition, a component in a data structure definition).

It can be seen that the object identification method used in the Metadata Structure Definition is indirect: the Metadata Attribute is "attached to" one or more of the Full Target Identifier or Partial Target Identifiers. The actual object type to which it can be attached and the way it is identified is specified within the Target and Partial Target Identifiers.

The object identifier comprises a number of Identifier Components. This is similar in concept to the Dimension in the key structure of a Data Structure Definition. Each such component must be linked to an Item Scheme. An Item Scheme in the SDM-IM is a "super class" and as such it has many sub classes, each sub class being a particular use of the Item Scheme and each having its own specific restrictions. A Code List is a sub class of Item Scheme: other sub classes are Concept Scheme and Category Scheme. Therefore, in a Metadata Structure Definition the value of the attachment key concept can be taken from a wider variety of "lists" than for a Dimension in a Data Structure Definition (which, if coded, is restricted to a Code List).

It can be seen from the explanation above that the underlying structure of the Metadata Structure Definition is similar to the Data Structure Definition. This underlying structure is the "Structure" pattern in the SDMX Base model and the map to this pattern is shown later in this section.

A Report Structure

This defines both the structure of the metadata report in terms of a hierarchy of Metadata Attributes, and the rules by which the Metadata Attributes are used in the structure definition.

Each Metadata Attribute links to a Metadata Concept (examples: Source, Periodicity). These concepts can be taken from one or more Concept Schemes. The following information is defined in the Metadata Structure Definition for each Metadata Attribute:

- the identity of the Metadata Concept (example: Periodicity in the Metadata Concept Scheme called IMF_DQAF maintained by the IMF)
- format and representation if these are different from that defined for the Concept

- the Properties that may be reported. This is an extensible mechanism that allows the definition of a sub structure to the concept when it is reported (e.g. there may be a URL reported and it is necessary for an application to know that it is a URL)
- whether the reporting of the attribute is mandatory or conditional
- the identity of the key that defines both to what object or structure the metadata reported is to be "attached", and the components comprising the identifier or "key" of the object (this is similar in concept to the "attachment level" of attributes in the Data structure definition)

6.3.7.3 Mapping the Schematic to the Information Model

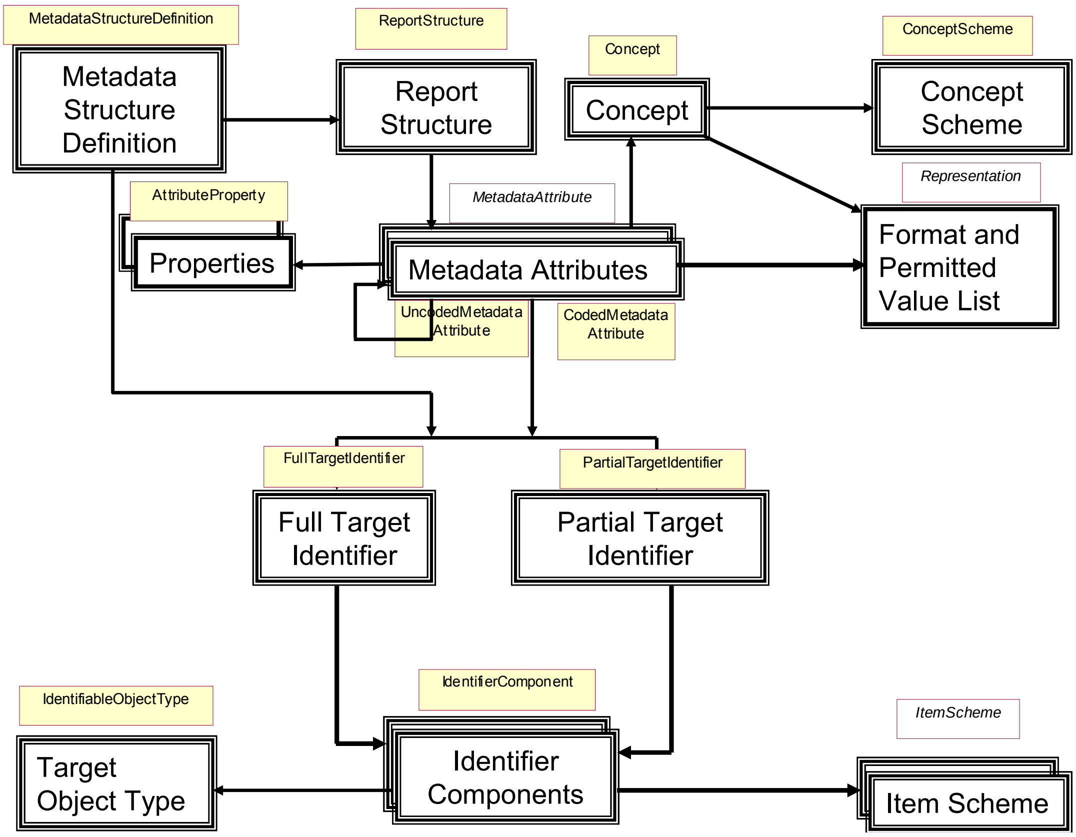
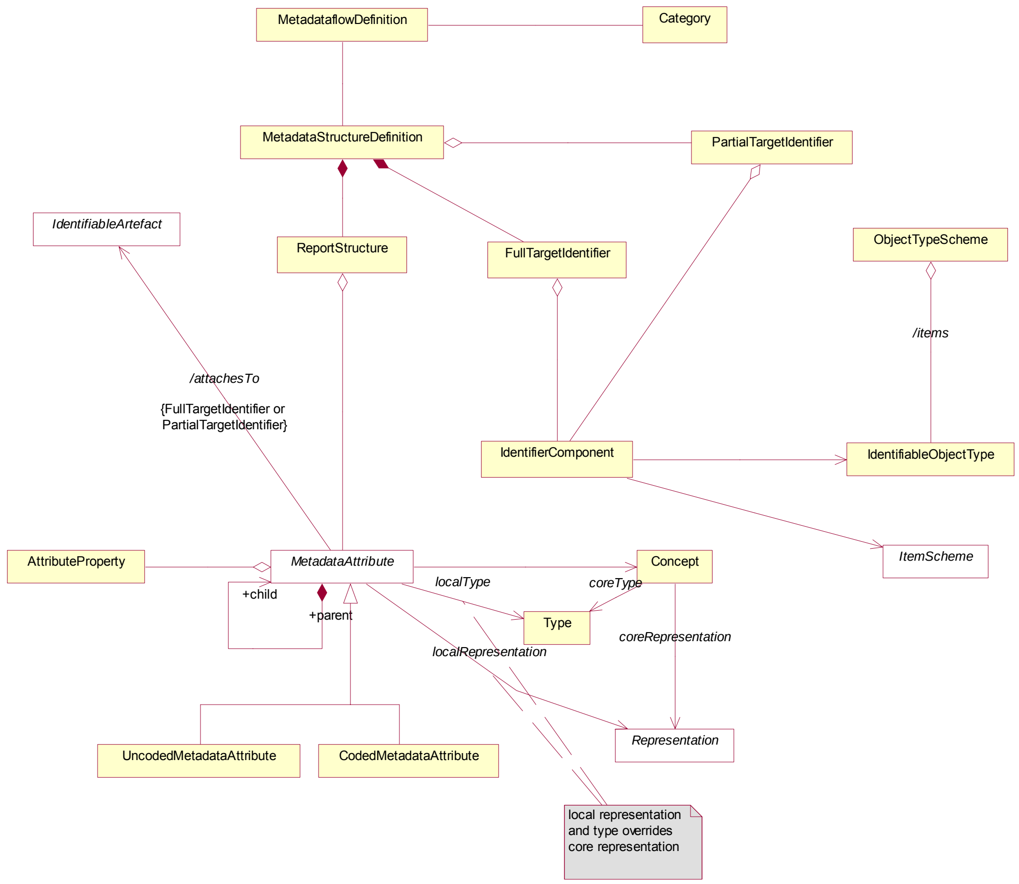


Figure 38: The map between the Metadata Structure schematic and the model classes

The actual model of the Metadata Structure Definition is shown below.



1.

Figure 39: The model of the Metadata Structure Definition

6.3.7.4 The Metadata Structure Definition Model and the Structure Pattern

The Metadata Structure Definition uses the same underlying pattern as the Data Structure Definition (Key Family), but in a slightly different way:

- Properties are allowed to be specified for the Metadata Attributes
- There are no Measures
- There is an additional association between the classes inheriting from "ComponentList" (Target Identifier and Partial Target Identifier) and "Component" (Identifier Component): this association is to the Identifiable Object Type which identifies the object type to which metadata may be attached
- The Identifier Component is not associated with a Concept but with an object type that represents a class in the SMMX-IM

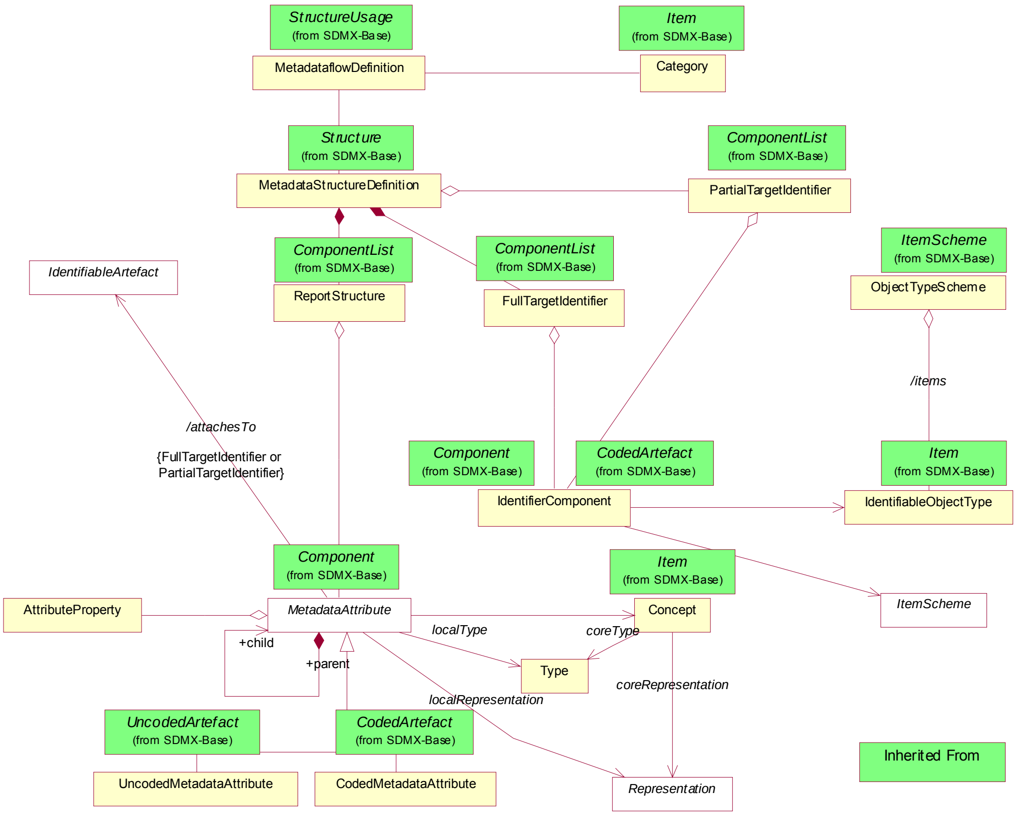
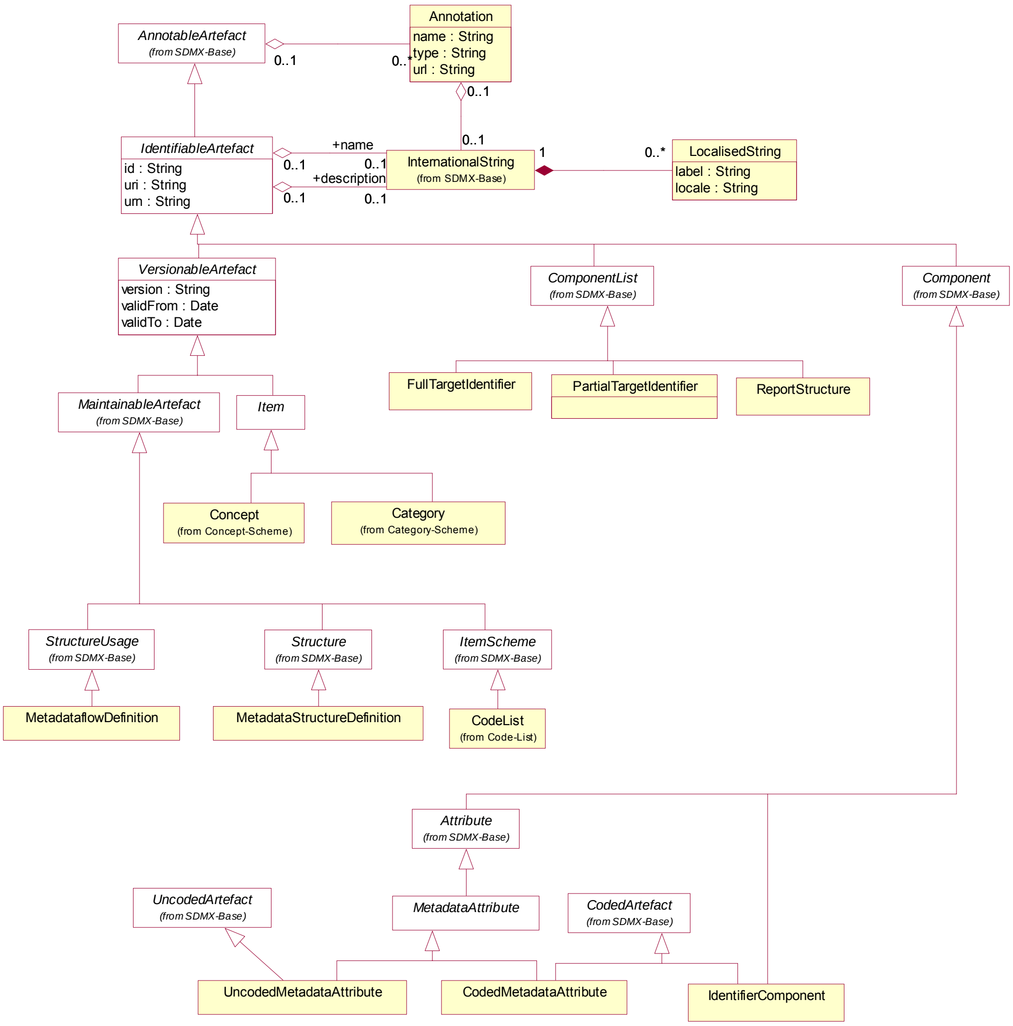


Figure 40: Derivation of the Metadata Structure Definition classes from the SDMX Base classes

Figure 11 shows the Structure pattern in the SDMX Base. The actual UML inheritance diagram of the Metadata Structure Definition classes from the SDMX Base classes is shown below.



2447

2448 **Figure 41: Inheritance of the classes in the Metadata Structure Definition model**

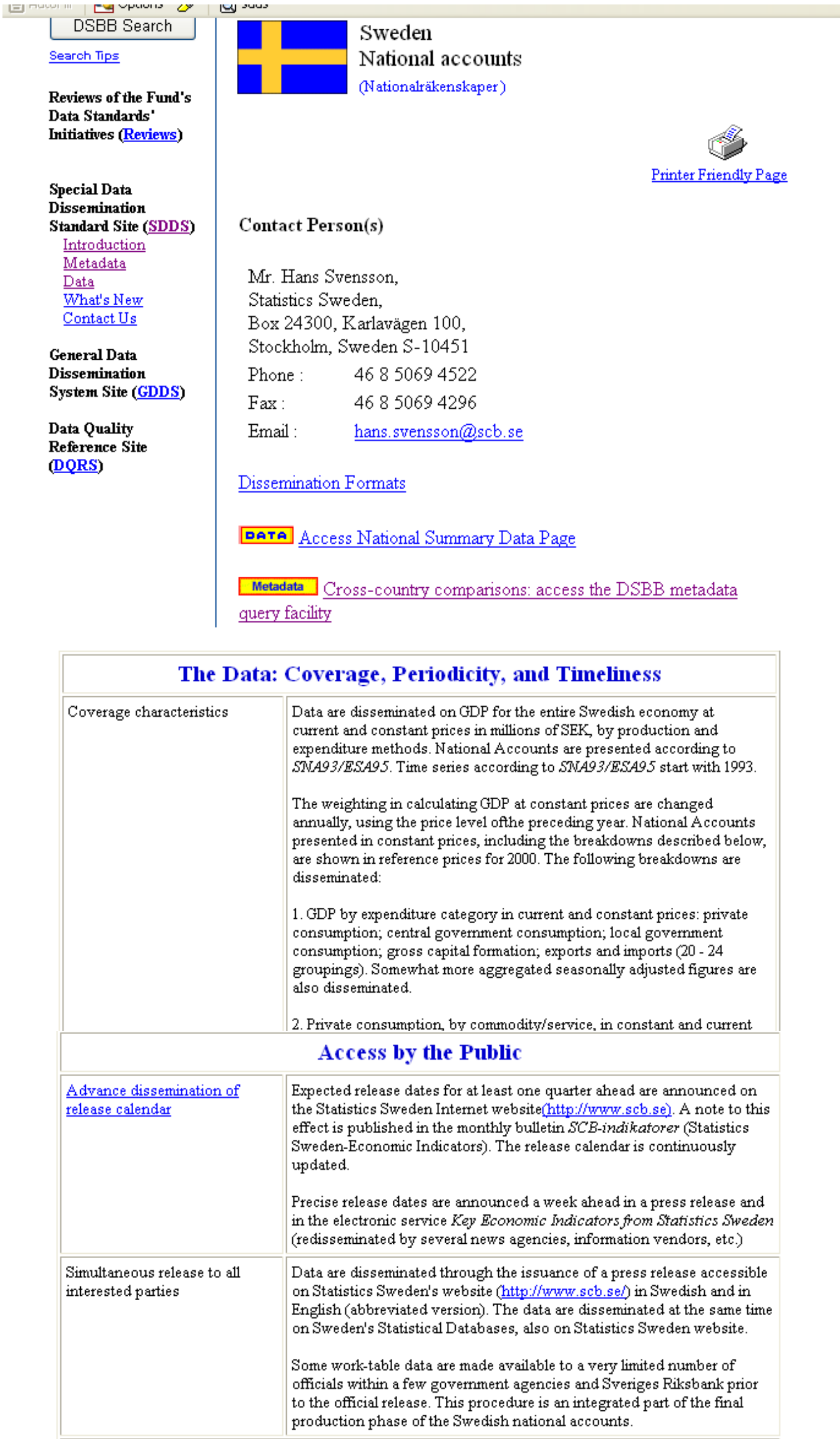
2449

2450

6.3.7.5 Example Implementation of the Model – Quality Metadata

6.3.7.5.1 SDDS example metadata

The following extract from a web page is taken from the IMF SDDS.


DSBB Search

Search Tips

Reviews of the Fund's Data Standards' Initiatives (Reviews)

Special Data Dissemination Standard Site (SDDS)
- Introduction
- Metadata
- Data
- What's New
- Contact Us

General Data Dissemination System Site (GDDS)

Data Quality Reference Site (DQRS)

Sweden
National accounts
(Nationalräkenskaper)

Printer Friendly Page

Contact Person(s)

Mr. Hans Svensson,
Statistics Sweden,
Box 24300, Karlavägen 100,
Stockholm, Sweden S-10451

Phone : 46 8 5069 4522
Fax : 46 8 5069 4296
Email : hans.svensson@scb.se

Dissemination Formats

DATA Access National Summary Data Page

Metadata Cross-country comparisons: access the DSBB metadata query facility

The Data: Coverage, Periodicity, and Timeliness	
Coverage characteristics	Data are disseminated on GDP for the entire Swedish economy at current and constant prices in millions of SEK, by production and expenditure methods. National Accounts are presented according to *SNA93/ESA95*. Time series according to *SNA93/ESA95* start with 1993. The weighting in calculating GDP at constant prices are changed annually, using the price level ofthe preceding year. National Accounts presented in constant prices, including the breakdowns described below, are shown in reference prices for 2000. The following breakdowns are disseminated: 1. GDP by expenditure category in current and constant prices: private consumption; central government consumption; local government consumption; gross capital formation; exports and imports (20 - 24 groupings). Somewhat more aggregated seasonally adjusted figures are also disseminated. 2. Private consumption, by commodity/service, in constant and current
Access by the Public	
Advance dissemination of release calendar	Expected release dates for at least one quarter ahead are announced on the Statistics Sweden Internet website(http://www.scb.se). A note to this effect is published in the monthly bulletin *SCB-indikatorer* (Statistics Sweden-Economic Indicators). The release calendar is continuously updated. Precise release dates are announced a week ahead in a press release and in the electronic service *Key Economic Indicators from Statistics Sweden* (redisseminated by several news agencies, information vendors, etc.)
Simultaneous release to all interested parties	Data are disseminated through the issuance of a press release accessible on Statistics Sweden's website (http://www.scb.se/) in Swedish and in English (abbreviated version). The data are disseminated at the same time on Sweden's Statistical Databases, also on Statistics Sweden website. Some work-table data are made available to a very limited number of officials within a few government agencies and Sveriges Riksbank prior to the official release. This procedure is an integrated part of the final production phase of the Swedish national accounts.

Figure 42: Example reference metadata

The characteristics of the SDDS reference metadata from the point of view of a defining a Metadata Structure Definition are:

1. The metadata concepts are maintained in a Concept Scheme.

2. Three "levels" of reporting are supported:

3. Dataflow Agreement - this is the most detailed level where a Data Provider is reporting data according a Dataflow

4. "Data category" (the example shows National Accounts) – in terms of the SDMX-IM this relates the (data) Category in a Category Scheme to the Data Provider

5. "Agency" – this is data category independent and refers to all data categories – in SDMX-IM terms this is a partial key of the "data category" level where only the identity of the Data Provider is required

6. Some Metadata Concepts can be reported at just one level, other can be reported at either or both levels.

7. The text represented on the web page can be built from the content of several concepts reported by the Data Provider.

6.3.7.5.2 The Identifiers

Note that the method of identifying the object type to which the metadata can be attached for artefacts (classes) that are in the SDMX-IM is fixed – there is only one way of identifying the artifacts and this is according to the relationships in the SDMX-IM. In essence it is as follows:

The full identifier for an artifact is

For objects that are maintained in a scheme (such as a Category Scheme, Organisation Scheme, Code List etc.) the Identifier Components are:

```
Contained object scheme id
Contained object id
```

For objects that are not contained in a scheme (such as Dataflow) the Identifier Component is

```
Object Id.
```

Note that the maintenance agency is not an Identifier Component as the "Item List" of allowable values constrains the possible values to those maintained by a specific maintenance agency.

The diagram below shows the way the Provision Agreement and the Data Provider are linked to a (data) Category.

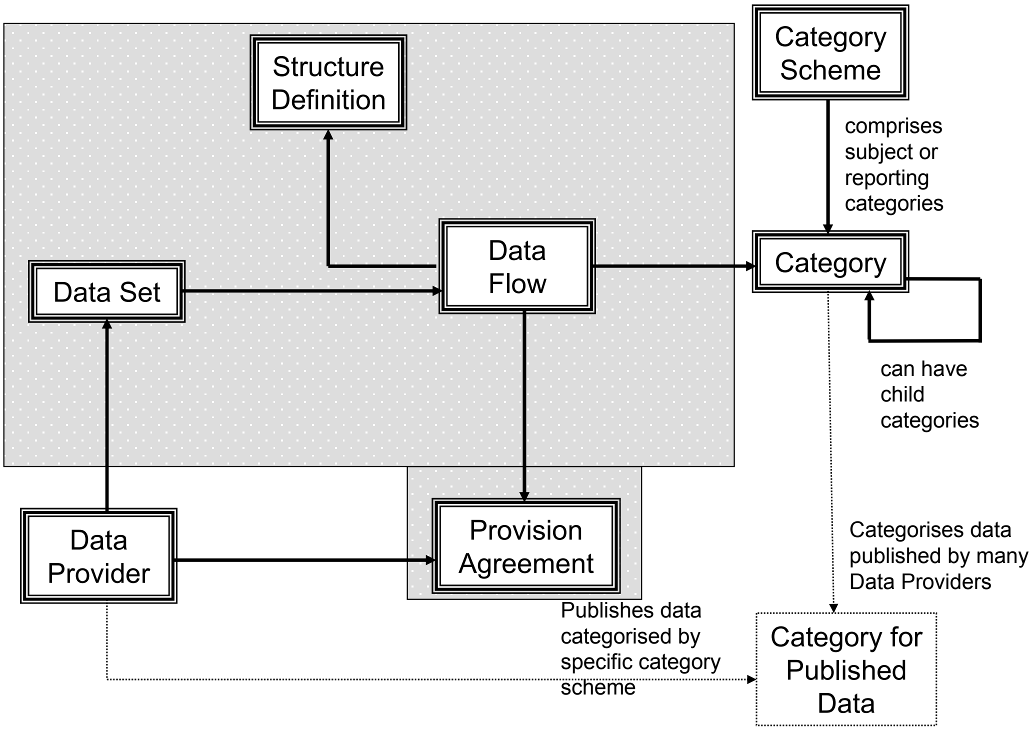


Figure 43: Data Category, Dataflow and Data Provider in the SDMX-IM

In order to construct the Identifier Components of the Target Identifier and the Partial Target Identifier, it is necessary to determine which component object types identify uniquely the target object type and which Item Scheme contains the list of valid identifiers.

The SDDS reference metadata linked to a (data) Category are metadata about the publication or provision of data by a Data Provider. However, even though the Data Provider may report this data according to a Dataflow and Provision Agreement this is not the concern of the SDDS reference metadata (and so is shown in grey on the figure above), which must be attached to the appropriate Category in the Category Scheme used by the SDDS. In the schematic above this is called “Category for Published Data”. There is no such class in the SDMX-IM as there is no require,ent in the SDMX-IM for such a class. However, providing the specific object to which the metadata is to be attached can be identified, the Reference Metadata mechanism can be used. In the case of the SDDS the object “Category for Published Data” is a union of the Category Id and the Data Provider Id.

The definition of the Identifier Components comprising the Full Target Identifier and the Partial Target Identifier for the SDDS Metadata Structure Definition is defined in the table below. Note that the Full Target Identifier is a list of all possible object types that can comprise components in a partial key, and the relevant Item Scheme that contains the list of valid identifiers for the object type. The Partial Target Identifier references a sub set of these specific object types to which metadata may be attached.

Identifier Type	Target Identifier Object Type	Target Component Object Type	Identified by	Agency:Item Scheme
Full Target Identifier id = CATEGORY				
		Category	Category Scheme Id Category Id	IMF:SDDS_ CATEGORY_ SCHEME
		Data Provider	Data Provider Id	IMF:DATA_ PROVIDER
Partial Target Identifier id = AGENCY	Data Provider			
		Data Provider		

Figure 44: Table of IdentifiersConcepts

The following extract of a Concept Scheme provides the Concepts. Note that the concepts presented here are a flat list. Hierarchical concept schemes are supported in the SDMX-IM but such schemes must represent the true semantic hierarchy of concepts. A reporting hierarchy is defined in the Metadata Structure Definition as a hierarchy of Metadata Attributes. It is possible, of course, that this hierarchy of Metadata Attributes follows exactly the hierarchy in a Concept Scheme.

Concept Scheme: DQAF		**Maintenance Agency: IMF**	
Concept Id	**Name**	**Description**	**Representation**
DATA	Source data	Describes the data collection programs, its comprehensiveness and how it takes into account country-specific conditions; describes how source data ….	text
ACCESS	Accessibility	Describes how statistics are presented (text, tables, and charts); describes the dissemination media and formats; describes the policy regarding the release of statistics according to a preannounced schedule ….	text
COVERAGE	Coverage	Specifies the population from which	text

Concept Scheme: DQAF		Maintenance Agency: IMF	
Concept Id	**Name**	**Description**	**Representation**
		observations for a particular topic can be drawn.	
ADR	Advance Dissemination of Release Calendar	A general statement on the schedule of release of data.	text
S_RELEASE	Simultaneous Release	The dissemination of statistical data to all interested parties at the same time.	text

Figure 45: Example list of Concepts

6.3.7.5.3 Metadata Attributes

Metadata Attribute (Report) Structure Specification			
Level One Attribute	**Level Two Attribute**	**Attachment**	**Usage Status**
IMF:DQAF.DATA			
	IMF:DQAF.COVERAGE	Partial Target Identifier id = CATEGORY	Mandatory
IMF:DQAF.ACCESS			
	IMF:DQAF.ADR	Partial Target Identifier id = AGENCY	Mandatory
	IMF:DQAF.S_RELEASE	Partial Target Identifier id = AGENCY	Mandatory

Figure 46: Example structure specification of Metadata Attributes

Note that in the Metadata Set the Metadata Attribute is identified by its full hierarchic identifier, which does not include the Concept Scheme (e.g. in the table above an example would be DATA/COVERAGE). Therefore, it is necessary to ensure that all such hierarchic identifiers are unique in the set of Metadata Attributes.

6.3.7.5.4 *The Example Structure and the Model*

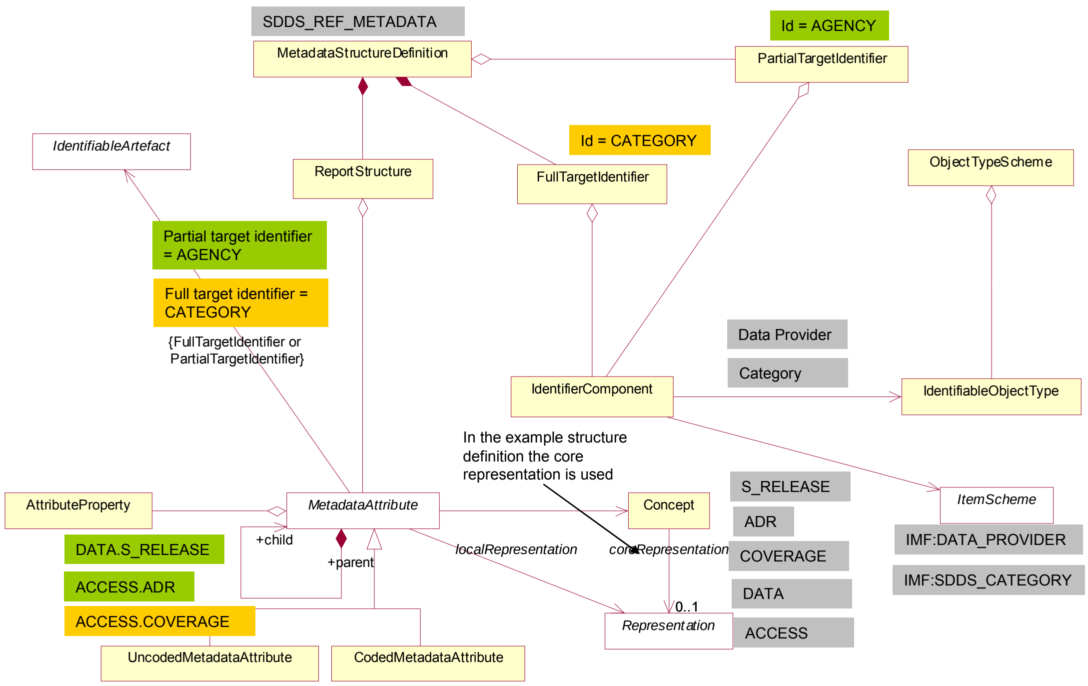


Figure 47: Example metadata structure mapped to the model classes

The identifier of the Metadata Structure Definition is SDDS_REF_METADATA. The Full Target Identifier has two components, each of which identifies an object type: Data Provider; Category. The Data Provider must be taken from the Item Scheme DATA_PROVIDER maintained by the IMF, and the Category must be taken from the Item Scheme SDDS_CATEGORY maintained by the IMF. Note that each of these schemes is a different subclass of Item Scheme – DATA_PROVIDER is an Organisation Scheme, and SDDS_CATEGORY is a Category Scheme. The Full Target Identifier is called “CATEGORY”.

One Partial Target Identifier is specified with the identifier of “AGENCY”, and this identifies the Data Provider and comprises just one Identifier Component – the Data Provider. Note that all the Identifier Components in the Partial Target Identifiers must be Identifier Components of the Full Target Identifier, and the Item Scheme used must be the same as that specified for the Target Identifier. In other words, each Identifier Component of the Partial Target Identifier is a reference to an Identifier Component of the Full Target Identifier.

Three Metadata Attributes are specified together with the hierarchy of the report structure. The COVERAGE is specified as being “attachable” to (in the case of SDDS this means “reported for”) the Full Target Identifier with the Id of “CATEGORY”, whilst the ADR and S_RELEASE are specified as being attachable to the Partial Target Identifier with the Id of “AGENCY”.

6.3.7.6 Example Implementation of the Model – SDDS Contact Metadata

6.3.7.6.1 SDDS Contacts

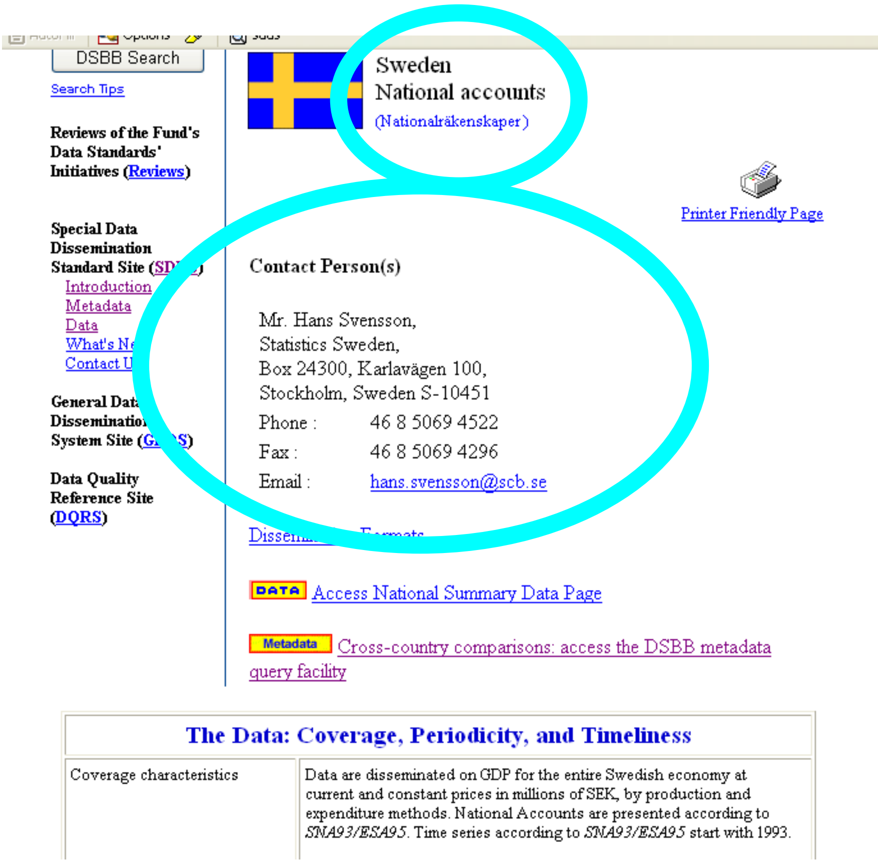


Figure 48: Example contact metadata

6.3.7.6.2 The Identifiers

The Full Target Identifier and Partial Target Identifier are similar to those for the SDDS reference metadata, but there is an additional Identifier Component – the Concept. This is required because sometimes the contact person is responsible for one or sub set of the SDDS reported concepts.

Identifier Type	**Target Identifier Object Type[1]**	**Target Component Object Type**	**Identified by**	**Agency: Item Scheme**
Full Target Identifier id = CONCEPT				
		Category	Category Id	IMF:SDDS_ CATEGORY_ SCHEME
		Data Provider	Data Provider Id	IMF:DATA_ PROVIDER
		Concept	Concept Id	IMF:DQAF
Partial Target Identifier id = CATEGORY	(all concepts, specific category)			
		Category		
		Data Provider		
Partial Target Identifier id = AGENCY	Data Provider (all concepts, all categories			
		Data Provider		

[1] The Target Identifier Object Type is not necessarily on the SDMX-IM but will be on the metadata model of the recipient of the metadata. Therefore the object type is not identified in this example but could be identified if required according to an Object Type Scheme maintained by the recipient that described the metadata system of the recipient. The semantics of the identifiers are:

Full Target Identifier (CONCEPT) – identifies a specific concept, category, and data provider(e.g. CONFIDENTIALITY for BOP for SCB Sweden)

Partial Target Identifier (CATEGORY) – identifies all concepts for a specific category for a data provider (this is known as the Country Contact in SDDS)

Partial Target Identifier (AGENCY) – identifies a data provider (Country Contact for SDDS in general)

6.3.7.6.3 Concepts

The contact information concepts shown below have semantic hierarchy where the PRIMARY_CONTACT is a specialised form of CONTACT. The other contact concepts, such as name, address, telephone, fax number, and e-mail address are at the same hierarchic level as CONTACT. The reporting hierarchy of these concepts is specified in the Report Structure of the Metadata Structure Definition.

Concept Scheme: CONTACTS		Maintenance Agency: IMF	
Concept Id	**Name**	**Description**	**Representation**
CONTACT			
PRIMARY_CONTACT			
NAME			text
ADDRESS			text
TELEPHONE			text
E-MAIL			text
CONTACT			

Figure 49: List of Contact concepts

6.3.7.6.4 Metadata Attributes

Metadata Attribute (Report) Structure Specification			
Level One Attribute	**Level Two Attribute**	**Attachment**	**Usage Status**
IMF:CONTACTS. PRIMARY_CONTACT		Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.PRIMARY_ CONTACT.NAME	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.PRIMARY_ CONTACT.ADDRESS	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.PRIMARY_ CONTACT. TELEPHONE	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional

Metadata Attribute (Report) Structure Specification			
Level One Attribute	**Level Two Attribute**	**Attachment**	**Usage Status**
	IMF:CONTACTS.PRIMARY_ CONTACT.E-MAIL	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
IMF:CONTACTS. CONTACT		Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.CONTACT. NAME	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.CONTACT. ADDRESS	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.CONTACT. TELEPHONE	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional
	IMF:CONTACTS.CONTACT. E-MAIL	Full Target Identifier Partial Target Identifier id = AGENCY and CATEGORY	Conditional

6.3.7.6.5 *The example Structure and the Model*

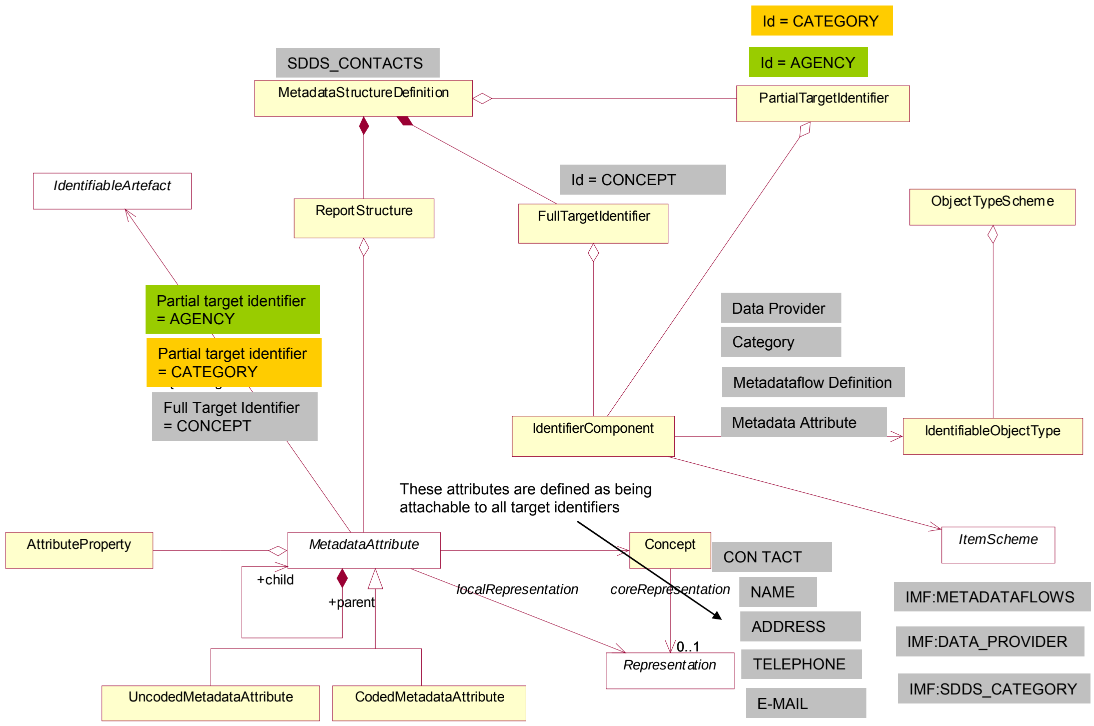


Figure 50: Example contact metadata structure mapped to the model classes

The Concepts comprise the concepts used for contact information. These are structured in a hierarchy – Contact has sub concepts of Name, Address etc. Primary contact concepts are not shown on the diagram.

All of the Metadata Attributes are specified as being attachable to all of the target identifiers (Full Target Identifier and both Partial Target Identifiers). In essence, this means that when a contact is reported in a metadata set the details can be specific to a category and a concept for the data provider (e.g. SDDS_METADATA, SCB, NAC, DATA.COVERAGE), or specific to a category for all concepts (e.g. SDDS_METADATA, SCB, NAC,) - in this case the Partial Key is CATEGORY - or they can be for all categories, all concepts (e.g. SDDS_METADATA, SCB) – in this case the Partial Key is AGENCY. Note that in all cases it is necessary to identity the Metadataflow Definition (called SDDS_METADATA in this example) as the contact is for the reporting of SDDS metadata (and not some other metadata that may be reported). Also, this will enable an application to validate the Concepts which should exist in one of the Report Structures of the Metadata Structure Definition linked to the Metadataflow Definition.

6.3.7.7 Example Implementation of the Model – Data Structure Definition Attached Metadata

In this example the Metadata Structure Definition defines the structure to attach metadata to a Key Family or Key Family components. The Report Structure attaches a Metadata Attribute to a Key Family.

6.3.7.7.1 The Identifiers

Identifier Type	**Target Identifier Object Type**	**Target Component Object Type**	**Identified by**	**Agency:Item Scheme**
Full Target Identifier Id = COMPONENT	Component			
		Key_Family	Key Family Id	ONS:KEY_FAMILIES
		Component_List	ComponentList Id	SDMX:COMPONENT_LIST (this will contain, amongst others, KeyDescriptor, MeasureDescriptor, AttributeDescriptor)
		Component	Concept Id	The Item Scheme is the Concept Scheme specified in the Key Family Definition. It is not referenced in this structure definition as the identity f the scheme is defined in the Key family Definition.
Partial Target Identifier id = KEY_FAMILY	Key Family			
		Key_Family		

Note that this metadata structure definition is confined to key family structure object types. It would be possible to define a metadata structure definition that included Code List and Concept Scheme. In this definition The Full Target Identifier would then describe the scope of the metadata structure definition but in itself would not identify a specific object type. It would include the following Target Component Types: Key Family; Component List; Component; Code List; Code; Concept Scheme; Concept. There would be six Partial Target Identifiers each of which would identify one of: Key Family; Component; Code List, Code; Concept Scheme; Concept.

6.3.7.7.2 Concepts

Concept Scheme: CORE_CONCEPTS		Maintenance Agency: SDMX	
Concept Id	**Name**	**Description**	**Representation**
COVERAGE	Coverage	Specifies the population from which observations for a particular topic can be drawn.	text

6.3.7.7.3 The Example Structure and the Model

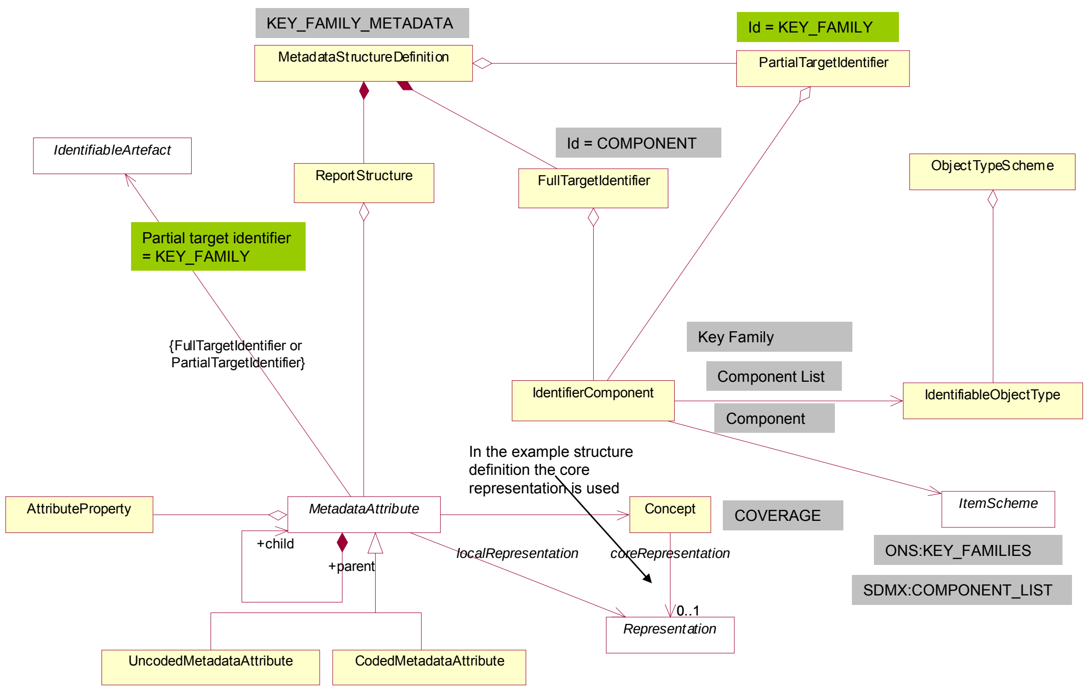


Figure 51: Schematic of a Metadata Structure Definition for a Key Family

The identifier of the Metadata Structure Definition is KEY_FAMILY_METADATA. The full Target Identifier identifies a Key Family Component (e.g. a specific dimension (such as COUNTRY)). A Partial Target Identifier is defined which identifies the Key Family. The Id of this partial identifier is called KEY_FAMILY.

One Metadata Attribute is specified in the Report Structure – COVERAGE. This is specified as being attached to the Partial Target Identifier (i.e. Key Family).

6.3.8 Hierarchical Code Scheme

6.3.8.1 Building Blocks

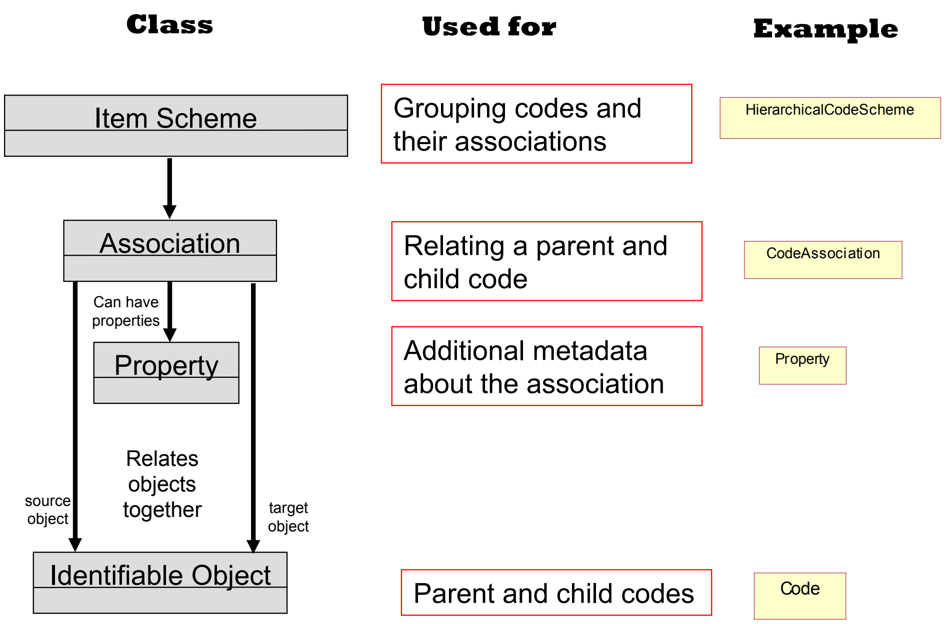


Figure 52: SDMX-IM and hierarchical code schemes

The fundamental constructs to support the hierarchical code scheme use the same building blocks as the Association. In the Hierarchical Code Scheme this is called a Code Association. The various Hierarchies are defined by grouping relevant Code Associations.

The way the Hierarchical Code Scheme is structured is shown in the diagram below.

6.3.8.2 Model Schematic and Class Diagram

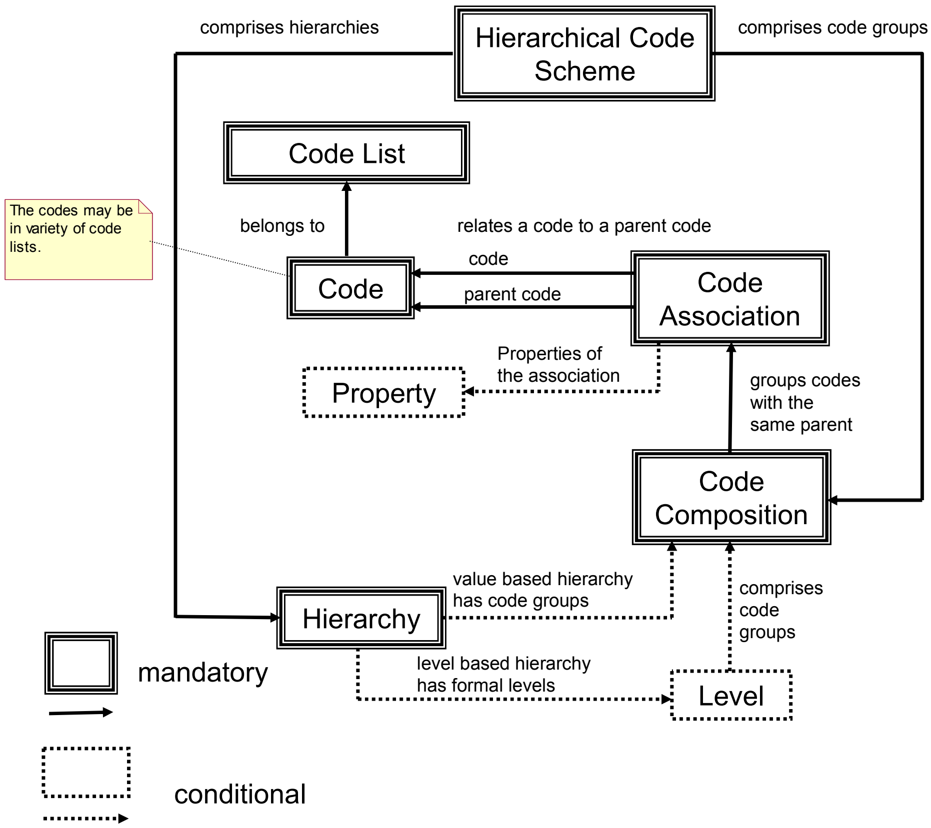


Figure 53: Schematic of the model for the hierarchical code scheme

The actual UML class diagram is very similar to the schematic drawn above and so the explanation is given following the UML diagram.

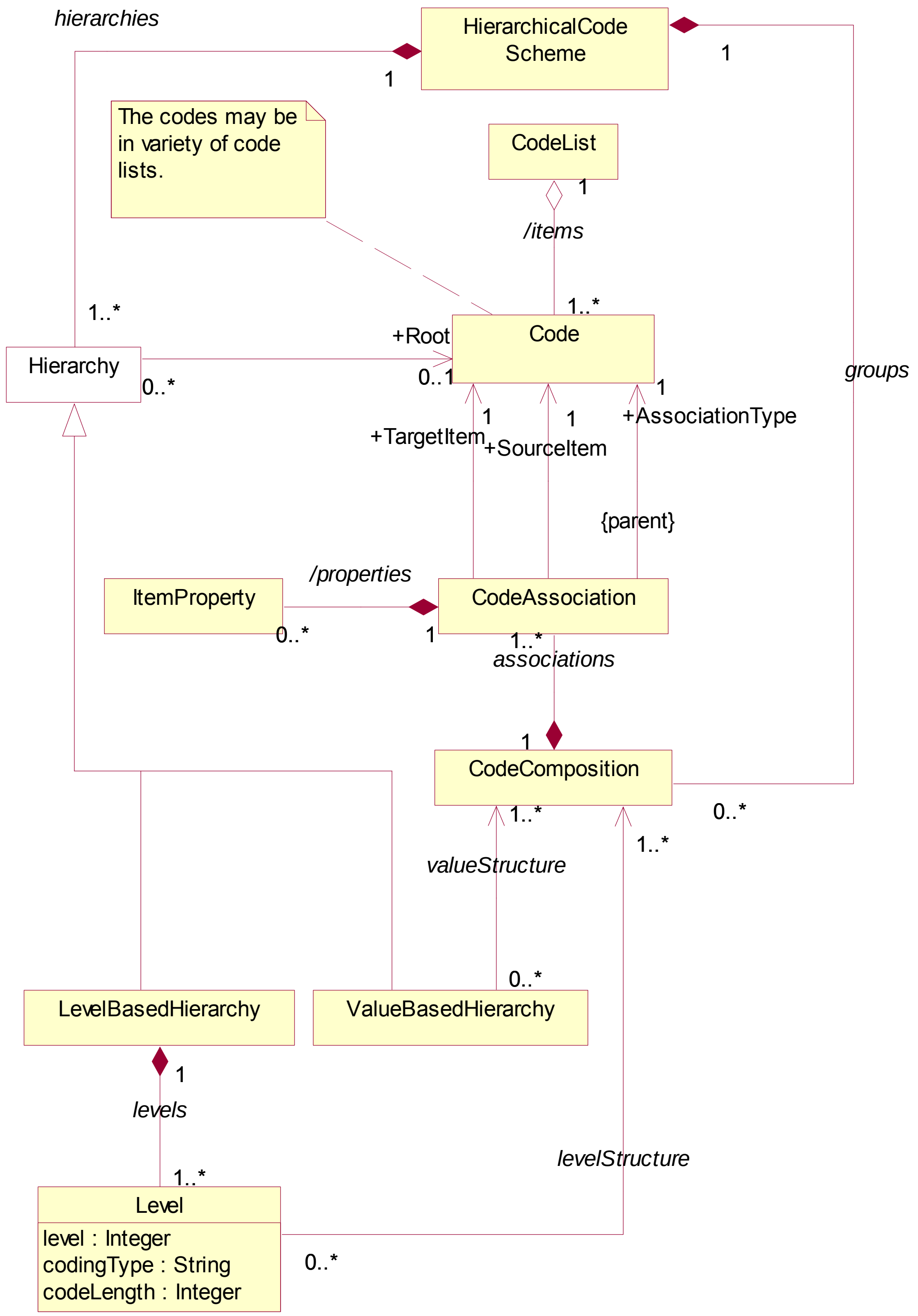


Figure 54: Class diagram of the Hierarchic Code Scheme

The Codes are in one or more Code Lists. The Code List can be hierarchic but the place of any Code in this hierarchy is not relevant to the Hierarchic Code Scheme, as this scheme defines its own parent/child associations and its own Hierarchies.

A Code Association relates a Code with its parent Code. If the Code and the parent Code are the same then the Code has no parent in that Code Association – this is usual for a Code that is the top level Code in a hierarchy.

Groups of Codes with the same parent are specified in the Code Composition.

Hierarchies are specified by ordering the Code Compositions. These hierarchies may have formal and explicitly defined levels - the level based hierarchy - or a hierarchy where there is no requirement to specify any metadata for the level – the value based hierarchy. A statistical classification is often specified as a balanced (level based) hierarchy where one Code can have only one parent. Such a hierarchy or “view” can be specified within a more fluid set of code associations. Properties can be specified for the Code Associations to give additional semantic, such as a sequence number in a group of Code Associations.

The Hierarchy specifies a root node which is the Code that defines the top of the hierarchy.

6.3.8.3 Example

The way this works is best shown by an example. Consider a simple set of codes depicting countries, and the grouping these countries into continents, economic areas, and political areas. The following “tables” of country groupings may be specified.

Table: Code

Code Id	Code description
AR	Argentina
AT	Austria
EE	Estonia
ES	Spain
FR	France
IT	Italy
US	United States
SA	South Africa
ZA	Zaire
	Etc.
EU	Europe
AU	Africa
UU	Americas
4F	OPEC countries
XM	Euro area countries
X0	EU countries
Z2	OECD countries
Z3	OECD European countries
XW	World

Table: Code Composition – comprises individual Code Associations

Composition	Association	
	Parent Code	**Child Code**
World	XW	XW
Europe_all_countries	EU	AT
	EU	EE
	EU	ES
	EU	FR
	EU	IT
America_all_countries	UU	US
	UU	AR
Africa_all_countries	AU	SA
	AU	ZA
World_all_countries	XW	AT
	XW	EE
	XW	ES
	XW	FR
	XW	IT
	XW	US
	XW	AR
	XW	SA
	XW	ZA
World_all_continents	XW	EU
	XW	UU
	XW	AU
EU countries	4F	AT
	4F	EE
	4F	ES
	4F	FR
	4F	IT
	4F	Etc.
Euro area countries	X0	AT
	X0	ES
	X0	FR
	X0	IT
	X0	Etc.

Table: Hierarchy

Hierarchy_Id	Description	Top level code
Political_World	Political world	XW

Table: Level

Hierarchy Id	Level Id	Level Description
Political_World	Lev_1	Whole world
Political_World	Lev_2	Continents
Political_World	Lev_3	Countries

Table: Level Composition

Hierarchy Id	Level Id	Code Composition
Political_World	Lev_1	World
Political_World	Lev_2	World_all_continents
Political_World	Lev_3	Africa_all_countries
		America_all_countries
		Europe_all_countries

The Hierarchical Code Scheme comprises a number of Code Associations, each one defining a parent child relationship. The codes in these associations may be drawn from a single or from multiple Code Lists. The "list" of Codes that participate in these associations can be seen as a simple flat list (the "Code" table). They only make sense in the context of the Code Association.

The Code Associations are grouped into Code Compositions, with each Code Composition grouping Codes which are meaningful in the context of one or more hierarchies that can be built. The constraint on the Code Composition is that all of the Codes must have the same parent.

Once these structures have been constructed a variety of hierarchies can be defined. Each Hierarchy can be seen as "view" on the entire code scheme. In the example the Hierarchy "Political World" is defined as a level based Hierarchy having three formal levels (Lev_1, Lev_2, Lev_3), with the top level node being the Code XW (World). These levels are composed of the appropriate Code Compositions i.e. those relating to the hierarchies that could comprise a political "view" of the world – continent, countries - (as opposed to an "economic" view like EU or "currency" view like Euro area).

6.4 Reporting and Dissemination Layer

6.4.1 Data Set

6.4.1.1 Schematic

The reporting of data is extremely simple and is depicted in the diagram below.

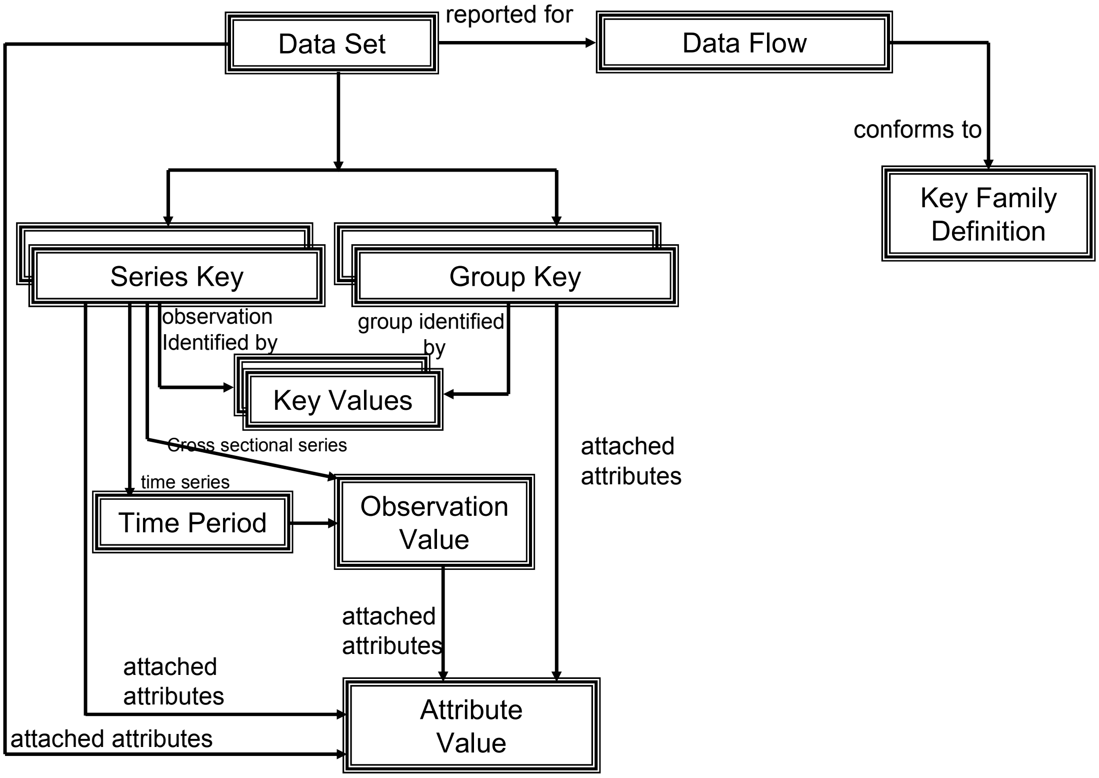


Figure 55: Schematic of the Data Set

The Data Set includes a reference to the Data Flow which in turn is linked to the Key Family Definition – this allows an application to retrieve the definition so that the Data Set can be processed, and validated if required.

The main structure of the Data Set is a set of Keys and Group Keys. Each Key comprises Key Values, a value for each of the Dimensions defined in the Key Family. For each key there may be one or more Observation Values: for a time series each Observation Value is related to a Time Period, whereas for a cross sectional series it is not. Attribute Values can be reported and each of these values can be attached to one of Data Set, Series Key, and Group Key.

6.4.1.2 The Model

The class diagram of the Data Set is shown below.

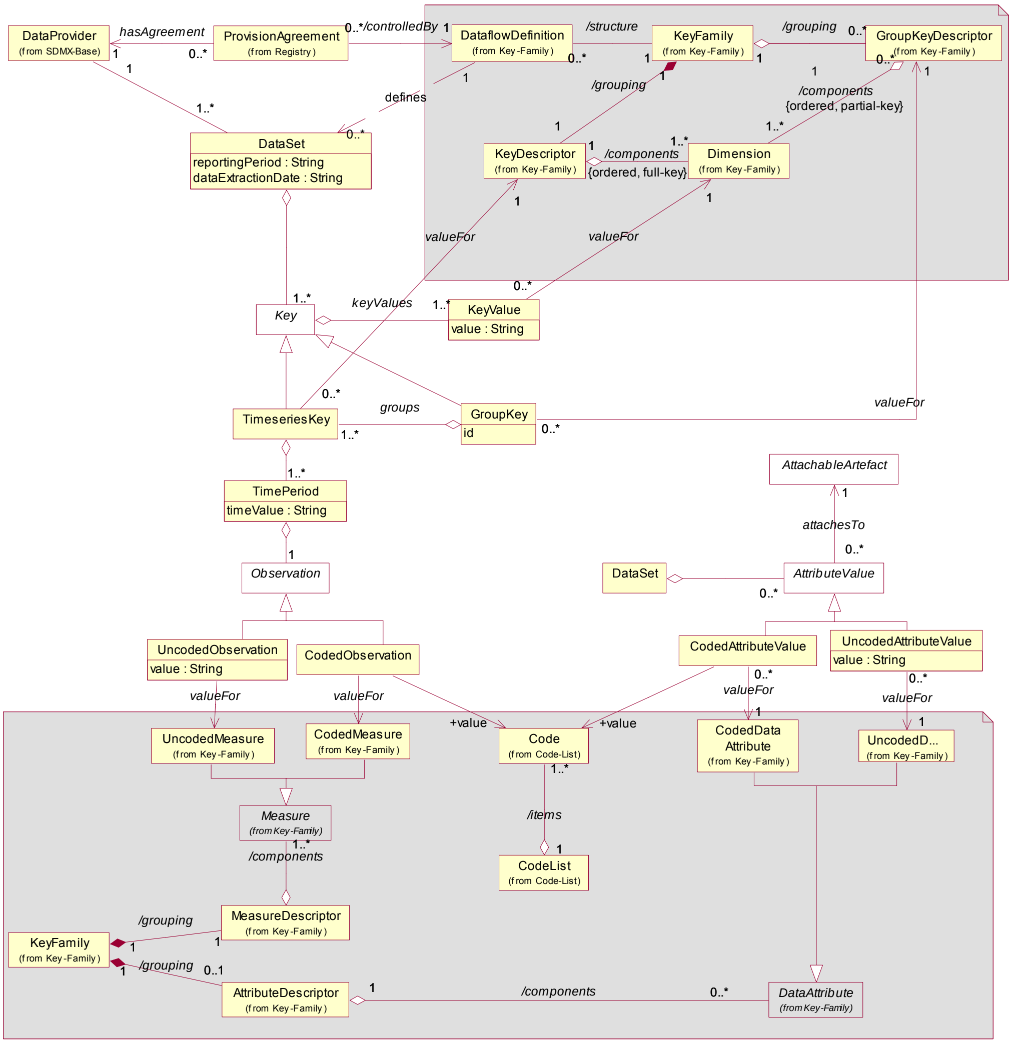


Figure 56: Full Relationship class diagram of the Data Set

This seems rather complex at first sight but this is because the diagram shows the association of the components of the Data Set to the components in the Key Family. The classes shown in the shaded areas are not a part of the data set – all that is required is the reference to the Dataflow Definition (as this enables an application to locate the Key Family definition). *Note that not all organisations will maintain Dataflow Definitions in their systems and create these links, and so the XML schema has an element that directly identifies the Key Family.*

If the shaded areas are eliminated then the diagram is much simpler. The diagram below shows the model for a time series data set.

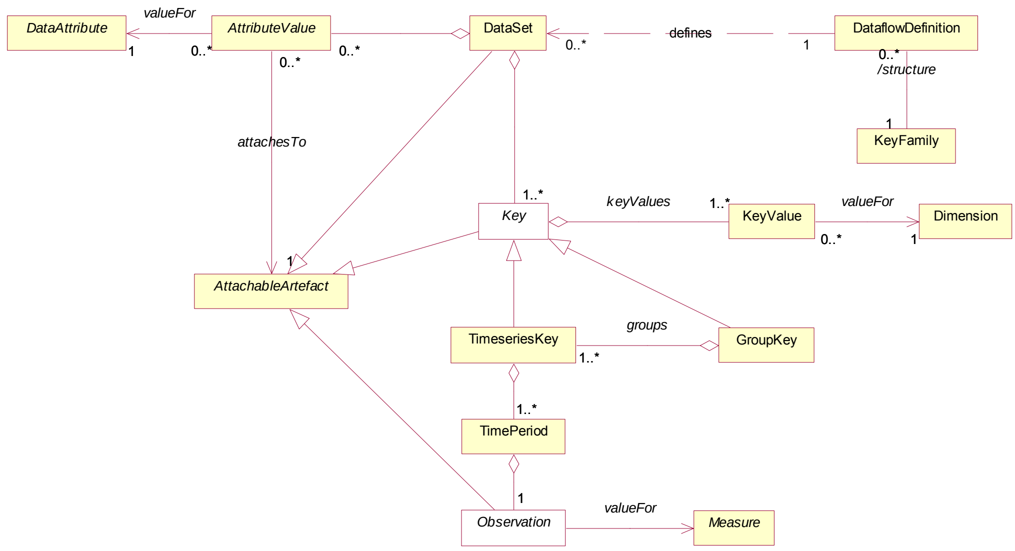


Figure 57: Simplified model of the Data Set

The map of this model to the schematic is shown below:

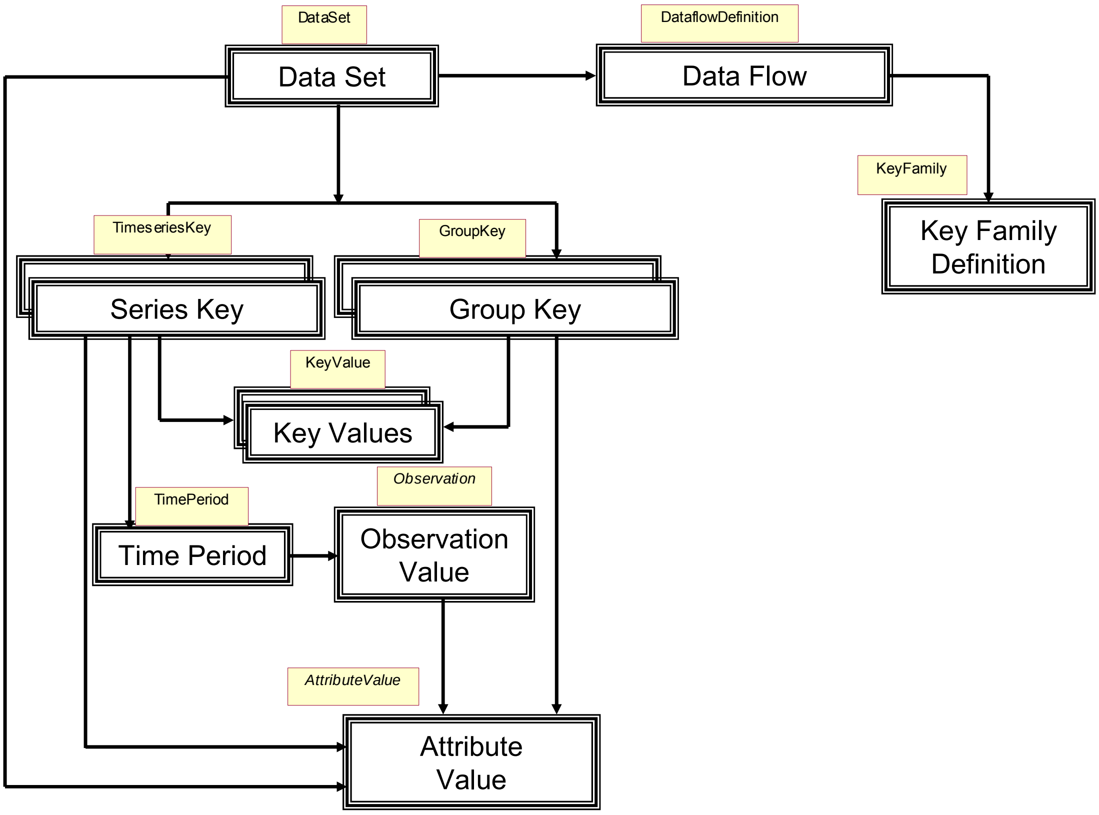


Figure 58: Map of the Data Set schematic to the SDMX Information Model

6.4.1.3 Example Data Set

This example shows how the model supports the data for the table at Figure 15. The Key Family supporting this data is shown at Figure 28 and is repeated below.

Dimensions - Key		
Type	**Concept**	**Representation/Code list**
Dimension (role is Frequency)	FREQ	Code List: CL_FREQ
Dimension	DEMOGRAPHIC_TYPE	Code List: CL_DEMOG_TYPE
Dimension	REGION	Code List: CL_REGION
Dimension	AGE_RANGE	Code List: CL_AGE_RANGE
Dimension (role is Time)	TIME	Date/Time
Measure		
OBS_VALUE (role is Primary Measure)		

Attributes			
Concept	**Assignment Status**	**Assignment Level**	**Representation/Code List**
OBS_STATUS	Mandatory	Observation	Code List: CL_OBS_STATUS
UNIT_OF_MEASURE	Mandatory	Series	Code List: CL_MEASURE_UNIT
TITLE	Mandatory	Data Set	Text
SOURCE	Conditional	Data Set	Text
PUBLICATION_DATE	Conditional	Data Set	Text

Figure 59: Table showing the components of a simple Key Family for demographic data

The table below shows the code lists used in this example (note that these are fictitious and do not represent any specific coding scheme).

Code List: CL_FREQ	
Code	**Name**
A	Annual
Q	Quarterly
M	Monthly
Code List: CL_AGE_RANGE	
1	0-15
2	16-64
3	Over 64
Code List: CL_DEMOG_TYPE	
A	Age
B	Mortality

C	Live Births
Code List: CL_REGION	
NE	North East
NW	North West
YH	Yorkshire and Humberside
WM	West Midlands
ES	East
LN	London
SE	South East
SW	South West
Code List: CL_OBS_STATUS	
A	Normal value
B	Break
E	Estimated value
F	Forecast value
Code List: CL_MEASURE_TYPE	
P	Percentage
C	Count

Figure 60: Table of codes

The diagram below highlights, by means of shading, the part of the example data set used for the example.

Age structure: regional comparison, 2002

	Percentages		
	0-15	16-64	65 and over
North East	19	64	17
North West	20	64	16
Yorkshire and the Humber	20	64	16
East Midlands	20	64	16
West Midlands	20	63	16
East	20	63	17
London	20	68	12
South East	20	64	16
South West	19	62	19
England	20	64	16
Wales	20	63	17
Scotland	19	65	16
Northern Ireland	23	63	13
United Kingdom	20	64	16

Sources:
Age structure up to 2002: population estimates, Office for National Statistics, General Register Office for Scotland, Northern Ireland Statistics and Research Agency
Age structure for 2031: population projections, Government Actuary's Department

Figure 61: Example data set

The way the model supports the reporting of this data is shown in the diagram below.

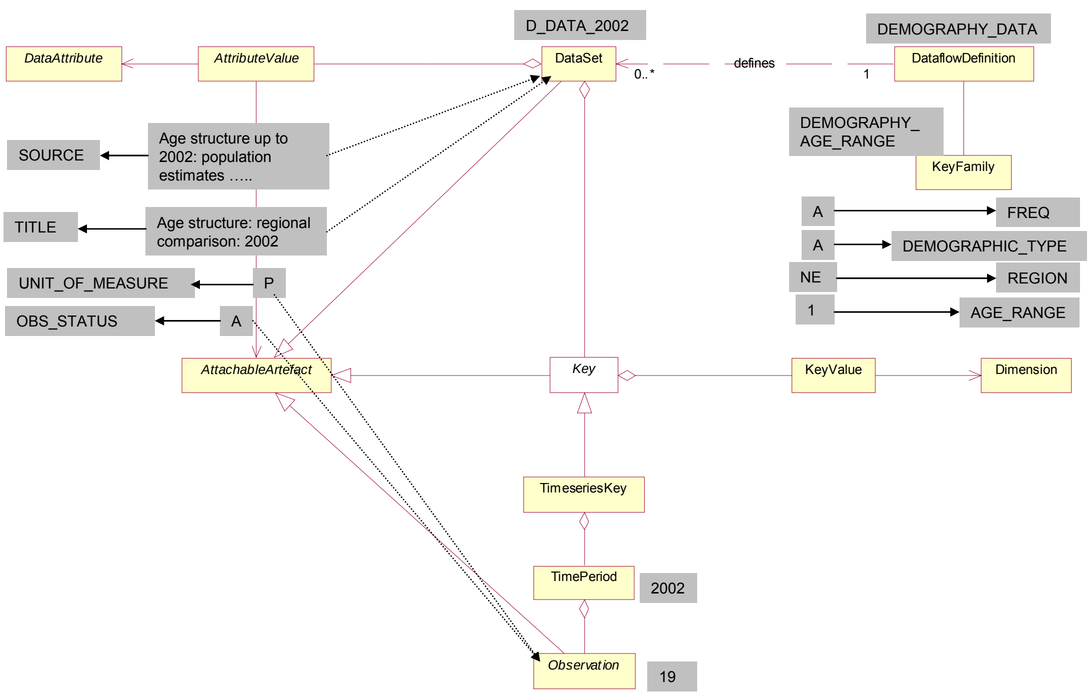


Figure 62: Example data mapped to the Data Set model

An extract from an SDMX-ML data set that contains this data is shown below.

```
<DataSet><generic:KeyFamilyRef>DEMOGRAPHY_AGE_RANGE</generic:KeyFamilyRef>
   <generic:Attributes>
<generic:Value concept="TITLE" value="Age structure: regional comparison: 2002"/>
<generic:Value   concept="SOURCE"   value="Age   structure   up   to   2002:   population
estimates.."/>
   </generic:Attributes>
   <generic:Series>
      <generic:SeriesKey>
         <generic:Value concept="FREQ" value="A"/>
         <generic:Value concept="DEMOGRAPHIC_TYPE" value="A"/>
         <generic:Value concept="REGION" value="NE"/>
         <generic:Value concept="AGE_RANGE" value="1"/>
</generic:SeriesKey>
      <generic:Obs>
         <generic:Time>2002</generic:Time>
         <generic:ObsValue  value="19"/>
         <generic:Attributes>
            <generic:Value concept="OBS_STATUS" value="A"/>
            <generic:Value concept="UNIT_OF_MEASURE" value="P"/>
</generic:Attributes>
      </generic:Obs>
      </generic:Series>
   </DataSet>
```

6.4.2 Metadata Set

6.4.2.1 Schematic

The reporting of metadata is extremely simple and is depicted in the diagram below.

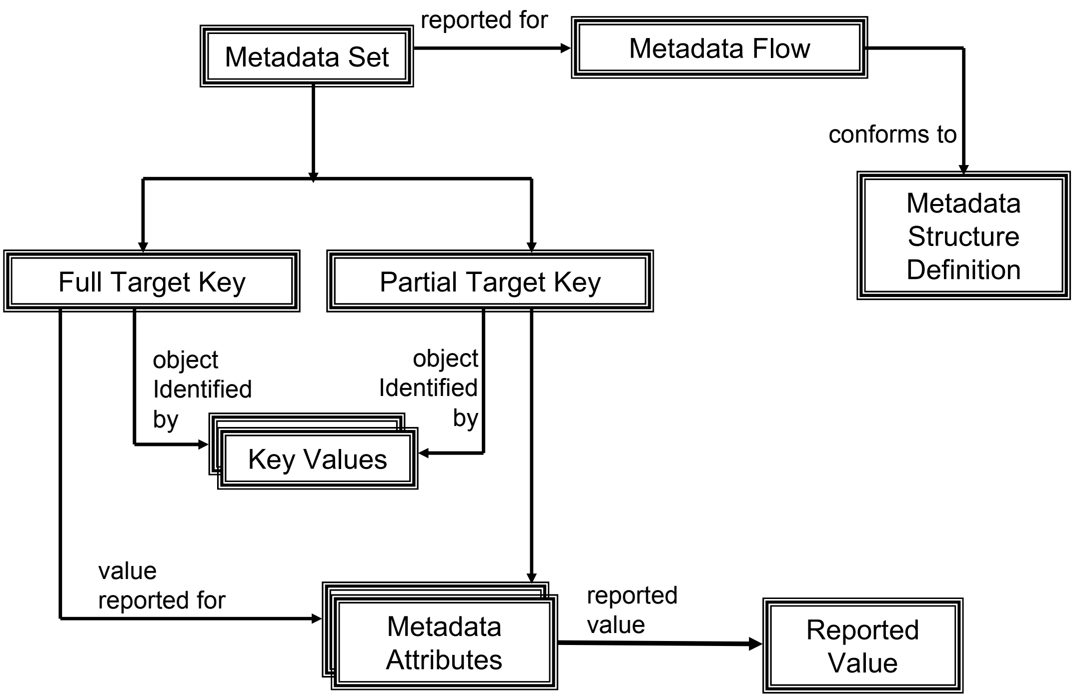


Figure 63: Schematic of the Metadata Set

The Metadata Set includes a reference to the Metadata Flow which in turn is linked to the Metadata Structure Definition – this allows an application to retrieve the definition so that the report can be processed, and validated if required.

The main structure of the report is a set of object identifiers or Keys. Each Key comprises Key Values, a value for each of the Object Types defined for the Identifier Components of the Full Target Identifier or Partial Target Identifier. For each key there may be one or more Reported Values, each value is related to the Metadata Attribute (i.e. the Concept) for which the value is reported.

6.4.2.2 The Model

The class diagram of the Metadata Set is shown below.

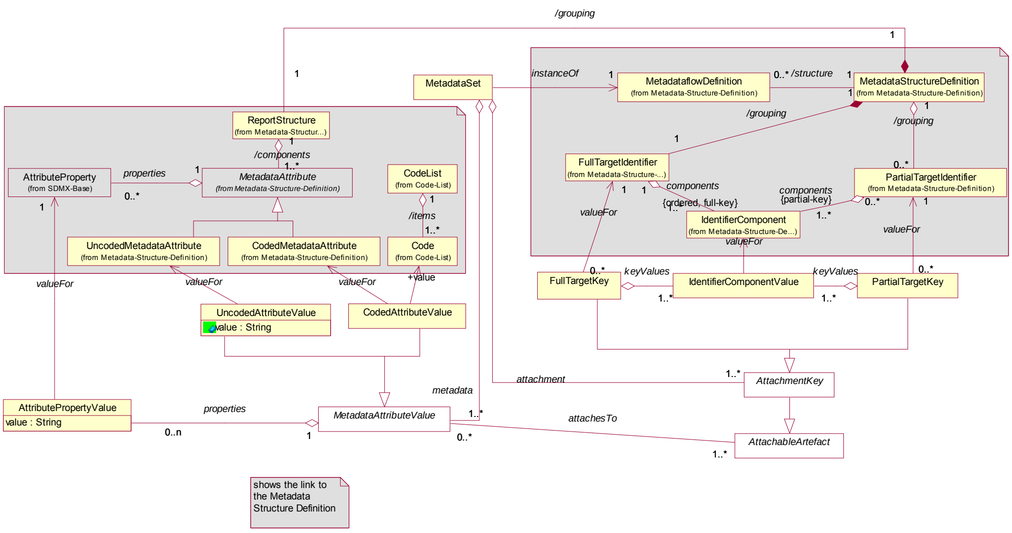


Figure 64: Full Relationship class diagram of the Metadata Set

This diagram also shows the link between the Metadata Set and the Metadata Structure Definition (the linked classes are shown in the grey shaded boxes). In a Metadata Set there need only be a reference to the Metadataflow Definition as this enables an application identify the relevant Metadata Structure Definition which will enable it to validate and process the metadata.

The Metadata Set comprises Metadata Attribute Values and a set of Attachment Keys. An Attachment Key is either a Full Target Key or a Partial Target Key (Note that both the Full Target Key and the Partial Target Key inherit from Identifiable Artefact and so have an Id – for both the Full Target Key and the Partial Target Key this Id will the same as the Id for the Full Target Identifier and the Partial Target Identifier).

The model does not specify, for an implementation, whether the Metadata Attribute Value references an Attachment Key or whether an Attachment Key references a Metadata Attribute Value.

The diagram below shows the Metadata Set with just the concrete classes and no link to the classes of the Metadata Structure Definition. This is the conceptual model of the XML schema. Note that the /association depicts an association inherited from the super (abstract) classes.

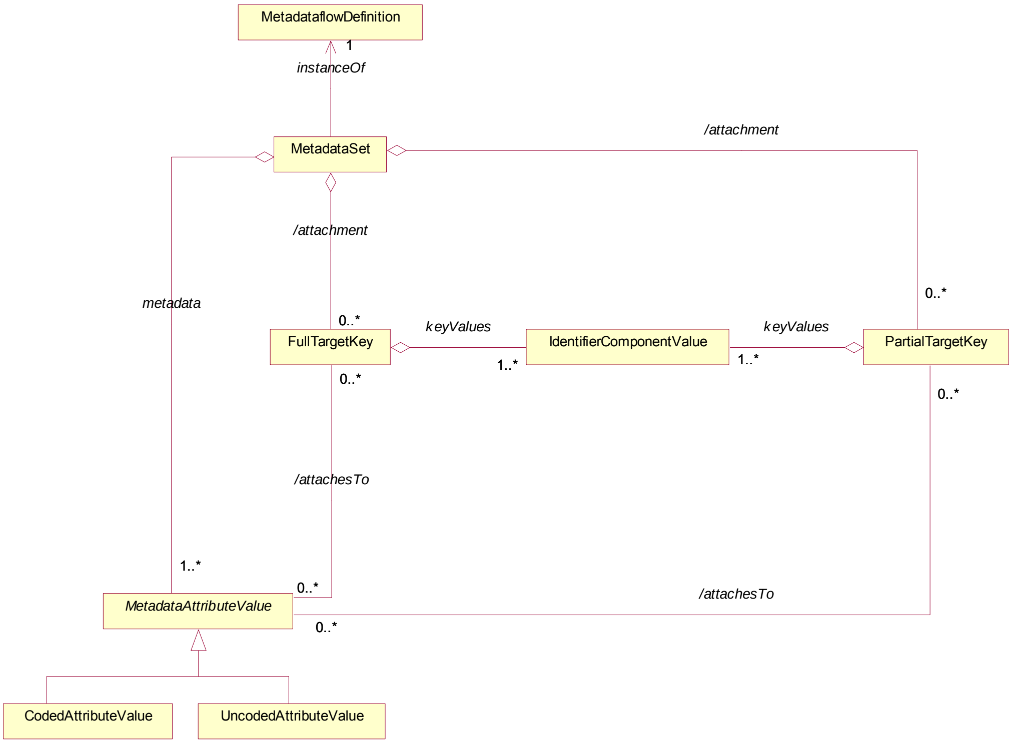


Figure 65: Simplified model of the Metadata Set

6.4.2.3 Example –SDDS Reference Metadata

An example of the content of the report for the Coverage concept (see Figure 41) is shown below.

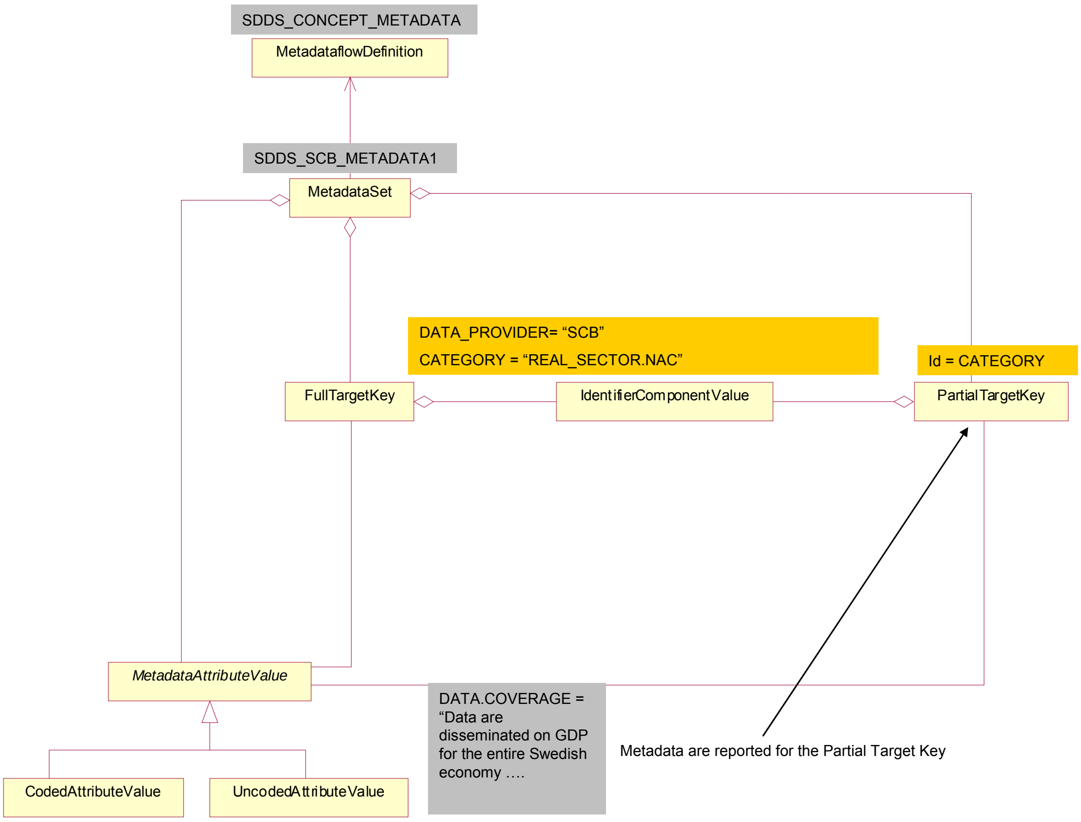


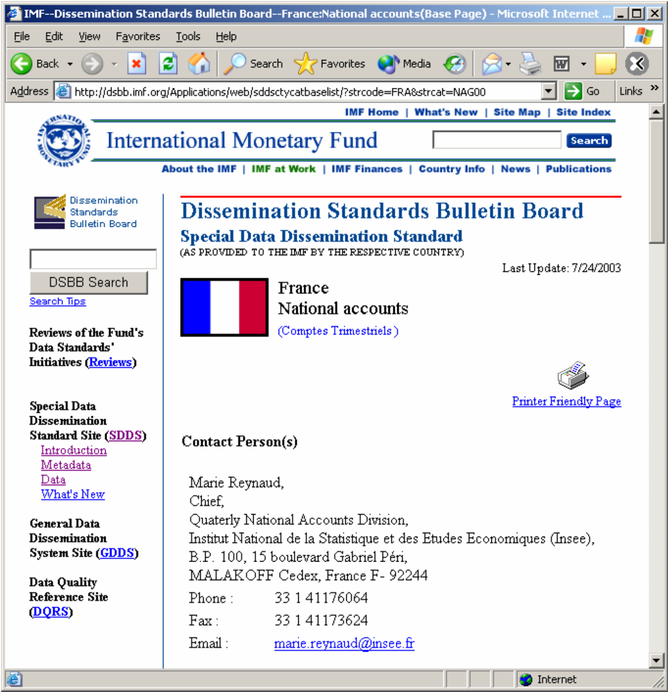
Figure 66: Schematic example of a metadata report

In the example above the Reported Value is for the Metadata Concept COVERAGE within the DATA concept. It is reported for the National Accounts (NAC) Category for the Data Provider Sweden (SE).

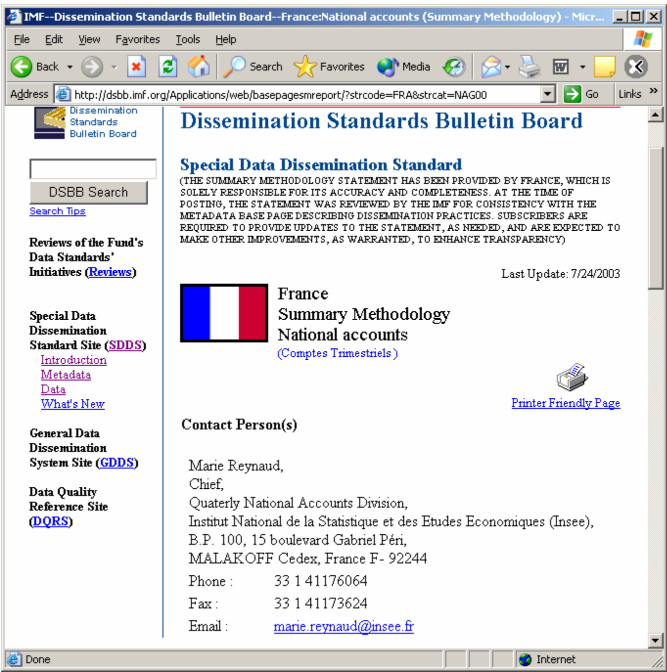
In a full report this value would be followed by the identification of other concepts for National Accounts from Sweden, each with a reported value, and then possibly another object identifier (e.g. Short Term Indicators for Sweden) and the values to be reported.

6.4.2.4 Example – Contact Metadata for SDDS

6.4.2.4.1 Example 1



Marie Reynaud is the Methodology contact for National Accounts and the (default) country representative for all concepts for National Accounts



The Metadata Set conforms to the Metadata Structure Definition described in Section 6.3.7.6 and would contain the following:

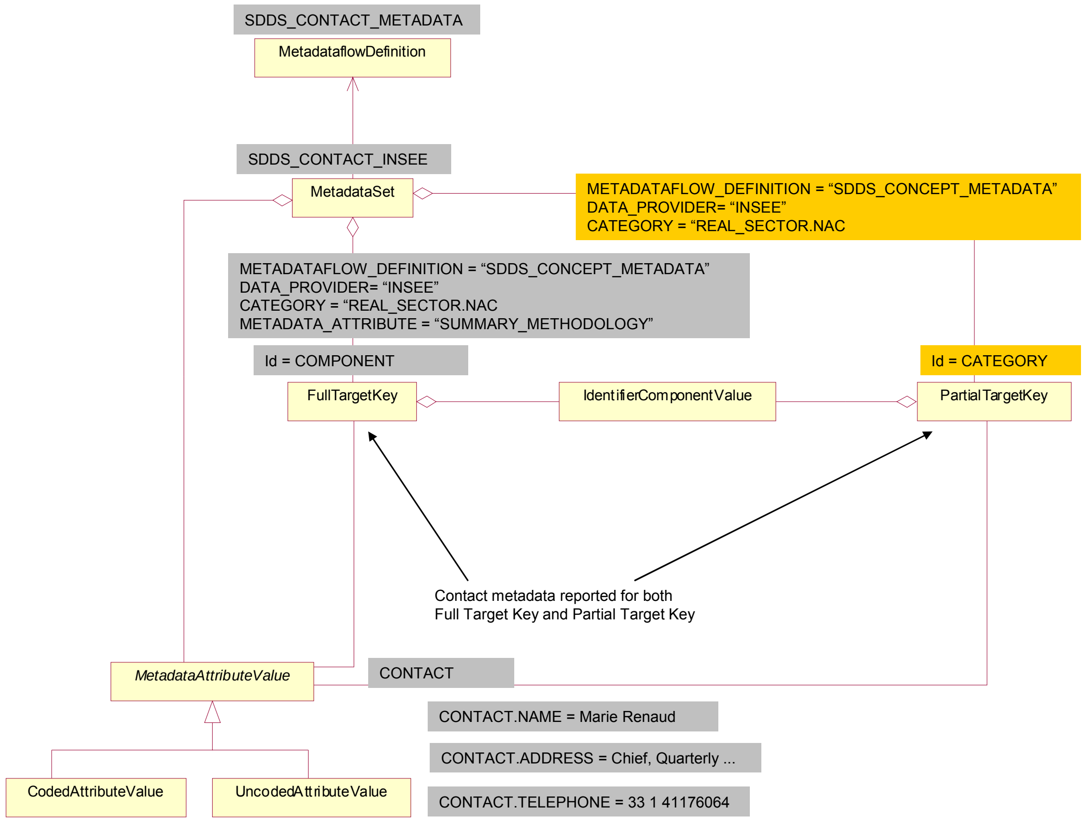


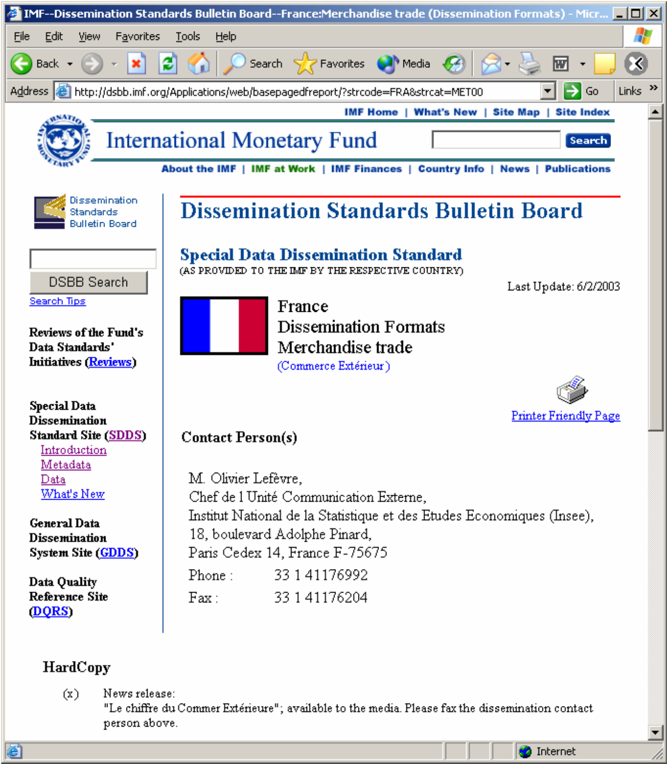
Figure 67: Example contact metadata

An extract from an SDMX-ML data set that contains this data is shown below.


```
<genericmetadata:MetadataSet>
   <genericmetadata:MetadataStructureRef>SDDS_CONTACT_INFO</genericmetadata:MetadataSt
ructureRef>
   <genericmetadata:MetadataStructureAgencyRef>IMF</genericmetadata:MetadataStructureA
gencyRef>
   <genericmetadata:ReportRef>PRIMARY_CONTACT_REPORT_COMPONENT</genericmetadata:Report
Ref>
         <genericmetadata:AttributeValueSet>
         <genericmetadata:TargetRef>COMPONENT</genericmetadata:TargetRef>
         <genericmetadata:TargetValues>
            <genericmetadata:ComponentValue
object="MetadataFlow">SDDS_METADATA</genericmetadata:ComponentValue>
            <genericmetadata:ComponentValue
object="Category">REAL_SECTOR.NAC</genericmetadata:ComponentValue>
            <genericmetadata:ComponentValue
object="DataProvider">INSEE</genericmetadata:ComponentValue>
            <genericmetadata:ComponentValue
object="MetadataAttribute">SUMMARY_METHODOLOGY</genericmetadata:ComponentValue>
         </genericmetadata:TargetValues>
         <genericmetadata:ReportedAttribute conceptID="CONTACT">
            <genericmetadata:ReportedAttribute conceptID="NAME">
               <genericmetadata:Value>Marie Renaud</genericmetadata:Value>
            </genericmetadata:ReportedAttribute>
            <genericmetadata:ReportedAttribute conceptID="ADDRESS">
               <genericmetadata:Value>Chief, Quarterly, ...</genericmetadata:Value>
            </genericmetadata:ReportedAttribute>
            <genericmetadata:ReportedAttribute conceptID="TELEPHONE">
               <genericmetadata:Value>33 1 41176064</genericmetadata:Value>
            </genericmetadata:ReportedAttribute>
         </genericmetadata:ReportedAttribute>
      </genericmetadata:AttributeValueSet>
<genericmetadata:ReportRef>PRIMARY_CONTACT_REPORT_CATEGORY</genericmetadata:ReportRef>
```

```
         <genericmetadata:AttributeValueSet>
         <genericmetadata:TargetRef>CATEGORY</genericmetadata:TargetRef>
         <genericmetadata:TargetValues>
            <genericmetadata:ComponentValue
object="MetadataFlow">SDDS_METADATA</genericmetadata:ComponentValue>
            <genericmetadata:ComponentValue
object="Category">REAL_SECTOR.NAC</genericmetadata:ComponentValue>
            <genericmetadata:ComponentValue
object="DataProvider">INSEE</genericmetadata:ComponentValue>
         </genericmetadata:TargetValues>
         <genericmetadata:ReportedAttribute conceptID="CONTACT">
            <genericmetadata:ReportedAttribute conceptID="NAME">
               <genericmetadata:Value>Marie Renaud</genericmetadata:Value>
            </genericmetadata:ReportedAttribute>
            <genericmetadata:ReportedAttribute conceptID="ADDRESS">
               <genericmetadata:Value>Chief, Quarterly, ...</genericmetadata:Value>
            </genericmetadata:ReportedAttribute>
            <genericmetadata:ReportedAttribute conceptID="TELEPHONE">
               <genericmetadata:Value>33 1 41176064</genericmetadata:Value>
            </genericmetadata:ReportedAttribute>
         </genericmetadata:ReportedAttribute>
      </genericmetadata:AttributeValueSet>
   </genericmetadata:MetadataSet>
```

6.4.2.4.2 Example 2



M. Olivier Lefèvre is contact for Access for both National Accounts and Merchandise Trade

The Metadata Set would contain the following:

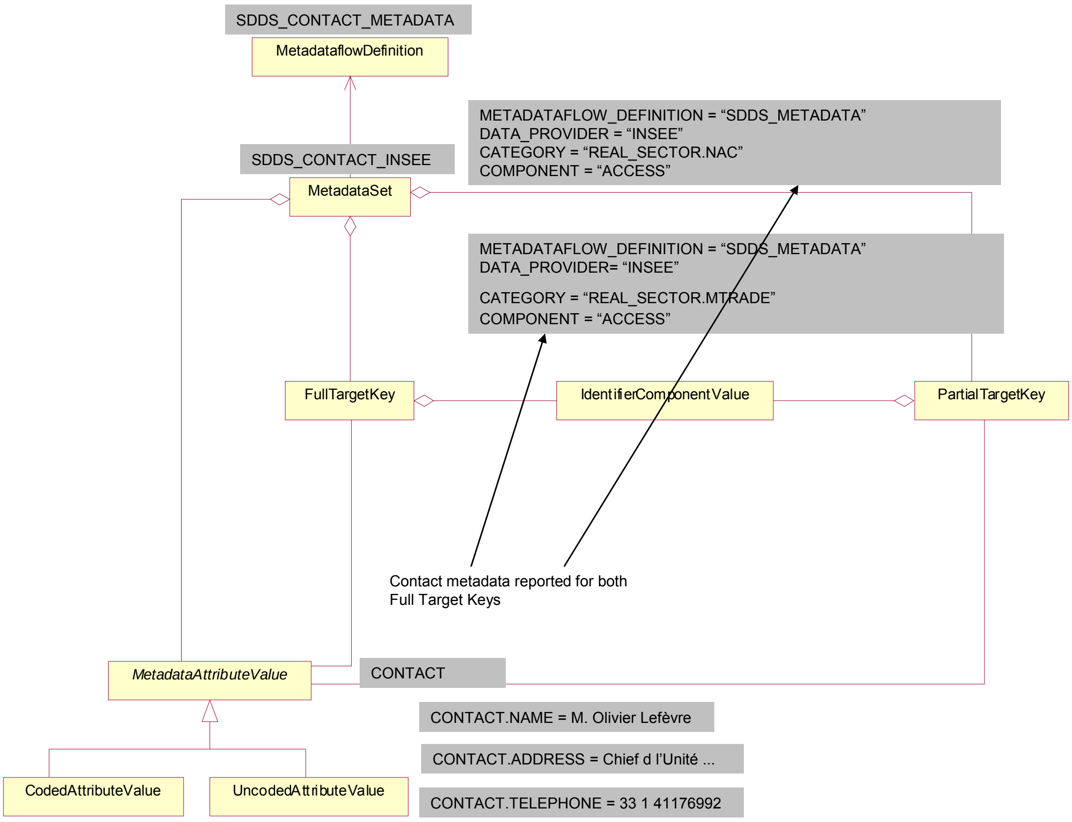


Figure 68: Example contact metadata

6.4.2.5 Example – Data Structure Definition Metadata

The following data set was used earlier as a basis for a Data Structure Definition.

Age structure: regional comparison, 2002

			Percentages
	0-15	16-64	65 and over
North East	19	64	17
North West	20	64	16
Yorkshire and the Humber	20	64	16
East Midlands	20	64	16
West Midlands	20	63	16
East	20	63	17
London	20	68	12
South East	20	64	16
South West	19	62	19
England	20	64	16
Wales	20	63	17
Scotland	19	65	16
Northern Ireland	23	63	13
United Kingdom	20	64	16

Figure 69: Example demographic data set

The structure was described by the following Data Structure Definition.

Dimensions - Key			
Type	**Concept**	**Representation/Code list**	
Dimension (role is Frequency)	FREQ	Code List: CL_FREQ	
Dimension	DEMOGRAPHIC_TYPE	Code List: CL_DEMOG_TYPE	
Dimension	REGION	Code List: CL_REGION	
Dimension	AGE_RANGE	Code List: CL_AGE_RANGE	
Dimension (role is Time)	TIME	Date/Time	
Measure			
OBS_VALUE (role is Primary Measure)			
Attributes			
Concept	**Assignment Status**	**Assignment Level**	**Representation/Code List**
OBS_STATUS	Mandatory	Observation	Code List: CL_OBS_STATUS
UNIT_OF_MEASURE	Mandatory	Series	Code List: CL_MEASURE_UNIT
TITLE	Mandatory	Data Set	Text
SOURCE	Conditional	Data Set	Text
PUBLICATION_DATE	Conditional	Data Set	Text

Figure 70: Demography Data Structure Definition (Key Family)

The following Metadata Set conforms to the Metadata Structure Definition described in Section 6.3.7.7 attaches metadata for the COVERAGE Metadata Attribute to the AGE_RANGE Dimension Concept.

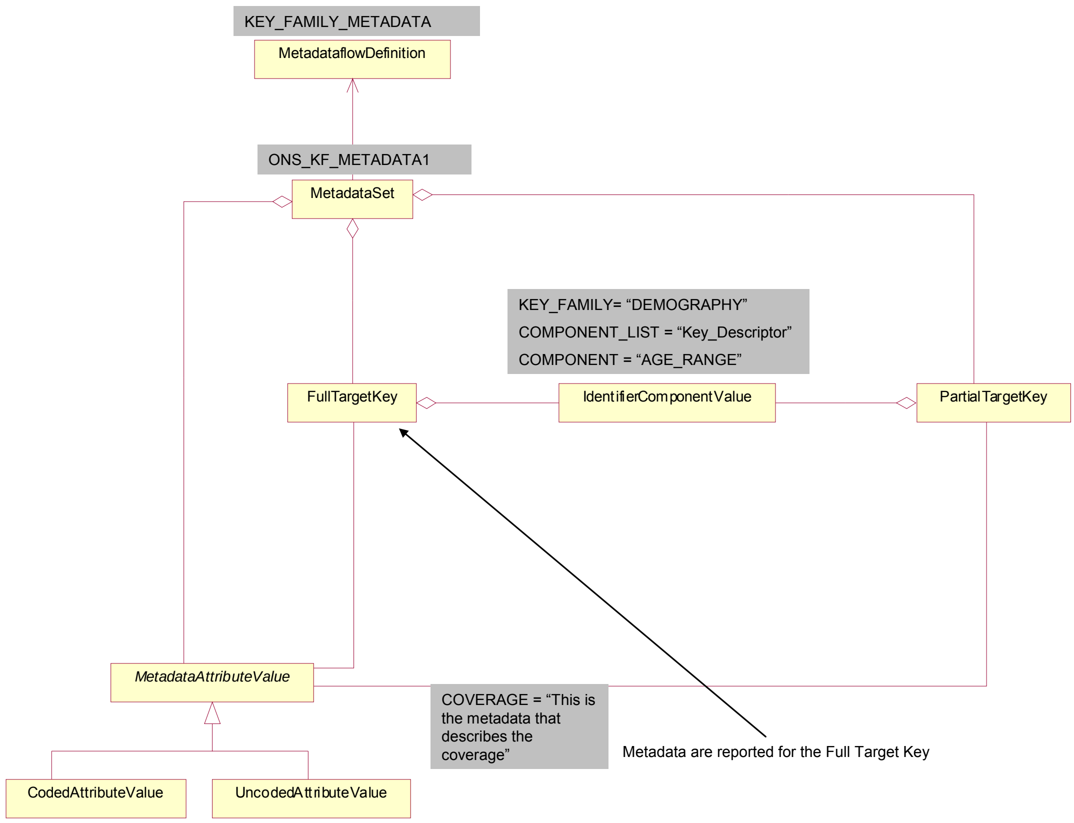


Figure 71: Example Metadata Set for a Key Family Dimension Component