SDMX self-learning package No. 3

Student book

SDMX-ML Messages

Produced by	Eurostat, Directorate B: Statistical Methodologies and Tools Unit B-5: Statistical Information Technologies
Last update of content	February 2010
Version	1.0

TABLE OF CONTENTS

1 Scope of the student book

The student book aims to provide an understanding and complete view of SDMX-ML Messages. This includes:

- SDMX-ML Background
- SDMX-ML Formats
- SDMX-ML Structure Messages & Data Messages
- SDMX-ML Schemas;

At the end of this student book, the user should be able to understand the different formats and messages and their application.

This student book is the third one of a set of student books (see Table 1 - Student books on SDMX) which altogether provides the complete set of information to master SDMX, with a particular focus on the data model.

Ref.	Title
[01]	Introduction to SDMX
[02]	The SDMX Information Model
[03]	SDMX-ML Messages
[04]	Data Structure Definition
[05]	Metadata Structure Definitions
[06]	XML based technologies used in SDMX
[07]	SDMX architecture using the pull method for data sharing – Part 1
[08]	SDMX architecture using the pull method for data sharing – Part 2

Table 1 - Student books on SDMX

Prerequisites

The reading of the first and the second Student Book is required to gain a global view of SDMX Information Model and the basics of the Data & Metadata structures.

2 History and SDMX Version 2.0

The SDMX Technical Standards Version 1.0 established an information model, which described aggregated statistical data sets and the structural metadata needed to exchange them in a standard fashion. A set of SDMX-ML messages was introduced.

- XML formats for exchange of structural metadata, data sets, and queries for these (SDMX-ML);
- EDIFACT formats for the structural metadata and data sets (SDMX-EDI). Unlike the SDMX-ML message formats, this student book does not cover the EDIFACT based ones (SDMX-EDI).

Based on the enhanced SDMX Information Model (SDMX-IM) of SDMX Version 2.0, the SDMX-ML formats have been extended by additional messages, like registry messages, SDMX-ML for reference metadata and additional data formats (e.g. Cross-sectional data message).

3 The SDMX-ML / XML Design

3.1 The formats of the SDMX-ML

Enhancements in SDMX standard v2.0 include the support of reference metadata, enhancements in structural metadata allow SDMX to support additional types of statistical information (Cross-sectional data) and the interaction with SDMX Registry / Repository services.

Therefore, the current SDMX standard offers different SDMX-ML formats for the exchange of data, structural and reference metadata, and query message to request SDMX-ML data messages from a web service or registry messages for the interaction with an SDMX Registry/Repository service[1].

To ensure an effective implementation and to reduce the complexity, a shared set of common XML message constructs was developed, to be reused throughout all the SDMX-ML formats. This includes a common header structure and basic common message structure parts for all SDMX-ML messages. These shared common elements are implemented with the following XML schemes[2]:

- SDMXMessage.xsd - representing common message construction rules including the header information;
- SDMXCommon.xsd – containing common low-level components to implement the basic structure.

[1] SDMX-ML registry messages are not covered in this student book.

[2] An overview of the other SDMX Schemas is presented below in section 3.3.

An XML schema[3] is a description of a type of a XML document. Inside the schema are expressed constraints and document content types above and beyond the basic syntactical constraints imposed by XML itself.

Finally, a division was established between 'generic' XML formats, which are not specific to a particular structure definition (DSD or MSD), and a set of formats which are specific to DSDs or MSDs or respectively the particular type of use.

3.2 Fostering the Use of a Standard SDMX-ML

In addition to the different formats, standard transformation (e.g. from a Data Structure definition to corresponding schemes to validate the SDMX-ML data messages) and corresponding transformation tools have been developed to support the implementation of SDMX.

These tools[4] include software to build DSD and MSD structures. Moreover, conversion support is provided to transform SDMX-ML data files from one format to another, and between SDMX-ML data files, SDMX-EDI and text files. The increasing number of free tools promotes the adoption of SDMX and permits the data to be easily used across all processes.

3.3 SDMX-ML Packaging

Several XML namespaces[5] are defined for SDMX. An SDMX-ML message may contain element or attribute names from more than one XML namespaces defined for SDMX. Thus, an ambiguity between identically named elements is resolved, when they are associated to different namespaces.

XML schemas have been implemented in association with the SDMX-ML namespaces. The SDMXMessage.xsd implemented in relation with the 'Message' namespace is the schema used for validating all SDMX-ML messages. Other namespaces and the corresponding XSD schemas, like the SDMXCommon.xsd and the message specific schemes, are imported by this schema. With this construct the specificities of each SDMX-ML message type are supported, for example: For a SDMX-ML structure message (e.g. containing a DSD), besides the 'message' and 'common' namespaces, the 'structure' namespace must be included, which implies also the usage of the SDMXStructure.xsd schema. This schema describes all structural metadata messages within SDMX.

[3] More details on XML schemas are provided in the student book 6 - XML based technologies used in SDMX.

[4] Two of the main tools featuring these characteristics are the SDMX Data Structure Wizard and SDMX Converter. Both tools can be downloaded from the web page www.sdmx.org.

[5] XML namespaces are used for providing uniquely named elements and attributes in an XML document. More details on XML namespaces are provided in the student book 6 - XML based technologies used in SDMX.

SDMX standard is built SDMX to own and share certain namespaces and schemes, and on the other hand it permits maintenance agencies to own and maintain namespaces and schemes specific to their DSDs or MSDs.

The table below presents a set of SDMX standard schemes. They make use of the XML Schema importing mechanism[6] to link other ones in order to embed their constructs.

SDMX schemes	
Schema file	**Use of the schema**
SDMXMessage.xsd	Contains the common message constructs, including the common header information for all SDMX-ML messages
SDMXStructure.xsd	Contains the descriptions of structural metadata such as DSDs, concepts, and code lists
SDMXCommon.xsd	Contains constructs shared in common across all of the SDMX-ML message types.
SDMXGenericData.xsd	Describes a generic format for formatting SDMX-ML Generic data files
SDMXQuery.xsd	Describes the structure of the generic query message for web services developers and users
SDMXCompactData.xsd	Provides the common format to be used for all DSD-specific 'Compact' schemas (not maintained by SDMX.org) for Data validation, processing, publication and production use of SDMX-ML Compact data files
SDMXUtilityData.xsd	Provides the common format to be used for all DSD-specific 'Utility' schemas (not maintained by SDMX.org) for Data validation, processing, publication and production use of SDMX-ML Utility data files
SDMXCrossSectionalData.xsd	Provides the common format to be used for all DSD-specific 'Cross-sectional' schemas (not maintained by SDMX.org) for Data validation, processing, publication and production use of SDMX-ML Cross-sectional data files
SDMXRefMetadata.xsd	Provides a generic format for reporting of reference metadata, regardless of Metadata Structure Definition
SDMXMetadataReport.xsd	Provides the common format to be used for all MSD-specific 'Metadata Report' schemas (not maintained by SDMX.org) for reference metadata reporting
SDMXRegistry.xsd	Provides standard interfaces for interactions with a set of registry services

Table 2 - Overview of SDMX XSD schemes

[6] The import element is used to add multiple schemas with different target namespace to a document. It enables schema components from different target namespaces to be used together, and hence enables the schema validation of instance content defined across multiple namespaces.

4 The SDMX-ML Messages for data and structure definitions and queries

4.1 General Rules and considerations

Any SDMX-ML message is constructed according to an XML schema file (with the extension .xsd).

In some cases, the schema file is already fixed in the SDMX standards (for example the ‘GENERIC’ data message schema).

In other cases, the schema is derived from the specific data structure definition message, where it depends on. This is the case for the Compact data message, the Utility Data message and the Cross-sectional Data Message.

With the schemes, validation can be performed for the data messages with generic XML tools at different levels: XML syntax and data types, as well as the enforcement of the DSD structures and codes for the given data. Dependencies among different coded Dimension and Attribute values are not captured in the DSD and consequently not included in the corresponding schemes. They have to be validated by a dedicated application for the data production.

A common part for each SDMX-ML message is the <Header>, which has the same basic structure for all message types. The example in Figure 1 - SDMX-ML Message Header displays typical <Header> information of an SDMX-ML data message on the transmitted dataset:

```
<Header>
    <ID>STSIND_PROD_M</ID>
    <Test>true</Test>
    <Prepared>2007-08-21T12:56:00</Prepared>
    <Sender id="SE1"/>
    <Receiver id="4D0"/>
    <DataSetID>STSIND_PROD_M-2007-08-21T12:53:00</DataSetID>
    <DataSetAction>Replace</DataSetAction>
    <Extracted>2007-08-21T12:53:00</Extracted>
    <ReportingBegin>2007-04-01T00:00:00</ReportingBegin>
    <ReportingEnd>2007-06-30T18:00:52</ReportingEnd>
</Header>
```

Figure 1 - SDMX-ML Message Header

As a suggestion for a good practice, the namespace prefix added to the XML elements of the messages is indicating the statistical area of the data and the reported data format.

Thus, in the following example, the namespace prefix **‘sts_c:’** would stand for ‘Short term business statistics (‘sts’)’ - data in compact format (‘_c’):

```
xmlns:sts_c="urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:compact"
```

In the SDMX-ML data message examples, the so-called ‘SiblingGroup’ was chosen as grouping. This used to be the most common grouping, containing a set of series, which are identical except that they are measured with different frequencies. Thus, the corresponding group key contains all series key dimensions except the Frequency (FREQ).

4.2 *Deriving one SDMX-ML message from another*

SDMX offers four different representations for reporting datasets according to the final use. There are equivalent in a way, but has different objectives. They will be described in detail in next chapters.

SDMX schemes	
Formats	**Objetives**
Generic Message	Format to convey the data using a representation that enumerates everything concerning the structure.
Compact Message	Format for large-volume exchange of data.
Utility Message	Format to convey the data using a representation that enumerates everything concerning the data validation.
Cross-Sectional Message	Format for non-time-series data. Exchange of many observation types.

Table 3 - Message formats

Much of the flexibility and elegance of SDMX standards arises from two facts:

1. First, since all the messages are built on the underlying consistent SDMX Information Model, the four SDMX-ML data messages can be converted to others (as being shown in Figure 2 - Model-based equivalence of SDMX-ML messages). This requires some preconditions, as for example the availability of the Time dimension, frequency and time format in the DSD to perform a conversion into generic, compact or utility data message.

   The Data Structure Definition (DSD), as being stored in the SDMX-ML Structure definition message, represents the basis for the derived SDMX-ML data message format schemes (Compact, Cross-sectional and Utility). These schemes (XSD) rule the structure and content of the underlying SDMX-ML data messages in the same format.

2. Second, because the four SDMX-ML data messages use standard XML, the conversions, and any other operations (validations, file editing, etc.) can also be carried out using standard XML tools/software, such as XML parsers/editors and XSLT processors. Such tools/software are available – often for free – and well supported with documentation and tutorials.

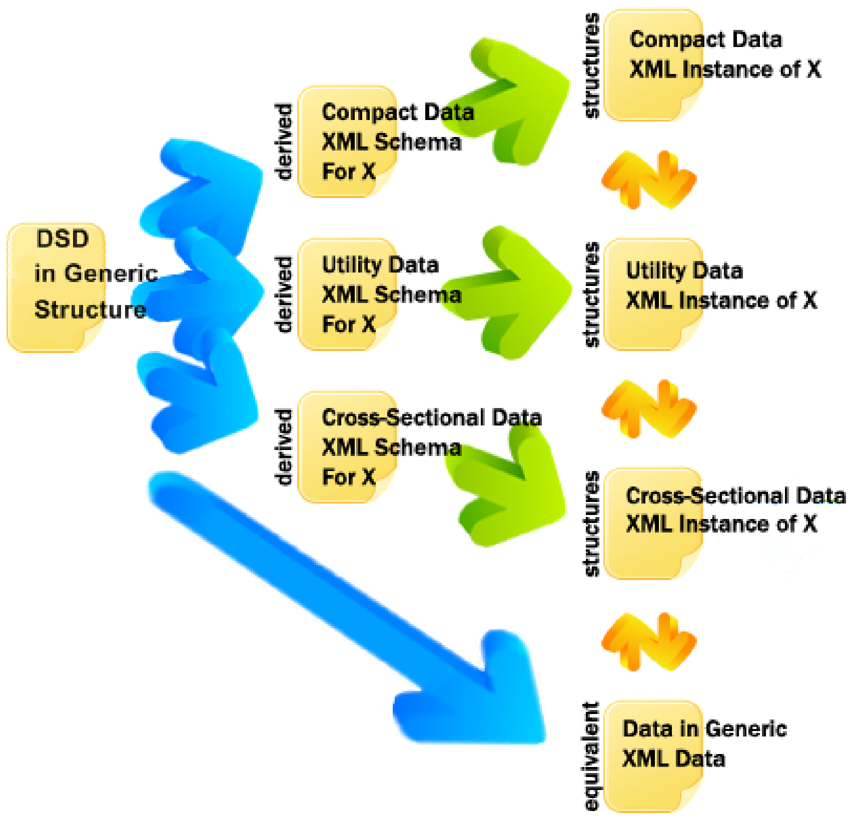


Figure 2 - Model-based equivalence of SDMX-ML messages

Conversion between the four SDMX-ML data formats can also be realized with software developed for this purpose. This approach can represent a benefit due to its expandability to other formats. For example, a user would be able to create SDMX-ML files from other data set formats like a GESMES/TS or a CSV-file.

Some restrictions apply per each non-SDMX-ML format, since the information conveyed by these formats is only a subset of the SDMX-ML formats' information exchange. For example, data send in one CSV file does usually not provide any <Header> information and the attachment level of the reported values of Dimensions and Attributes is not indicated.

SDMX tools & software exist, which support transformations and the translation between the different SDMX message formats[7]. They are freely available from the SDMX.org web site. The SDMX Converter, for example, is a Java tool that allows converting from/to various data set message formats based on the structure provided by SDMX-ML DSD structure files. The data file types the Converter handles for conversion are:

- SDMX-ML formats (in order to go from/to Cross-Sectional type the DSD has to contain Cross-Sectional information and Time Dimension)
- GESMES/TS, also known as SDMX-EDI; but no conversion from/to SDMX-ML Cross-sectional format;
- GESMES/2.1 and GESMES/DSIS;
- CSV, FLR.

[7] More details provided in the student book 6 - XML based technologies used in SDMX.

4.3 Generic SDMX-ML Data set

4.3.1 Generic Data message characteristics

The characteristics of a Generic data message can be summarized as follows:

- Conveys data in a form independent of a data structure definition message, as the data structure is imbedded in the message. Thus, the generic data message is not intended as a concise and efficient format for large volumes of data;
- Used when applications receiving the data do not have detailed understanding of the data set structure before they obtain the data set itself;
- It is applicable for the transmission of partial data sets (incremental updates) and whole data sets;
- Does not provide for strict validation between the data set and its structural definition using a generic XML parser (the parser cannot validate the codes since they are not contained in the schema). In terms of XML syntax, all codes and observation values are elements.
- All key values of the key dimensions are specified at the series level and the SDMX attribute values are attached at the level defined in the underlying DSD.

The Generic Data Set is the only format that is syntactically common for any underlying DSD. Thus, the XML elements and attributes used are common for all Data sets and independent from the related DSD for the particular Data set. This implies that the XML Schema for the syntactical validation of Generic data messages is common for all data sets. This schema is provided by the SDMX standard (http://www.sdmx.org/docs/2_0/SDMXGenericData.xsd).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GenericData
 xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message"
 xmlns:common="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common"
 xmlns:generic="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/generic"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd">
   <Header>…</Header>
   <DataSet>
      <generic:KeyFamilyRef>STS</generic:KeyFamilyRef>
      <generic:Group type="SiblingGroup">
         <generic:GroupKey>
            <generic:Value concept="REF_AREA" value="IT"/>
            <generic:Value concept="ADJUSTMENT" value="W"/>
            <generic:Value concept="STS_INDICATOR" value="PROD"/>
            <generic:Value concept="STS_ACTIVITY" value="NS0020"/>
            <generic:Value concept="STS_INSTITUTION" value="1"/>
            <generic:Value concept="STS_BASE_YEAR" value="2000"/>
         </generic:GroupKey>
         <generic:Attributes>
            <generic:Value concept="UNIT" value="PC"/>
            <generic:Value concept="UNIT_MULT" value="0"/>
            <generic:Value concept="DECIMALS" value="2"/>
            <generic:Value concept="TITLE_COMPL" value="Elements of the full national etc."/>
         </generic:Attributes>
         <generic:Series>
            <generic:SeriesKey>
               <generic:Value concept="FREQ" value="M"/>
               <generic:Value concept="REF_AREA" value="IT"/>
               <generic:Value concept="ADJUSTMENT" value="W"/>
               <generic:Value concept="STS_INDICATOR" value="PROD"/>
               <generic:Value concept="STS_ACTIVITY" value="NS0020"/>
               <generic:Value concept="STS_INSTITUTION" value="1"/>
               <generic:Value concept="STS_BASE_YEAR" value="2000"/>
            </generic:SeriesKey>
            <generic:Attributes>
               <generic:Value concept="COLLECTION" value="A"/>
               <generic:Value concept="AVAILABILITY" value="A"/>
               <generic:Value concept="TIME_FORMAT" value="P1M"/>
            </generic:Attributes>
            <generic:Obs>
               <generic:Time>2005-01</generic:Time>
               <generic:ObsValue value="111.11"/>
                  <generic:Attributes>
                     <generic:Value concept="OBS_STATUS" value="A"/>
                     <generic:Value concept="OBS_CONF" value="F"/>
                  </generic:Attributes>
            </generic:Obs>
            <generic:Obs>
               <generic:Time>2005-02</generic:Time>
               <generic:ObsValue value=« 222.22"/>
                  <generic:Attributes>
                     <generic:Value concept="OBS_STATUS" value="A"/>
                     <generic:Value concept="OBS_CONF" value="F"/>
                  </generic:Attributes>
            </generic:Obs>
         </generic:Series>
         <generic:Series>
          …
         </generic:Series>
```

Figure 3 - SDMX-ML Generic data message

The data in Generic format consists of four different levels in a nested arrangement. These levels, which are actually XML elements, viewed from the outer to the inner, are:

- **<DataSet>**: is the container of the DataSet information. It includes all SDMX Attributes attached at the Dataset level and any Groups included in the DataSet. In case no Groups are reported, Series are included directly under the <DataSet> element;
    - **<Group>**: One <Group> element is defined for each SDMX Group Key. It includes all SDMX Attributes attached at the particular Group. This element includes the Time-Series (<Series> element) reported sharing the same partial key (Group Key). Each dimension included in the Group Key has identical values for all included Series. There is no obligation that a data message has to report a group declared in the DSD, if no attached SDMX attributes need to be reported.
        - **<Series>**: One <Series> element per Key is defined and includes the SDMX Attributes attached at the Series level.
          <Series> includes in addition the <Obs> elements that represent the actual (time related) measurements reported.
            - **<Obs>**: One <Obs> element per observation is included in each <Series>. Each <Obs> element contains all SDMX Attributes attached at the Observation level, the time of the observation and the observation value.

The Generic format is a 'self-descriptive' message. The XML elements (DataSet, Group, Series, Obs, Attributes) and the XML attributes (concept, value) included in this Generic message can be understood and interpreted independently from the DSD. The semantic of the concepts participating can be derived from the position and context they reside.

Example:

For the Generic message presented above, in the XML element <generic:ObsValue> the XML attribute 'value' holds the observation '111.11':

```
<generic:ObsValue value="111.11"/>
```

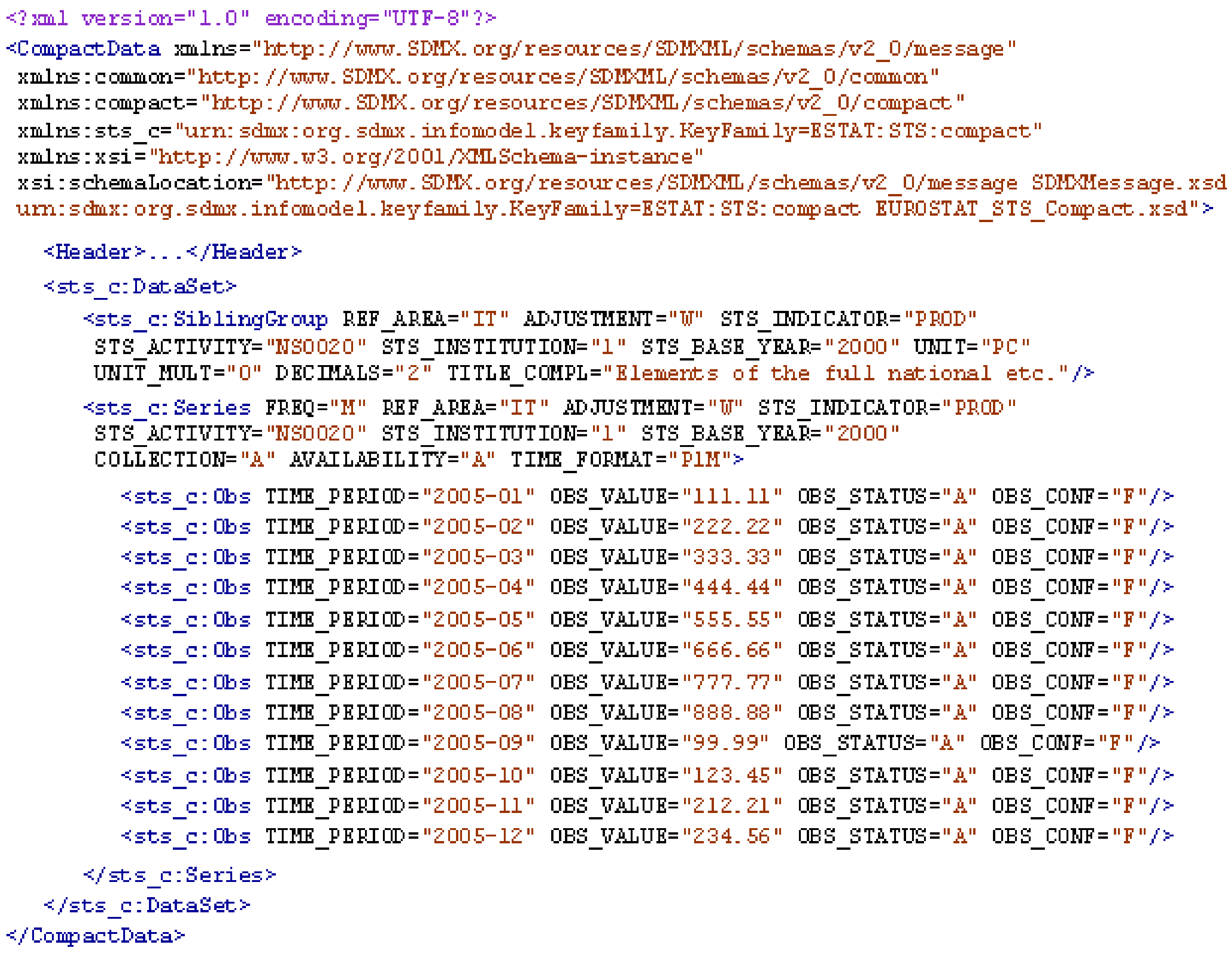
But the same XML attribute 'value' in a <generic:Value> XML element represents the value 'IT' of the XML Attribute representing the SDMX concept 'REF_AREA':

```
<generic:Value concept="REF_AREA" value="IT"/>
```

4.4 Compact SDMX-ML Data set

4.4.1 Compact data message characteristics

- As an efficien & concise format it is used for the exchange of large data sets in a data structure definition-dependent form in XML format;
- It is specific to and dependent on the data structure definition of the data set it encodes. Thus, it follows mappings between constructs in Structure Definition message and compact format;
- For transmission of partial data sets (incremental updates) and whole data sets;
- In terms of XML syntax, all codes and observation values are attributes. The permissible values of the codes are defined in the schema (which is specific to the data structure definition) so that a generic XML parser can be used to validate a data file against its structural definition including codes for values.
- For the compact format key values can also be defined at Group level;



```xml
<?xml version="1.0" encoding="UTF-8"?>
<CompactData xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message"
 xmlns:common="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common"
 xmlns:compact="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/compact"
 xmlns:sts_c="urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:compact"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd
 urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:compact EUROSTAT_STS_Compact.xsd">
   <Header>...</Header>
   <sts_c:DataSet>
      <sts_c:SiblingGroup REF_AREA="IT" ADJUSTMENT="W" STS_INDICATOR="PROD"
       STS_ACTIVITY="NS0020" STS_INSTITUTION="1" STS_BASE_YEAR="2000" UNIT="PC"
       UNIT_MULT="0" DECIMALS="2" TITLE_COMPL="Elements of the full national etc."/>
      <sts_c:Series FREQ="M" REF_AREA="IT" ADJUSTMENT="W" STS_INDICATOR="PROD"
       STS_ACTIVITY="NS0020" STS_INSTITUTION="1" STS_BASE_YEAR="2000"
       COLLECTION="A" AVAILABILITY="A" TIME_FORMAT="P1M">
         <sts_c:Obs TIME_PERIOD="2005-01" OBS_VALUE="111.11" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-02" OBS_VALUE="222.22" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-03" OBS_VALUE="333.33" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-04" OBS_VALUE="444.44" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-05" OBS_VALUE="555.55" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-06" OBS_VALUE="666.66" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-07" OBS_VALUE="777.77" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-08" OBS_VALUE="888.88" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-09" OBS_VALUE="99.99" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-10" OBS_VALUE="123.45" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-11" OBS_VALUE="212.21" OBS_STATUS="A" OBS_CONF="F"/>
         <sts_c:Obs TIME_PERIOD="2005-12" OBS_VALUE="234.56" OBS_STATUS="A" OBS_CONF="F"/>
      </sts_c:Series>
   </sts_c:DataSet>
</CompactData>
```

Figure 4 - SDMX-ML Compact data message

Compact DataSet is one of the DSD-specific and DSD-dependent formats.

In the top declarations of SDMX-ML Compact data message, a DSD specific Compact Schema (XSD) – in the example: EUROSTAT_STS_Compact.xsd – is included in order to enable syntactical validation of the message:

```
xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd
urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:compact EUROSTAT STS Compact.xsd">
```

This DSD-specific scheme can be found on Eurostat SDMX registry.

The layout of the SDMX-ML Compact data format structure, contains the Groups at the same level as the Series within the <DataSet> element. In the above example this is shown for the elements < SiblingGroup> and <Series>.

A <Series> element is common to any Compact data message, but the <SiblingGroup> element is included in the example, since the underlying DSD has this group defined. The Group's name is the Group-ID defined in the DSD.
The names of the contained XML attributes are the IDs of the Dimensions and Attributes established in the DSD. The values of these XML attributes are the values of the Dimensions forming the GroupKey and the Attributes attached at this Group.

Similarly, in the <Series> element the XML attributes holding the SDMX Dimensions and all SDMX Attributes attached at the Series level. Here the SDMX Dimensions forming the Key of the Series.

Finally, all observations are represented by the <Obs> elements within the <Series> element. Within the <Obs> element, the SDMX Time Dimension, Observation values and Attributes attached at observation level are included as XML attributes.

Thus, the structure for the SDMX-ML compact data message is as follows:

- **< DataSet>**: is the container of the DataSet information. It includes all SDMX Attributes attached at the Dataset level and any Groups included in the DataSet. In case no Groups are reported, Series are included directly under the <DataSet> element;
    - **<Group>**: One <Group> element is defined for each Group Key (like the SiblingGroup) defined in the DSD. It includes all SDMX Attributes attached at the particular Group. Each dimension included in the Group Key has identical values are for all included Series.
    - **< Series>**: One <Series> element per Key is defined. It includes the SDMX Attributes attached at the Series level and <Obs> elements that represent the actual (time related) measurements reported.
        - **< Obs>**: One <Obs> element per observation is included in each <Series>. Each <Obs> element contains all SDMX Attributes attached at the Observation level, the time and the observation value.

The Compact SDMX-ML data message is capable of conveying data at a much smaller size than the equivalent Generic one. This message does not reveal the context of the concepts reported, so that neither a human actor nor a software can determine the role (SDMX Dimension or Attribute) of an XML attributes included in the <Series> element of the Compact data message. The XSD derived from the corresponding DSD allows for syntax validation of XML elements, attributes and XML attribute values.

Time Ranges in CompactData:

Unlike any other SDMX-ML data format, the DSD-specific CompactData format can express a set of observation values without having to provide a time for each of them. If a Series has a time provided for the first observation, subsequent observations in the series may omit the time, and only provide an observation value and the related attributes. The times of the following observations can be calculated by an application according to the frequency specified by the related time format attribute value or the frequency dimension value. It is also allowed to supply a time value for the last observation in the series, to permit optionally double-checking of the calculation.

Delete and Update Messages in CompactData:

In the Header element and action attribute at the DataSet level, the action field specifies whether a message is an update message (to append or replace) or a delete message.

- An update message is used to send new information or updated information, which may include updated observations or attribute values.
- For a delete message, the requirements are that a complete series key always be sent for the deletion of data. Thus, it is not supported to delete multiple series, by submitting only the corresponding group key.
  Deletion can be performed for entire series, if no time periods are specified, or a set of reported time periods. Attribute values may be deleted by sending a complete or partial set of attributes, with any valid value for the attribute (including an empty value - if the attribute is not mandatory) indicating that the current attribute value should be deleted.

4.5 Utility SDMX-ML Data set

4.5.1 Utility data message characteristics

- The message is specific to and dependent on the data structure definition of the data set;
- It may be considered a special-purpose message for extended validation and other XML schema based functions;
- Cannot be used for incremental updates, since it requires according to the SDMX standard the sending of a complete data set;
- The key values are generally found at the series level.

```
<?xml version="1.0" encoding="UTF-8"?>
<UtilityData
 xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message"
 xmlns:sts_u="urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:utility"
 xmlns:utility="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/utility"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd
 urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:utility ESTAT_STS_Utility.xsd">
   <Header>…</Header>
   <sts_u:DataSet>
      <sts_u:SiblingGroup UNIT="PC" UNIT_MULT="0" DECIMALS="2"
       TITLE_COMPL="Elements of the full national etc.">
         <sts_u:Series COLLECTION="A" AVAILABILITY="A" TIME_FORMAT="P1M">
            <sts_u:Key>
               <sts_u:FREQ>M</sts_u:FREQ>
               <sts_u:REF_AREA>IT</sts_u:REF_AREA>
               <sts_u:ADJUSTMENT>W</sts_u:ADJUSTMENT>
               <sts_u:STS_INDICATOR>PROD</sts_u:STS_INDICATOR>
               <sts_u:STS_ACTIVITY>NS0020</sts_u:STS_ACTIVITY>
               <sts_u:STS_INSTITUTION>1</sts_u:STS_INSTITUTION>
               <sts_u:STS_BASE_YEAR>2000</sts_u:STS_BASE_YEAR>
            </sts_u:Key>
            <sts_u:Obs OBS_CONF="F" OBS_STATUS="A">
               <sts_u:TIME_PERIOD>2005-01</sts_u:TIME_PERIOD>
               <sts_u:OBS_VALUE>111.11</sts_u:OBS_VALUE>
            </sts_u:Obs>
            <sts_u:Obs OBS_CONF="F" OBS_STATUS="A">
               <sts_u:TIME_PERIOD>2005-02</sts_u:TIME_PERIOD>
               <sts_u:OBS_VALUE>222.22</sts_u:OBS_VALUE>
            </sts_u:Obs>
            <sts_u:Obs OBS_CONF="F" OBS_STATUS="A">
               <sts_u:TIME_PERIOD>2005-03</sts_u:TIME_PERIOD>
               <sts_u:OBS_VALUE>333.33</sts_u:OBS_VALUE>
               …
```

Figure 5 - SDMX-ML Utility data message

The Utility SDMX-ML data format was included in the SDMX standard as a specific data message allowing extended data validation and other XSD based operations. Since it is DSD specific, the DSD specific Utility XSD link has to be included in the top declarations of this message:

```
xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd
urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:utility ESTAT_STS_Utility.xsd">
```

This enables the syntactical validation of the message.

The layout of the Utility format adopts a four-level nesting of XML elements:

- **<DataSet>**: It represents the top-level element that includes Group elements, for example the <SiblingGroup>. It may also include XML attributes to represent DSD Attributes attached at the DataSet level.
    - **<Group>**: A DSD specific element (named after GroupID of the related DSD). The XML attributes included in this element represent the DSD Attributes

attached to this Group. In contrary to the Compact format the Dimensions forming the GroupKey are not included here.

- **<Series>**: This element contains DSD specific XML attributes representing the DSD Attributes attached at Series level. In addition, it includes the <Key> element to describe the Key (the values of the key Dimensions) and the <Obs> elements representing the observations.
    - **<Key>**: This element includes the DSD specific key Dimensions with its values, to identify the Series. The incorporated naming of the dimensions as XML elements (FREQ, REF_AREA, ADJUSTMENT, etc.) enables to validate the order of the Key elements using the DSD specific XSD.
    - **<Obs>**: includes the SDMX Attributes attached at the observation level. It also contains DSD specific elements for reporting Time period and Observation value.

The purpose of this message type is to enforce sophisticated validation using the DSD specific XSD. It is neither small in size as the Compact nor self explanatory as the Generic format.

According to the SDMX standard the UtilityData DSD-specific schema cannot express partial time ranges of observation values (i.e. it is not possible to submit only a part of a time series). Also a 'delete' message is not specified and an available action field in the message header would be ignored. Consequently, the single message type for utility schema is the 'update' message containing the entire set of attributes and observation values for the transmitted time series.

4.6 Cross-sectional SDMX-ML Data set

4.6.1 Cross-sectional data message characteristics

- The Cross-sectional data message Exchange is data structure definition specific and dependent. It is intended for data organisations where the statistical data consist of multiple observations at a single point in time. Thus, it contains multiple observation values for different 'Cross-sectional measures'. For example, in foreign trade statistics where, for combination of reporting country, partner country, commodity and time period there may be three observations: a value, a weight and a quantity;
- The Cross-Sectional format is the only one capable of expressing non-Time series DSDs. The Dimension Time period and Frequency are not mandatory in a Cross-Sectional DSD. As an exception, the SDMX standard allows to employ the Primary measure concept, if no measures are defined in the Cross-sectional format.
- Key values are found from the Group down to the Observation level;
- The time dimension (if present) is found at a higher level (e.g. DataSet or Group level).

```
<?xml version="1.0" encoding="UTF-8"?>
<CrossSectionalData
 xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message"
 xmlns:common="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common"
 xmlns:cross="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross"
 xmlns:demo="urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:cross"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message
 SDMXMessage.xsd urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:cross
 EUROSTAT_STS_Cross.xsd">
 <Header>...</Header>
 <demo:DataSet REV_NUM="1" TAB_NUM="RQFI05V1">
    <demo:Group COUNTRY="FI" FREQ="A" TIME="2005" TIME_FORMAT="P1Y">
        <demo:Section DECI="0" UNIT="PERS" UNIT_MULT="0">
            <demo:ADJT OBS_STATUS="P" SEX="F" value="35"/>
            <demo:DEATHST OBS_STATUS="P" SEX="F" value="23871"/>
            <demo:LBIRTHST OBS_STATUS="P" SEX="F" value="28345"/>
            <demo:NETMT OBS_STATUS="P" SEX="F" value="4187"/>
            <demo:PJAN1T OBS_STATUS="P" SEX="F" value="2683230"/>
            <demo:PJANT OBS_STATUS="P" SEX="F" value="2674534"/>
            <demo:ADJT OBS_STATUS="P" SEX="M" value="131"/>
            <demo:DEATHST OBS_STATUS="P" SEX="M" value="24057"/>
            <demo:LBIRTHST OBS_STATUS="P" SEX="M" value="29400"/>
            <demo:NETMT OBS_STATUS="P" SEX="M" value="4799"/>
            <demo:PJAN1T OBS_STATUS="P" SEX="M" value="2572350"/>
            <demo:PJANT OBS_STATUS="P" SEX="M" value="2562077"/>
            <demo:ADJT OBS_STATUS="P" SEX="T" value="166"/>
            <demo:DEATHST OBS_STATUS="P" SEX="T" value="47928"/>
            <demo:LBIRTHST OBS_STATUS="P" SEX="T" value="57745"/>
            <demo:NETMT OBS_STATUS="P" SEX="T" value="8986"/>
            <demo:PJAN1T OBS_STATUS="P" SEX="T" value="5255580"/>
            <demo:PJANT OBS_STATUS="P" SEX="T" value="5236611"/>
        </demo:Section>
        <demo:Section DECI="0" UNIT="PURE_NUMB" UNIT_MULT="0">
            <demo:DIV OBS_STATUS="P" SEX="T" value="13383"/>
            <demo:MAR OBS_STATUS="P" SEX="T" value="29283"/>
        </demo:Section>
        <demo:Section DECI="3" UNIT="PURE_NUMB" UNIT_MULT="0">
            <demo:TFRNSI SEX="T" value="1800"/>
        </demo:Section>
    </demo:Group>
 </demo:DataSet>
 </CrossSectionalData>
```

Figure 6 - SDMX-ML Cross-sectional data message

In the top declarations of this message, a DSD specific Cross-sectional XSD has to be included in order to enable syntactical validation of the message.

```
xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message
SDMXMessage.xsd urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=ESTAT:STS:cross
EUROSTAT_STS_Cross.xsd">
```

Although there are also four levels defined in the Cross-Sectional format, only the <DataSet> element is used in the same way.

The <Group> element is independent of the Groups included in the DSD.

The <Section> element replaces the <Series> element in the sense that it does not include a Time series but a Section representing a 'vertical slice' across several Time-series. The content of one slice/one section is multiple measures/observation values for a single point in time. These observation values are now included at the fourth level in DSD specific Cross-sectional Measures.

The DSD Dimensions and Attributes can be attached at any of the four levels according to the definition in the related DSD. Thus, the levels in the example look as follows:

- **<DataSet>**: It represents the top-level element that includes the Group elements. It may also include XML attributes to represent DSD Attributes attached at the DataSet level.
    - **<Group>**: A Cross-sectional grouping element independent from the groups defined in the related DSD Attributes. The Group Key are also included here.
        - **<Section>**: One <Section> element per Key is defined. This element carries DSD specific XML attributes representing the Key and DSD Attributes attached at Section level. In addition, the <**'Cross-sectional measure/observation'**> elements representing the Cross-sectional measures measures/observations are included.
            - **<'Cross-sectional measure observation'>**: includes the part of the Key to identify the observation and the XML attributes for SDMX Attributes attached at the observation level for each Cross-sectional measures/observations reported.

The Cross-sectional data message is applied for reporting Data sets including multiple measures and/or the exchange of data not organized in Time series. The size of the resulting messages in this format is comparably small.

Delete and Update Messages in CrossSectionalData:

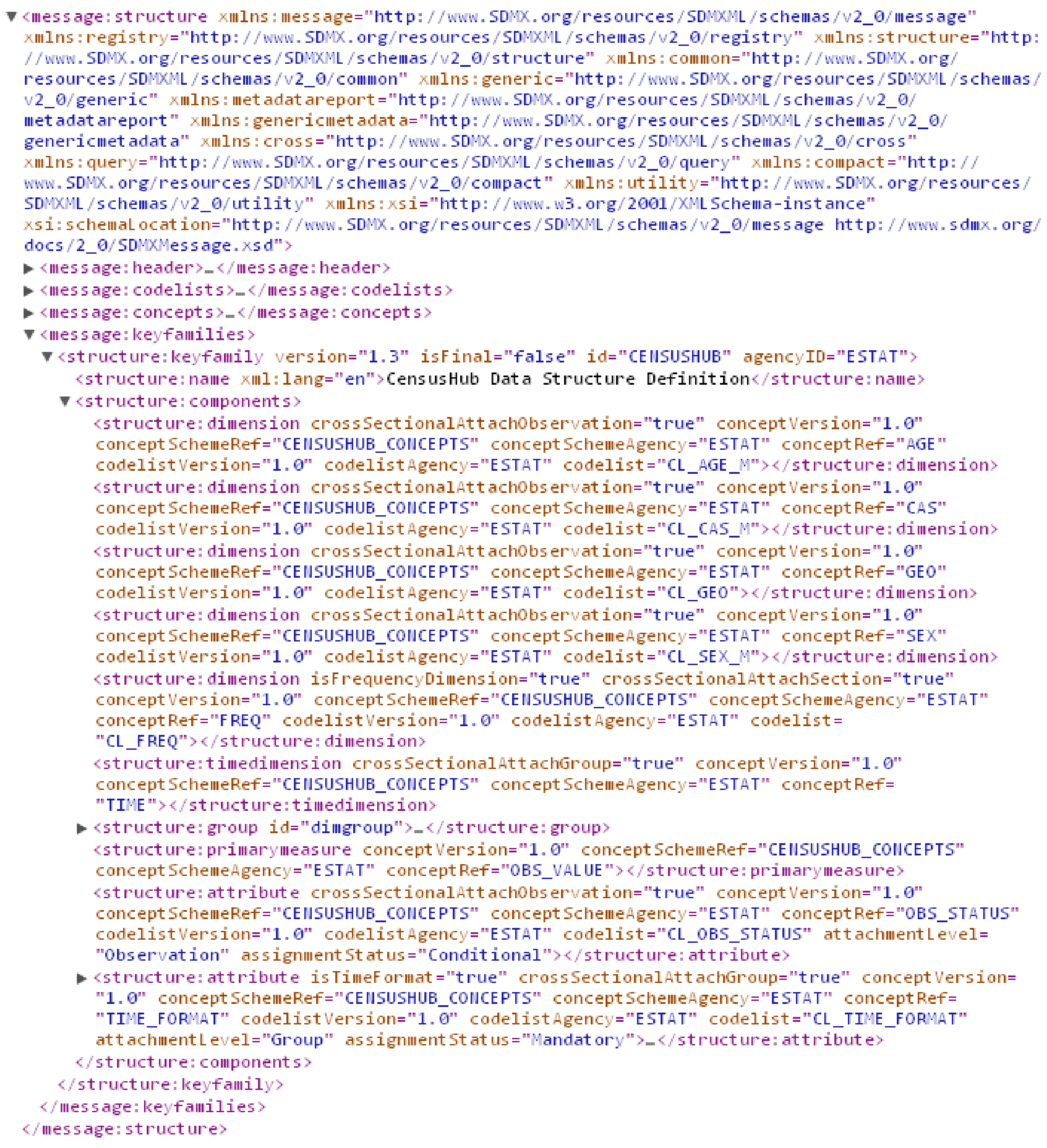
For CrossSectionalData both are defined in SDMX standard:

- Update message (Append, Replace), which submits new information or updated information;
- Delete message, where the entire key has to be sent for the deletion of data. The data to be deleted is identified either as an entire section or a set of measures within a section. Attribute values may be deleted by sending a complete or partial set of attributes, with any valid value for the attribute (including an empty value - according to the related XSD schema) indicating that the current attribute value should be deleted.

4.7 SDMX-ML Structure Definition Message

The SDMX-ML Structure Definition Message contains a Structure definition (e.g. a DSD) with or without the referenced artefacts. It may also comprise one or several stand-alone code lists, concept schemes, categories, etc. (without a link to a DSD).

- All SDMX-ML message types share this common XML expression of the metadata needed to understand and process a data set or metadata set;
- ConceptSchemes and codelists may be included as referenced artefacts within a structure message containing a DSD or MSD;
- Annotations are supported a various levels within the message;



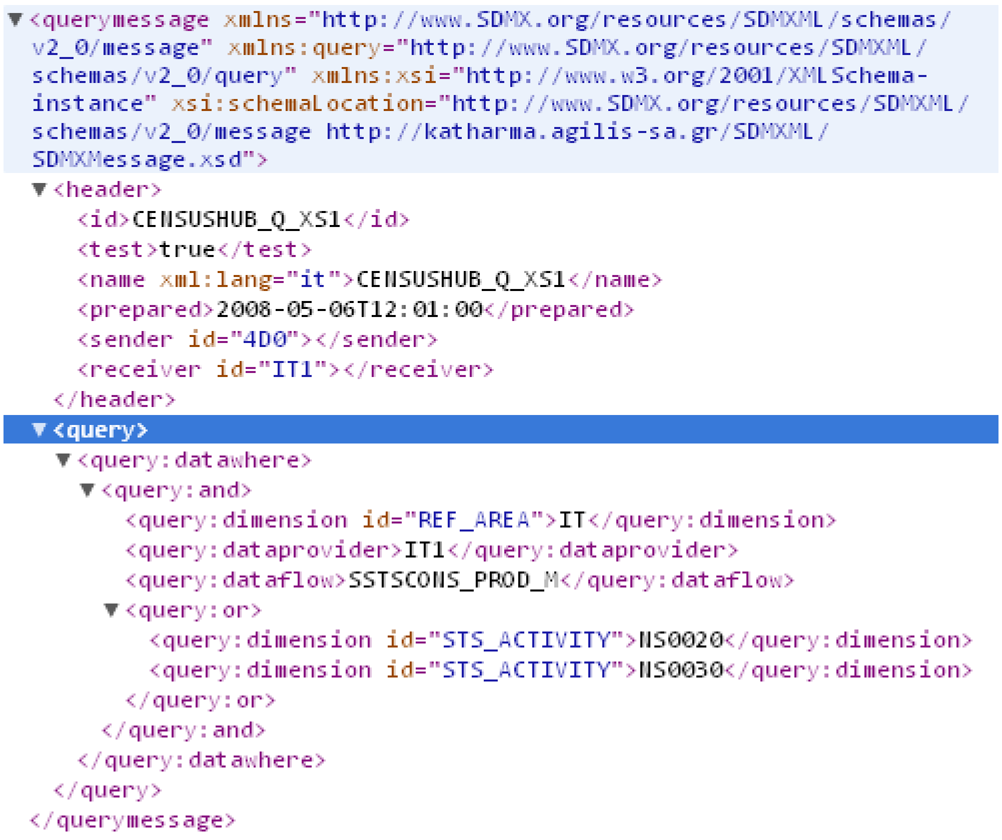
```
▼<message:structure xmlns:message="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message"
  xmlns:registry="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/registry" xmlns:structure="http:
  //www.SDMX.org/resources/SDMXML/schemas/v2_0/structure" xmlns:common="http://www.SDMX.org/
  resources/SDMXML/schemas/v2_0/common" xmlns:generic="http://www.SDMX.org/resources/SDMXML/schemas/
  v2_0/generic" xmlns:metadatareport="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/
  metadatareport" xmlns:genericmetadata="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/
  genericmetadata" xmlns:cross="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross"
  xmlns:query="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/query" xmlns:compact="http://
  www.SDMX.org/resources/SDMXML/schemas/v2_0/compact" xmlns:utility="http://www.SDMX.org/resources/
  SDMXML/schemas/v2_0/utility" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message http://www.sdmx.org/
  docs/2_0/SDMXMessage.xsd">
  ▶<message:header>…</message:header>
  ▶<message:codelists>…</message:codelists>
  ▶<message:concepts>…</message:concepts>
  ▼<message:keyfamilies>
    ▼<structure:keyfamily version="1.3" isFinal="false" id="CENSUSHUB" agencyID="ESTAT">
        <structure:name xml:lang="en">CensusHub Data Structure Definition</structure:name>
      ▼<structure:components>
          <structure:dimension crossSectionalAttachObservation="true" conceptVersion="1.0"
          conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef="AGE"
          codelistVersion="1.0" codelistAgency="ESTAT" codelist="CL_AGE_M"></structure:dimension>
          <structure:dimension crossSectionalAttachObservation="true" conceptVersion="1.0"
          conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef="CAS"
          codelistVersion="1.0" codelistAgency="ESTAT" codelist="CL_CAS_M"></structure:dimension>
          <structure:dimension crossSectionalAttachObservation="true" conceptVersion="1.0"
          conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef="GEO"
          codelistVersion="1.0" codelistAgency="ESTAT" codelist="CL_GEO"></structure:dimension>
          <structure:dimension crossSectionalAttachObservation="true" conceptVersion="1.0"
          conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef="SEX"
          codelistVersion="1.0" codelistAgency="ESTAT" codelist="CL_SEX_M"></structure:dimension>
          <structure:dimension isFrequencyDimension="true" crossSectionalAttachSection="true"
          conceptVersion="1.0" conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT"
          conceptRef="FREQ" codelistVersion="1.0" codelistAgency="ESTAT" codelist=
          "CL_FREQ"></structure:dimension>
          <structure:timedimension crossSectionalAttachGroup="true" conceptVersion="1.0"
          conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef=
          "TIME"></structure:timedimension>
        ▶<structure:group id="dimgroup">…</structure:group>
          <structure:primarymeasure conceptVersion="1.0" conceptSchemeRef="CENSUSHUB_CONCEPTS"
          conceptSchemeAgency="ESTAT" conceptRef="OBS_VALUE"></structure:primarymeasure>
          <structure:attribute crossSectionalAttachObservation="true" conceptVersion="1.0"
          conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef="OBS_STATUS"
          codelistVersion="1.0" codelistAgency="ESTAT" codelist="CL_OBS_STATUS" attachmentLevel=
          "Observation" assignmentStatus="Conditional"></structure:attribute>
        ▶<structure:attribute isTimeFormat="true" crossSectionalAttachGroup="true" conceptVersion=
          "1.0" conceptSchemeRef="CENSUSHUB_CONCEPTS" conceptSchemeAgency="ESTAT" conceptRef=
          "TIME_FORMAT" codelistVersion="1.0" codelistAgency="ESTAT" codelist="CL_TIME_FORMAT"
          attachmentLevel="Group" assignmentStatus="Mandatory">…</structure:attribute>
        </structure:components>
      </structure:keyfamily>
    </message:keyfamilies>
  </message:structure>
```

Figure 7- SDMX-ML Data Structure Message without references artefacts

4.8 SDMX-ML Query Message

Message used to query a database to obtain an SDMX-ML message (data or structure) as the result.

- The aim of this message is to convey a query to a database which then returns an SDMX-ML data or structure message representing the result;
- The query message is employed to interact with web services and database-driven applications;
- Queries may be issued with regard to data and structural metadata;



```
▼<querymessage xmlns="http://www.SDMX.org/resources/SDMXML/schemas/
 v2_0/message" xmlns:query="http://www.SDMX.org/resources/SDMXML/
 schemas/v2_0/query" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
 instance" xsi:schemaLocation="http://www.SDMX.org/resources/SDMXML/
 schemas/v2_0/message http://katharma.agilis-sa.gr/SDMXML/
 SDMXMessage.xsd">
  ▼<header>
     <id>CENSUSHUB_Q_XS1</id>
     <test>true</test>
     <name xml:lang="it">CENSUSHUB_Q_XS1</name>
     <prepared>2008-05-06T12:01:00</prepared>
     <sender id="4D0"></sender>
     <receiver id="IT1"></receiver>
   </header>
  ▼<query>
    ▼<query:datawhere>
      ▼<query:and>
         <query:dimension id="REF_AREA">IT</query:dimension>
         <query:dataprovider>IT1</query:dataprovider>
         <query:dataflow>SSTSCONS_PROD_M</query:dataflow>
        ▼<query:or>
           <query:dimension id="STS_ACTIVITY">NS0020</query:dimension>
           <query:dimension id="STS_ACTIVITY">NS0030</query:dimension>
         </query:or>
       </query:and>
     </query:datawhere>
   </query>
 </querymessage>
```

Figure 8 - SDMX-ML Query Message

5 SDMX message types for Reference metadata

5.1 Overview

Three standard messages are used for reference metadata and MSDs:

- The Structure Message contains a Metadata Structure Definition. The schema for this Structure Message is the same as for the one for data. It is provided by the SDMX standard.
- The Generic Metadata Message conveys reference metadata in a form independent of a Metadata structure definition. The schema is fixed and provided by the SDMX standard.
- The Metadata Report Message is the message for the exchange of reference metadata according to a specific MSD. The schema is derived from the Metadata Structure Message.

5.2 Structure Message

- The Structure Message contains the Metadata Structure Definition: all SDMX-ML Metadata Report Message types using the same MSD share this common structure of the metadata needed to understand and reuse a data set;
- It supports XML annotations containing explanations and further specification of the content of metadata to be provided (for instance guidelines for compilation);

Metadata concepts are described in terms of their content (definition, guidelines) representation (free text or code list) and usage (such as mandatory or optional). If a concept has to be represented by a code, the relevant code lists needs to be referenced from within the message.

The MSD also describes the structure of the Metadata Report to be exchanged. The Metadata Report is normally composed of a hierarchy of metadata concepts which depends on the kind of metadata elements that a maintaining agency intends to exchange.

The ESMS Report Structure was recently defined by Eurostat containing 21 main concepts and sub-concepts. References on the code lists for coded concepts, as well as the 'Usage Status' (Mandatory/Conditional) are included.

```
– <structure:ReportStructure id="ESMS_FULL_REPORT" target="CATEGORY_PROVIDER_TARGET"
    urn="urn:sdmx:org.sdmx.infomodel.metadatastructure.ReportStructure=EUROSTAT:ESMS[1.0].ESMS_FULL_REPORT">
    <structure:Name xml:lang="en">Full ESMS Report Structure</structure:Name>
  – <structure:MetadataAttribute conceptRef="CONTACT" usageStatus="Mandatory" conceptSchemeRef="CROSS_DOMAIN_CONCEPTS"
      conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
    – <structure:MetadataAttribute conceptRef="CONTACT_ORGANISATION" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" maxLength="70" />
      </structure:MetadataAttribute>
    – <structure:MetadataAttribute conceptRef="CONTACT_ORGANISATION_UNIT" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" maxLength="70" />
      </structure:MetadataAttribute>
    – <structure:MetadataAttribute conceptRef="CONTACT_NAME" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" maxLength="70" />
      </structure:MetadataAttribute>
    – <structure:MetadataAttribute conceptRef="CONTACT_MAIL" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" />
      </structure:MetadataAttribute>
      <structure:MetadataAttribute conceptRef="CONTACT_FUNCT" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0"
        representationScheme="CL_FUNCTION" representationSchemeAgency="EUROSTAT" />
    – <structure:MetadataAttribute conceptRef="CONTACT_EMAIL" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" />
      </structure:MetadataAttribute>
    – <structure:MetadataAttribute conceptRef="CONTACT_PHONE" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" />
      </structure:MetadataAttribute>
    – <structure:MetadataAttribute conceptRef="CONTACT_FAX" usageStatus="Conditional"
        conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
        <structure:TextFormat textType="String" />
      </structure:MetadataAttribute>
    </structure:MetadataAttribute>
  + <structure:MetadataAttribute conceptRef="METADATA_UPDATE" usageStatus="Conditional"
      conceptSchemeRef="CROSS_DOMAIN_CONCEPTS" conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
  + <structure:MetadataAttribute conceptRef="STAT_PRES" usageStatus="Conditional" conceptSchemeRef="CROSS_DOMAIN_CONCEPTS"
      conceptSchemeAgency="EUROSTAT" conceptVersion="1.0">
  </structure:ReportStructure>
```

Figure 9: ESMS report structure SDMX-ML sample.

5.3 Generic Metadata Message

The Generic Metadata Message contains reported metadata in an XML format which supports any Metadata structure definition. All reference metadata expressible in SDMX-ML format can be marked up according to this format.

- Only a minimum of validation can be performed;
- It eases the creation of generic software tools and services for processing reference metadata, since no MSD definitions need to be taken into account.

5.4 Metadata Report message

Metadata Report message contains a metadata report which is specific to a particular MSD. This format allows for the validation of the codes and structures described in the Metadata structure definition with a generic XML parser.

- For each MSD, an MSD-specific XML schema can be derived;
- Validation against these schemes on sets of reported data may be conducted;
- This message needs the link to the respective MSD for a full interpretation of the content;
- The XML mark-up relates directly to the reported concepts.

6 SDMX-ML Schemas

The following sections describe in the SDMX XML schemas ruling the various SDMX messages. These schemas are divided into the generic schemas, for which a complete set of schema definitions is provided in the SDMX standard, and DSD-specific schemas, for which only the core structure is provided as standard schemas.

For the DSD-specific schemas, there is a guide available to how a specific DSD or MSD can be mapped onto the core structure. Respective XSL/XSLT transformation files are included in various SDMX tool packages (e.g. Eurostat's SDMX Registry package) for the performance of the transformation of a particular Data Structure Definition to its dependent XSD schemas (in Compact, Utility or Cross-sectional format).

6.1 Structure definition independent schemas

The core message structures and the special purpose messages for querying or Registry services are defined by the following set of DSD/MSD independent standard schemas:

SDMXMessage.xsd, SDMXCommon.xsd, SDMXStructure.xsd, SDMXQuery.xsd, SDMXRegistry.xsd, SDMXGenericData.xsd and SDMXGenericMetadata.xsd.

Of these, only the SDMXStructure message and the SDMXGenericData message are required for general exchange of SDMX-ML Generic data sets.

For generic exchange of reference metadata, only the SDMXStructure message and the SDMXGenericMetadata message are required.

6.2 Structure Definition dependent schemas

Other schemas are specific to and dependent on Data Structure Definitions and Metadata Structure Definitions, and therefore there is no single schema for all type of data to be exchanged.

In these cases, standard transformations are provided so that the schemas can be predicted from an examination of the related SDMX-ML Structure messages (DSD or MSD files). This correspondence allows automatic creation of structure-specific schemas by SDMX-tools running the dedicated transformation files (XSL/XSLT).

All DSD- and MSD-specific schemas are based on a core of identical constructs, allowing the smallest possible number of tags to differ from schema to schema. The structure dependent schemas are similar within the respective message format. They vary only for key values and attributes specified within the common structure.

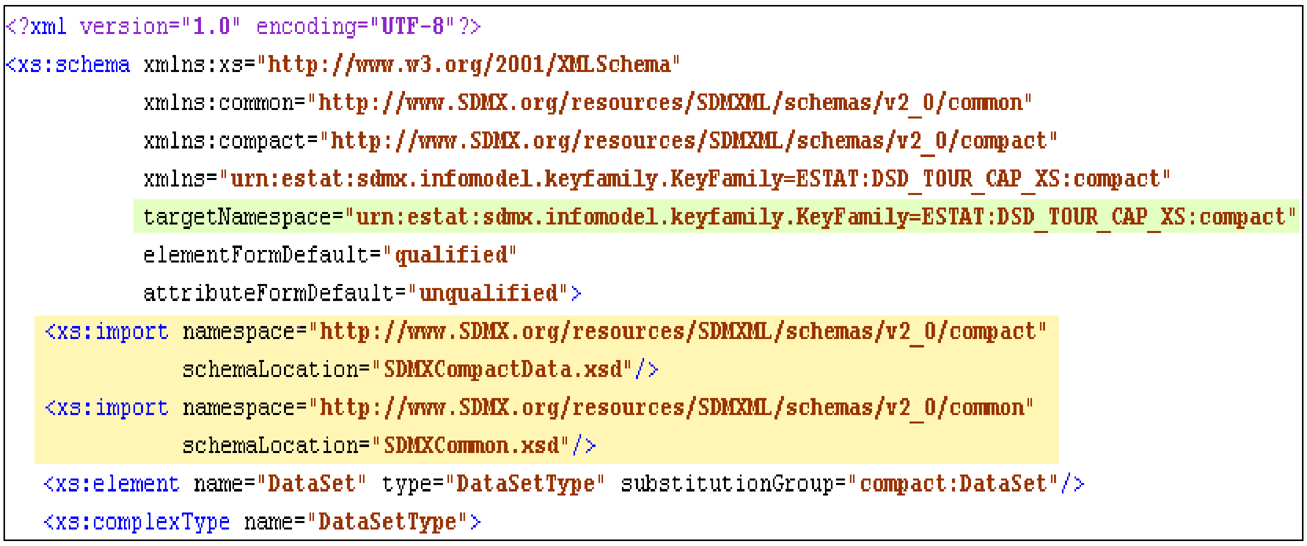
The differences between several Utility or Metadata Report schemas is even lower, since both schemas are designed to contain as much structural metadata as possible in order to allow 'typical' XML tools to perform validation based on this information. Such tools are generally incapable of consulting directly the DSD or MSD for structural metadata.

6.2.1 General Rules

For all DSD-specific schemas (Compact, Utility, and Cross-Sectional) the SDMX standard provides a namespace to be used as the extension base for DSD schemas of that type.

The DSD-specific schema will be created in its own target namespace, owned and maintained by the creating agency. It will use the **targetNamespace** attribute of the schema element to identify the namespace which contains the DSD-specific schema.

The namespace module provided by SDMX for that class of DSD-specific schema (Compact – in the example in Figure 10) will be incorporated using the import element in the DSD-specific schema. The SDMX Common namespace module must also be imported into the schema.



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:common="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common"
           xmlns:compact="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/compact"
           xmlns="urn:estat:sdmx.infomodel.keyfamily.KeyFamily=ESTAT:DSD_TOUR_CAP_XS:compact"
           targetNamespace="urn:estat:sdmx.infomodel.keyfamily.KeyFamily=ESTAT:DSD_TOUR_CAP_XS:compact"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/compact"
              schemaLocation="SDMXCompactData.xsd"/>
   <xs:import namespace="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common"
              schemaLocation="SDMXCommon.xsd"/>
   <xs:element name="DataSet" type="DataSetType" substitutionGroup="compact:DataSet"/>
   <xs:complexType name="DataSetType">
```

Figure 10 - TargetNamespace and import of namespaces

Other xml:namespace attributes may be added to the schema element as needed.

All additions to this construct use the ‘xs:extension’ element from W3C XML Schema. The term ‘levels of structure’, when referring to the imported SDMX modules, include the following:

- DataSet level
- Group level
- Series or Section level
- Observation level

These levels refer to the element provided by the SDMX module to which attributes and elements may be assigned. The extension for the ‘Series level’ is illustrated in Figure 11.

```
<xs:element name="Series" substitutionGroup="compact:Series" type="SeriesType"/>
<xs:complexType name="SeriesType">
   <xs:complexContent>
      <xs:extension base="compact:SeriesType">
         <xs:sequence>
            <xs:element ref="Obs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Annotations" type="common:AnnotationsType" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="TIME_FORMAT" type="CL_TIME_FORMAT" use="optional"/>
         <xs:attribute name="UNIT" type="CL_UNIT" use="optional"/>
         <xs:attribute name="FREQ" type="CL_FREQ" use="optional"/>
         <xs:attribute name="COUNTRY" type="CL_COUNTRY" use="optional"/>
         <xs:attribute name="INDIC_TO" type="CL_TOUR_INDICAT" use="optional"/>
         <xs:attribute name="ACTIVITY_TO" type="ACTIVITY_TOType" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

Figure 11 - xs:extension for Series level

For all of the DSD-specific mappings, SDMX-ML namespace modules are identified with the abbreviations used in the standard schemas (‘compact:’ refers to the CompactData module; ‘common:’ to the Common namespace module, ‘utility:’ to the UtilityData namespace module; and ‘cross:’ to the CrossSectionalData module).

6.3 Compact Schemas

Compact schemas express all attribute and dimension values as XML attributes. These may be placed at various levels within the imported SDMX ‘compact’ structure. The DSD-specific schema uses XSD substitution groups to attach DSD-specific elements and attributes to the structures provided in the ‘compact:’ namespace.

6.4 Cross-Sectional Schemas

DSD-specific cross-sectional schemas express non-time-series-based presentations of data dependent on the DSD. They also are capable of expressing statistical data for which time is not a concept. Thus, its the only SDMX-ML format for data which only of cross-sectional nature. Although, key values and attribute values are attached to the known four-level structure as XML attributes, the term ‘Series’ – indicating ‘time series’ – is replaced by the equivalent ‘Section’-level construct.

Named groups declared in the DSD are ignored for the purposes of the cross-sectional data format. They are replaced by a generic Group element, leaving it up to an application to enforce the validity of attribute values for groups of Sections.

A SDMX-ML cross-sectional schema may – according to the related DSD definitions - allow for more than one dimension to be expressed at the observation level. This replaces the role of time in time-series-oriented formats, and therefore it allows key values and attribute values to be attached at more than one level.

6.5 Utility Schemas

Utility schemas are different from the Compact and Cross-Sectional schemas because they distinguish between attributes and dimensions established in the respective DSD. This design enforces to preserve the ordering of the keys (key dimensions).

The utility schema is enriched with DSD structural metadata. This makes the rules inherent in the structure of the DSD available to such tools as schema-guided XML editors, which don't need to access the XML structure message describing the DSD. The extended validation capacity based on the enhanced content of structural metadata with the schema was the primary reason for the establishment of the Utility schema format.

The Utility schema employs a technique similar to the Compact and Cross-Sectional schemas by creating substitution groups which are headed by elements at the DataSet, Group, Series, and Observation levels.

The advantage that the utility messages can be more comprehensively validated with a generic XML parser is going along with the fact that this message format is considerably larger in size than the CompactData or CrossSectionalData ones.

6.6 Metadata Structure Definition-Specific Metadata Schemas

For all metadata-structure-specific schemas SDMX provides a namespace to be used as the extension base: SDMXMetadataReport.xsd. The metadata-structure-specific schema will be created in its own target name space, owned and maintained by the creating agency. It will use the targetNamespace attribute of the schema element to identify the namespace which contains the MSD-specific schema.

The SDMXMetadatReport.xsd namespace module provided by SDMX will be incorporated using the import element in the DSD-specific schema.

The SDMXCommon.xsd namespace module must also be imported into the schema.

Other xml:namespace attributes may be added to the schema element as needed.

Thus, the process is similar to the one explained for the DSD-specific schemas above.

7 Glossary

Table 4 presents the list of concepts and acronyms with their definition.

Concept	Definition
CS	Cross-Sectional
DSD	Data Structure Definition
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport
ID	Identifier
MSD	Metadata Structure Definition
SDMX	Statistical Data and Metadata eXchange
SDMX-EDI	SDMX Electronic Data Interchange - EDIFACT format for exchange of SDMX-structured data and metadata
SDMX-IM	SDMX Information Model
SDMX-ML	SDMX Mark-up Language - XML format for the exchange of SDMX-structured data and metadata
TS	Time Series
XML	EXtensible Mark-up Language
XSD	XML Schema Definition (file extension .xsd)
W3C	World Wide Web Consortium - the main international standards organization for the World Wide Web

Table 4 – Glossary