SDMX self-learning package No. 8

Student book

SDMX Architecture Using the Pull Method for Data Sharing

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

TABLE OF CONTENTS

1 Scope of the student book

Both student books No 7 and No 8 deal with the architectures for data sharing as defined by the SDMX (Statistical Data and Metadata eXchange) standard.

While student book No 7 provides a general overview of SDMX architectures using the pull method for data sharing, student book No 8 provides an in-depth description of the SDMX NSI Reference service infrastructure Eurostat has developed. The modules that constitute the architecture are explained in a step-by-step approach.

Ref.	Title
[01]	Introduction to SDMX
[02]	The SDMX Information Model
[03]	SDMX-ML Messages
[04]	Data Structure Definitions
[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 – Students books on SDMX

Prerequisites

Reading the previous student books is strongly recommended.

2 Introduction

This student book describes solutions on how to implement the pull method using SDMX. A data provider can make data available for the following purposes:

1. To produce an SDMX data file directly from a data file having another format.
2. To produce an SDMX data file using data stored in a local database.

Point 2 above has been chosen as a case study on how to implement the pull method using SDMX. The underlying infrastructure, its different components and the workflow will also be explained.

The main objective is to provide a description/specification for a general service infrastructure that can be re-used partially or as whole by organisations interested in starting SDMX projects.

3 Context and workflow

The architecture described in Figure 1 uses the open-source architecture developed by Eurostat (SDMX NSI Reference service infrastructure).

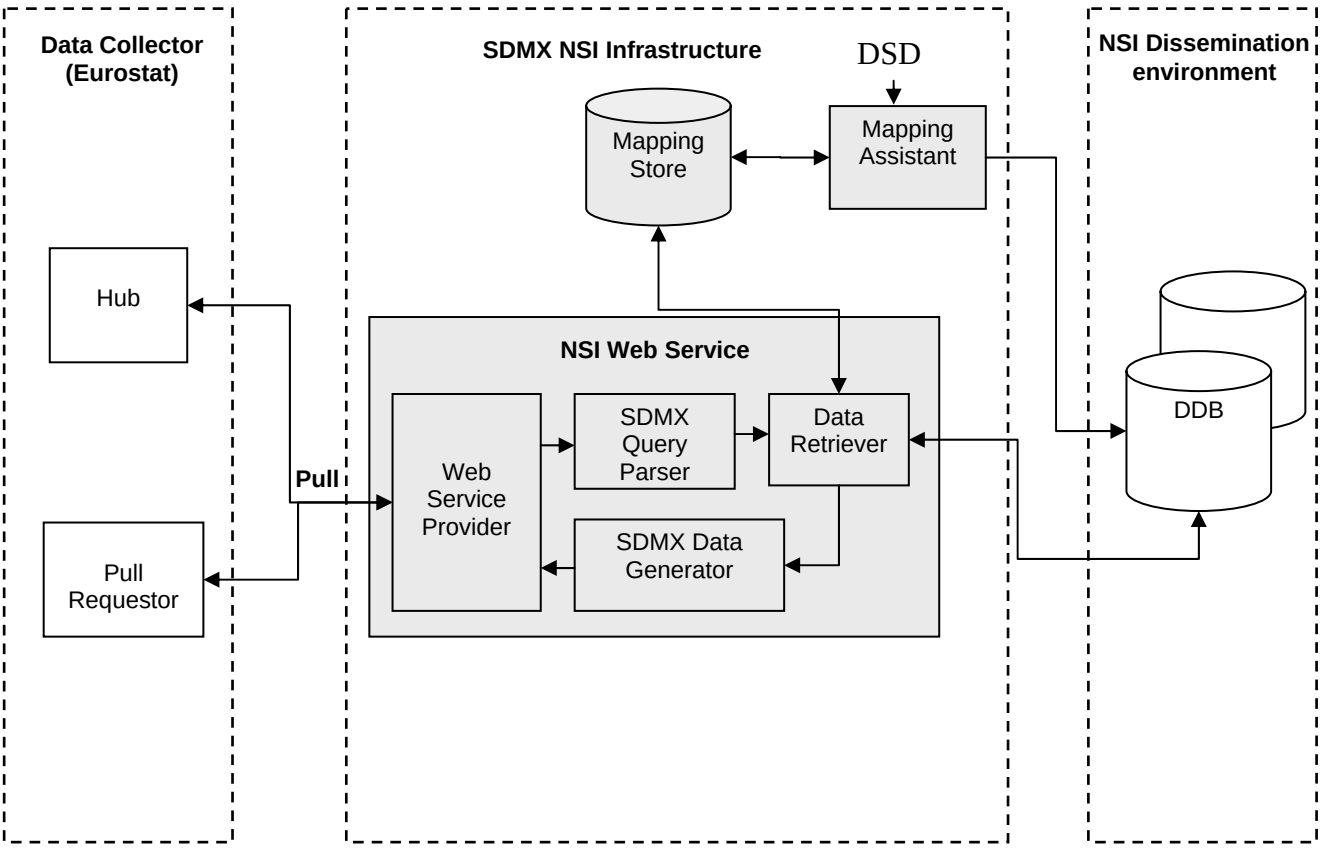

Figure 1 – A simplified view of the SDMX NSI Reference service infrastructure

The building blocks are:

- Web Service Provider: The web service interface, which has an SDMX Query Message as input and an SDMX Data File as response.
- SDMX Query Parser: The module that parses the SDMX Query Message (XML Document) into an internal SDMX Data Model.
- Data Retriever: Ensures access to the dissemination database and retrieves the data related to the SDMX Query Message.
- SDMX Data Generator: Transforms data stored in the internal SDMX Data model into an SDMX Data File (XML Document).

In addition, the SDMX Model Library, as a class representation of the SDMX Information Model, is used for storing/handling SDMX messages within the building blocks of the SDMX NSI Reference service infrastructure. It is a reusable component, which simplifies the interfaces for the SDMX Reference service infrastructure modules, and represents all SDMX-related information in Java/.NET objects.

The workflow begins with a dissemination database and ends with making data available by using a Web Service. In the following chapters, a complete description is provided on how to use/reuse the different modules.

3.1 Sequence Data Process

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner (see Figure 2).

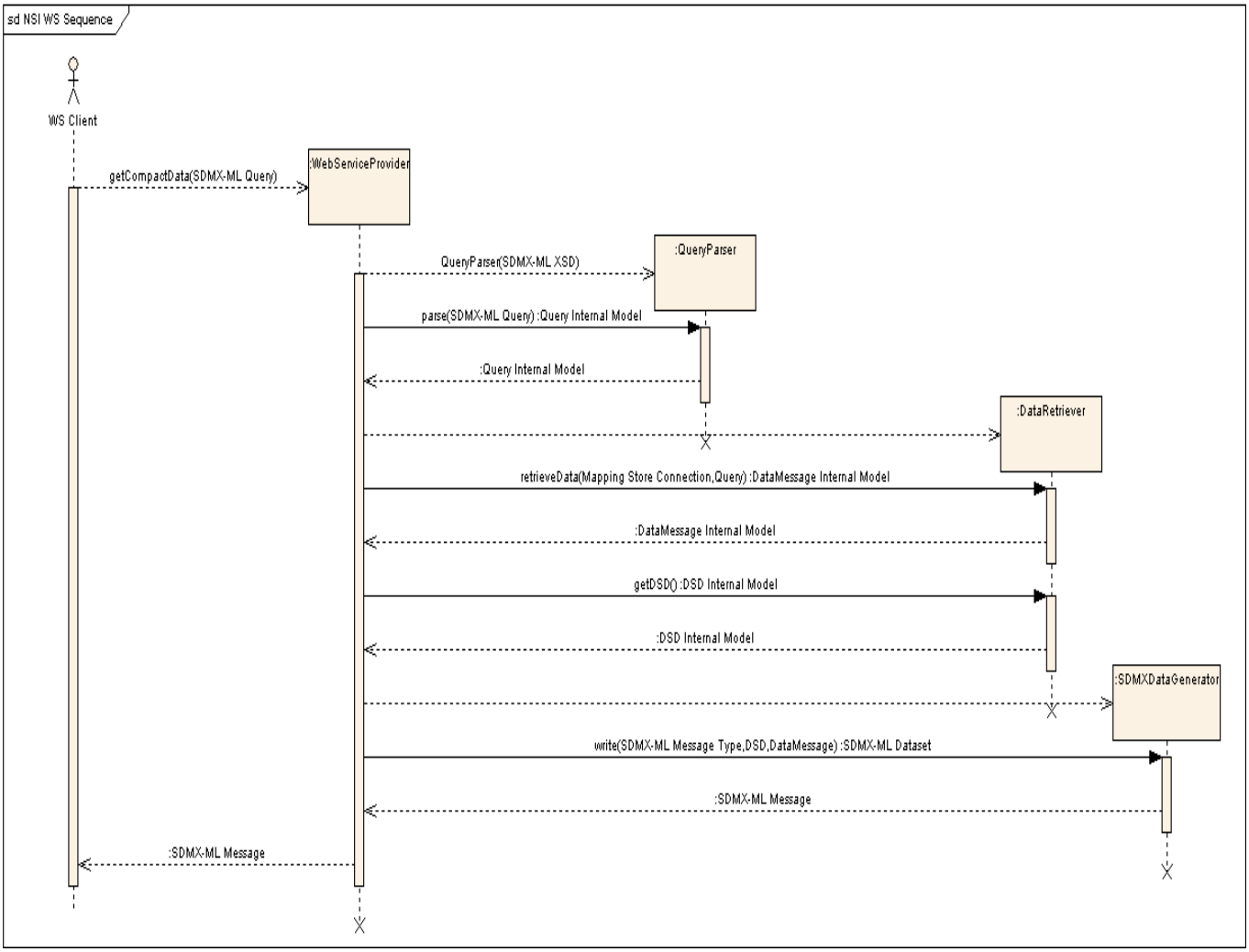

Figure 2 – Sequence diagram ('getCompactData' method)

The sequence begins with a call to the web service with an SDMX-ML Query Message requesting Compact Data via the web service method 'getCompactData'.

This SDMX-ML Query Message is sent to the Query Parser Module, where it is parsed and converted into an SDMX Query Model inside the SDMX Data Model of the application.

The Internal SDMX Data Model (SDMX Query Model) and the mapping store connection data (mapping rules for the dissemination database between local databases and the DSD) are used as input for the Data Retriever. Subsequently, the Internal SDMX Data Model receives an SDMX Data Message that will also be stored inside the SDMX Data Model of the application. For this purpose, the DSD of the domain used in the SDMX-IM Query Message will be required to access the dissemination database.

Finally the information stored in the Internal SDMX Data Model will be translated into an SDMX-ML Data File inside the Data Generator in the format required by the method requested by the client in the web service (in this case Compact Format). This translation will be done taking the DSD as an input, corresponding to the input SDMX-IM Query Message. This SDMX-ML Data File is the response sent to the client.

4 Possible database architectures

Physical data architecture encompasses database architecture. Database architecture is a schema of the actual database technology that will support the designed data architecture.

As outlined in the introduction, the workflow starts with a dissemination database (DD) and ends with the data being made available via a Web Service. The dissemination database is the storage data warehouse (or database) in the organisation's dissemination environment that each organisation maintains in order to store data ready for publication/dissemination to potential Data Consumers. In some cases, the DD may consist of files, i.e. PC-Axis files.

There are different ways in which data can be stored on the data provider's premises:

- Data are not stored in any database yet. This means a new database may be created which needs to be 'SDMX-compliant' (database having the structure of the artefacts of the SDMX information model: code lists, concepts, descriptors, primary measure, time dimension, etc.).
- Data are already stored in a 'local database'. Data are described using local structural metadata. This means that the other structure tables are necessary in order to carry out the mapping between the SDMX information model structure and the local database structure.

Storing data in a database has the advantage of making these data available using a web service.

4.1 The database schema

The schema of a database system is its structure described in a formal language supported by the database management system. This paragraph presents the situation of 'SDMX-compliant' and 'non-SDMX-compliant' databases. The examples are taken from the Istat SDMX framework. However, it must be stressed that the schemas can be designed and produced in different ways.

4.1.1 Database based on SDMX-IM

In the case of an 'SDMX-compliant' database, two distinct conceptual areas can be delimited in the database structure.

- Metadata Structure tables area: Tables containing the structural metadata for every DSD loaded (code lists, concepts, descriptors, key_descriptors, etc.)
- Data tables area: Tables containing data described by the DSD concepts.

4.1.2 Database not based on SDMX-IM

The structure of the database can be described schematically. It is composed of the following conceptual areas:

- *Local Metadata Structure tables area*: Tables containing local structural metadata.

- *Mapping tables area:* Tables containing the mapping done for all the concepts and values, between the SDMX information model structures and the local database structure.
- *Data tables area*: Tables containing data, described by the local concepts.
- *Metadata Structure tables area*: Tables containing DSD structural metadata (as in the previous case).

4.2 The database architecture

4.2.1 SDMX-compliant database (based on SDMX-IM) [1]

The production 'flow' of SDMX data files can be described as follows:

- When new data are loaded, the associated Data Structure Definition must be stored in the database (in the Metadata Structure Table area).
- Data can be selected from the database using the Web Service with the required parameters (dataflow, range of data, etc.).
- The database containing these parameters is accessed in order to create the data file.

4.2.2 Non-SDMX-compliant database (not based on SDMX-IM) [2]

For this type of database architecture the production 'flow' to SDMX data files can be described as follows (Figure 3):

- When new data are loaded, the associated Data Structure Definition (DSD) must be stored in the database (in the Metadata Structure Tables area).
- It is necessary to map the new DSD with local concepts and store the information in the database.
- Data can be selected from the database using the Web Service and required parameters (dataflow, range of data, etc.).
- The database containing these parameters is accessed in order to create the data file.

[1] An example of an SDMX-Compliant Database can be found in the annex chapter: 6.1 SDMX-Compliant Database.

[2] An example of a non-SDMX-Compliant Database can be found in the annex chapter: 6.2 Non-SDMX-Compliant Database.

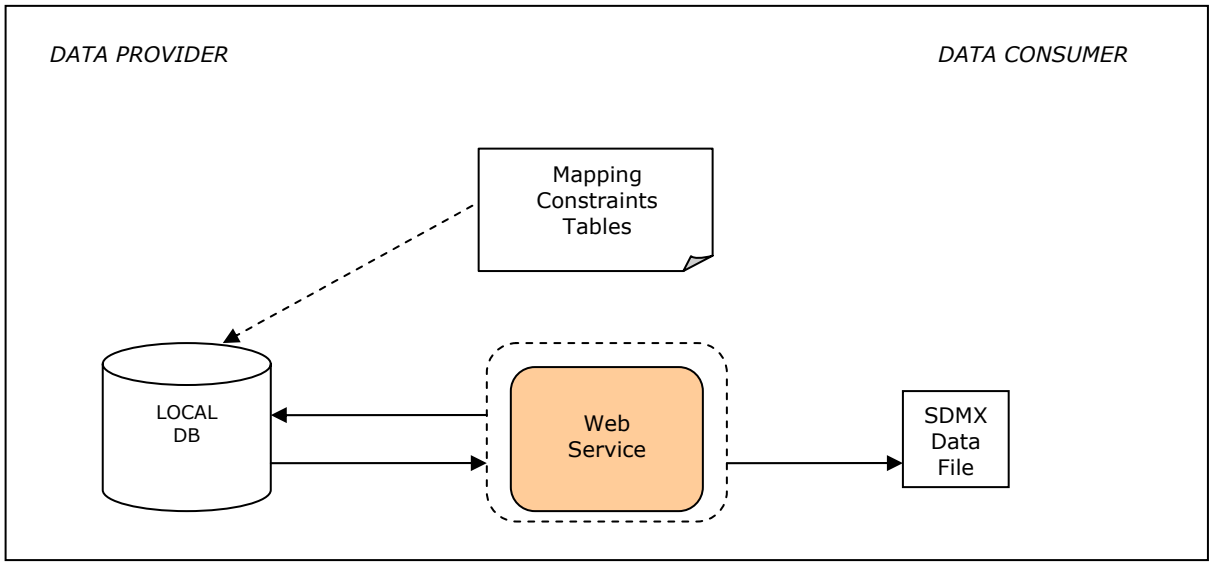

Figure 3 – From a non-SDMX-compliant database to an SDMX data file

5 SDMX NSI Reference service infrastructure experiences

An organisation can decide to use the SDMX service infrastructure depicted in Figure 4 as a whole, can extend the infrastructure adding new modules, can modify some modules, or can integrate some building blocks within its existing dissemination environment.

The architecture exists both in Java and .NET implementations and the modules/components are released as Open-Source Software.

The example below is based on experiences gathered from an NSI using the SDMX infrastructure for data sharing and interoperability. References are made to .NET technology, so a certain understanding/knowledge of .NET is required.

The roles of the components/modules are documented below:

- Web Service Provider;
- SDMX Query Parser;
- Data Retriever;
- SDMX Data Generator.

The ‘SDMX Data Model’ in this architecture will also be described as it is a representation of the SDMX-ML structures and it is used internally in the exchange of information between different modules.

This architecture's main interface is a Web Service in which SDMX Queries are sent as parameter and SDMX Data Files are returned in response. Three methods are available to the data consumer in this web service:

- ‘GetCompact’ (to get SDMX Data File messages in Compact format)
- ‘GetCrossSectional’ (to get SDMX Data File messages in Cross-Sectional Format)
- ‘GetGeneric’ (to get SDMX Data file messages in Generic Format)

As outlined in the SDMX guidelines for the use of web services ‘SDMX-ML, as a standard XML for exchanging data and structural metadata within the statistical realm, provides a useful XML format for the public serialization of web-services data’ and ‘the use of a set of XML exchanges based on a common information model is seen as a better approach for achieving interoperability’.

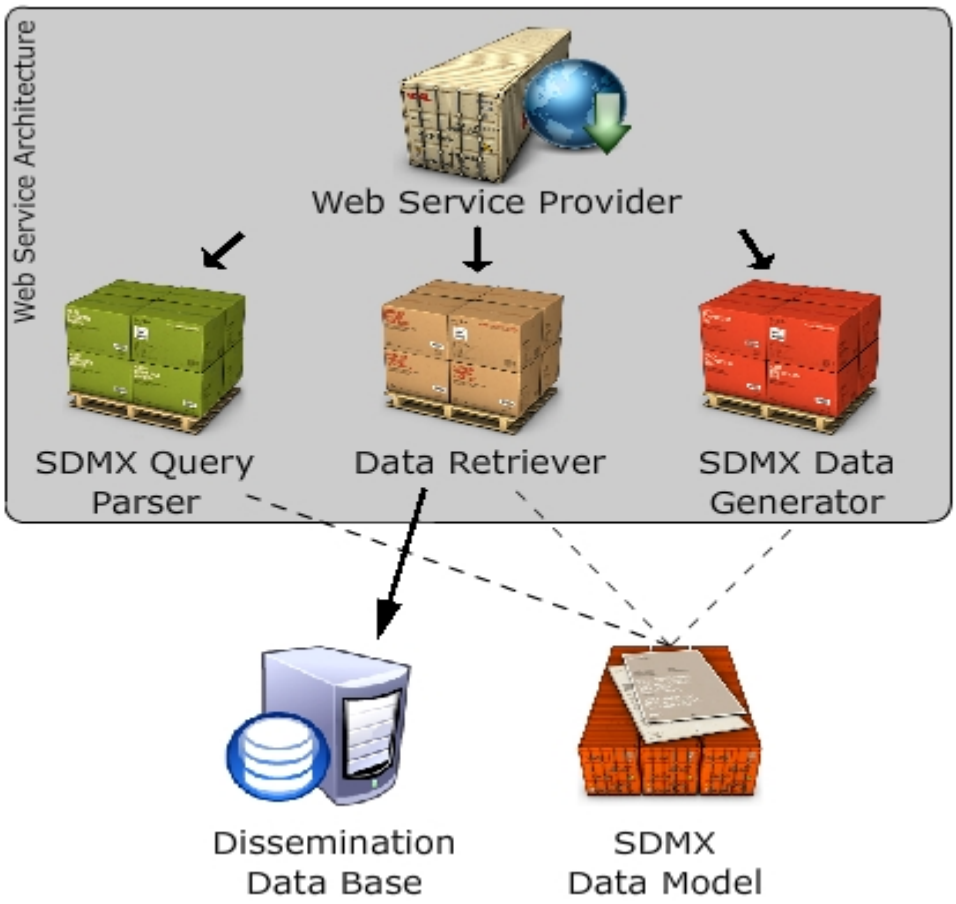

Figure 4 – SDMX NSI Reference service infrastructure

5.1 The Web Service Provider

The main objectives of the ‘Web Service Provider’ are to receive an SDMX Query Message, control the exchange of information between the other modules, and respond with an SDMX data file message according to the SDMX Query taken as input.

The web service is provided with an ‘XML Validation’ functionality, which enables the validation of the SDMX query received through it, and with a ‘SOAP Error Handler’, which checks for

errors that can occur during the process and sends either a positive response (SDMX Data File) or negative response (XML Error) to the data consumer.

The following UML Deployment Diagram in Figure 5 depicts the Web Service Provider working as an API. This component presents the underlying data-retrieval functionality using a SOAP interface. More specifically, its input is a SOAP message, which includes an SDMX-ML Query message, and its output is a SOAP message including either an SDMX-ML data file message or a SOAP Fault in case of an error. In order to ensure complete distribution, this module requires the introduction of configuration files as parameters. It is also responsible for passing these files to the other building blocks.

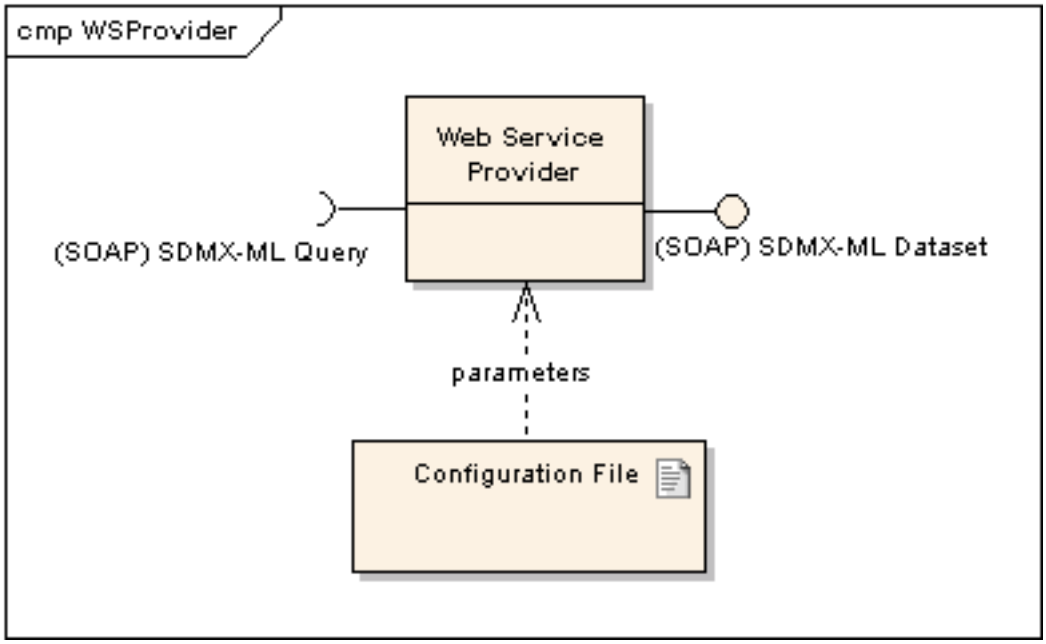

Figure 5 – Web Service Provider

'Step 1' The sequence begins with a call to the web service with an SDMX-ML Query Message asking for Compact Data using the web service method 'getCompactData'.

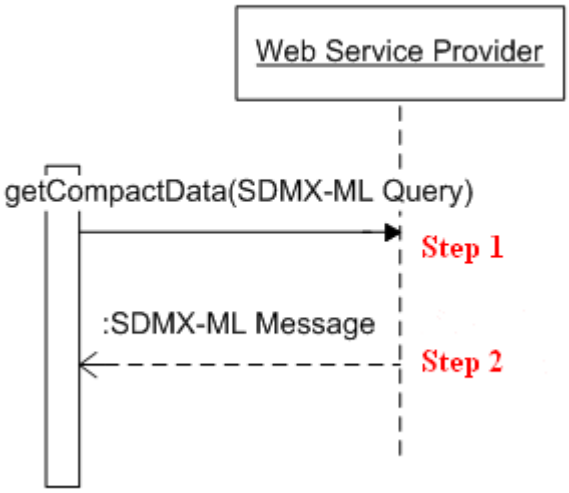

Figure 6 – Web Service Provider sequence diagram ('getCompactData' method)

'Step 2' The operation of the Web Service Provider and the other modules of the SDMX NSI Reference service infrastructure are hidden to the final user. The user receives an SDMX-ML Message in compact format in response to his query.

5.2 Reusing the Web Service Provider

The Web Service Provider module is a building block which can be reused within any dissemination environment requiring its functionality.

In case it is reused as a package, the internal SDMX Data Model needs to be attached to the API since the parsed element created in the Query Parser is an SDMX Query instance defined inside the structure of the internal SDMX Data Model. This instance is used to call the Data Retriever. An SDMX-ML data file instance defined inside the internal SDMX Data Model generated by the Data Retriever is also used to get the output of the Data Generator as SDMX-ML File in XML format.

This module is the least independent module since it is in fact working as a controller and caller[3] for the rest of the modules and for the information exchanged between them using the internal SDMX Data Model.

The 'Web Service Provider' component is released as open-source software (OSS). It can thus either be reused as is (as a package) or by making use of its source code (in case modifications are needed).

[3] The Web service provider "calls" the other modules - Query Parser, Data Retriever & Data Generator - if the entire framework is applied. Alternatively, they can also be called by any kind of software developed to reuse one or more of these API modules.

Input

1. **SOAP (SDMX Query)**: Soap message including the SDMX Query.

Output

2. **SOAP (SDMX-ML Dataset)**: Soap message including the SDMX-ML Dataset.

5.3 The API SDMX Query Parser

The 'Query Parser' module is a simple XML Parser API implementation specifically foreseen for SDMX-ML Query messages. The Query Parser's main objective is to store an SDMX-ML Query message in an in-memory SDMX Data Model, making it available for processing by other modules within a development environment.

The UML Deployment diagram in Figure 7 shows that the inputs into the SDMX Query Parser are SDMX-ML Query messages (XML documents) and SDMX Query xml schemas (SDMX-ML XSDs), as defined at the following address: www.sdmx.org, http://sdmx.org/docs/2_0/SDMXQuery.xsd. The output is a Query represented in the internal SDMX Data Model.

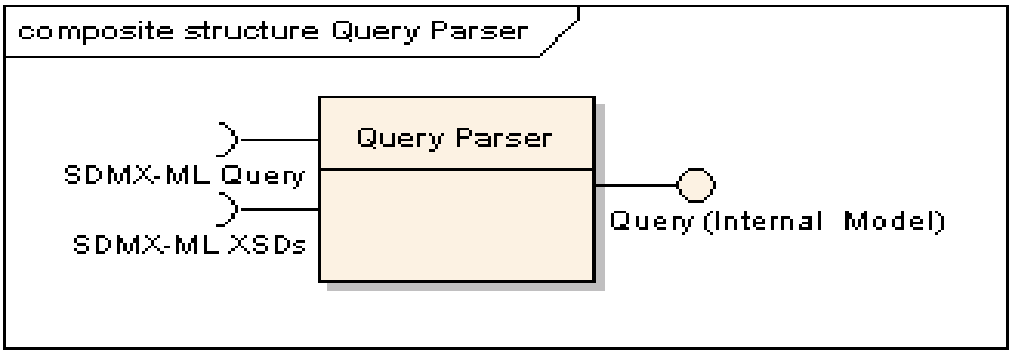

Figure 7 – Query Parser

'Step 1' The Web Service Provider (as 'caller') initiates the creation of a Query Parser instance using the 'QueryParser' method with SDMX-ML Schemas as parameters.

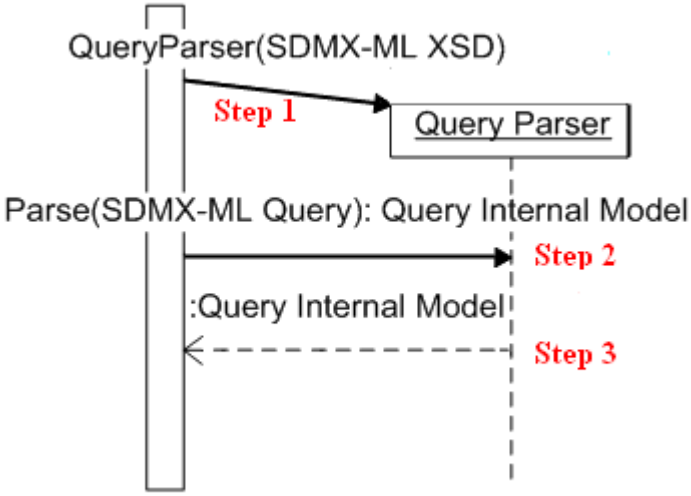

Figure 8 – Query Parser sequence diagram ('getCompactData' method)

'Step 2' An SDMX-ML Query is sent to the Query Parser calling the method 'Parse'. The SDMX-ML Query is stored in the 'Internal Model' and the reference to the translated query is passed to the caller.

'Step 3' The output of the Query Parser is the 'Internal Model' of the Query. The entire operation of the Query Parser is hidden for the user.

5.3.1 Reusing the Query Parser

The Query Parser is also released as open-source software (OSS). This means it can be used as an API module or its source code can be used in case any modification is needed. When using the Query Parser as an API module, in order to obtain a compliant output of the API, it is recommended to also use the SDMX Data Model.

The Query Parser has to validate the SDMX query by using the schemas proposed by the SDMX standard and then translating it into a structure that is easier to use inside the architecture.

Input

1. **SDMX-ML Query**: String data to be serialised that represents the SDMX Query sent by the data consumer inside the SOAP message obtained by the Web Service.
2. **SDMX**-ML XSDs: SDMX Schemas used to validate the SDMX Query.

Output

The output is an SDMX Query Message represented by the Internal SDMX Data Model.

Example

As an example, a SHORT-TERM STATISTICS SDMX Query is presented in Figure 9:

```
<query:DataWhere>
      <query:And>
        <query:Dimension id="FREQ">M</query:Dimension>
        <query:Dimension id="ADJUSTMENT">W</query:Dimension>
       <query:Time>
          <query:StartTime>2006-01</query:StartTime>
          <query:EndTime>2006-03</query:EndTime>
        </query:Time>
        <query:DataProvider>IT1</query:DataProvider>
        <query:Dataflow>SSTSIND_PROD_M</query:Dataflow>
       <query:Or>
          <query:Dimension id="STS_ACTIVITY">N100DA</query:Dimension>
          <query:Dimension id="STS_ACTIVITY">N100DB</query:Dimension>
        </query:Or>
      </query:And>
</query:DataWhere>
```

Figure 9 – Example of SDMX query for STS

This query requests the SSTS_IND_PROD dataflow for the period between 2006-01 and 2006-03, for activities 'N100DA' and 'N100DB', with a frequency 'M' and adjustment 'W', by data provider Italy (IT1). The output of the QueryParser is a 'QueryBean' object containing the following information:[4]

Class element	Name of property	Value of property	Parent class element
OperatorBean(0)	OperatorType	And	QueryBean
TimeBean	StartTime	2006-01	OperatorBean(0)
	EndTime	2006-03	OperatorBean(0)
DataProviderBean	Value	IT1	OperatorBean(0)
DataFlowBean	Value	SSTSIND_PROD_M	OperatorBean(0)
Dimension(0)	Id	FREQ	OperatorBean(0)
	Value	M	
Dimension(1)	Id	ADJUSTMENT	OperatorBean(0)
	Value	W	
OperatorBean(1)	OperatorType	Or	OperatorBean(0)
Dimension(0)	Id	STS_ACTIVITY	OperatorBean(1)
	Value	N100DA	
Dimension(1)	Id	STS_ACTIVITY	OperatorBean(1)
	Value	N100DB	

Table 2 – Content of the 'QueryBean' object

[4] The architecture exists both in Java and .Net implementations. However, this example illustrates the .Net architecture.

Table 2 should be read in the following way:

- Column 1 represents the name of the classes that are inside the QueryBean object.
- Column 2 represents the name of the properties belonging to the classes indicated in column1.
- Column 3 represents the value of the properties indicated in column2.
- Column 4 represents the parent classes to which the classes indicated in column1 belong.

5.4 The Data Retriever

This module offers the possibility of retrieving data from dissemination databases given a parsed SDMX-ML Query stored in the SDMX Data Model. The query translation to native SQL for the target dissemination database is assisted by mappings already created on the data provider side that can be stored in another local database (Mapping Store). Its main responsibility is to translate somehow the incoming SDMX-ML Query into an SQL Query that will be used for retrieving data from the Dissemination Database. The retrieved data are returned as an SDMX-ML Dataset represented in the SDMX Data Model.

It is considered as the most important and complex class of the Web Service architecture.

Using a UML Development Diagram (Figure 10), access to the Dissemination Database is required to retrieve data and access the Mapping Store database, so as to retrieve the mappings and the connection string to the ‘Dissemination Database’ used according to the domain needed. The connection string to the ‘Mapping Store’ database is provided to the ‘Data Retriever’ as an input parameter. Another input of the ‘Data Retriever’ is the SDMX-ML Query represented in the internal SDMX Data Model. The output of the ‘Data Retriever’ is the SDMX-ML Dataset represented in the internal SDMX Data Model.

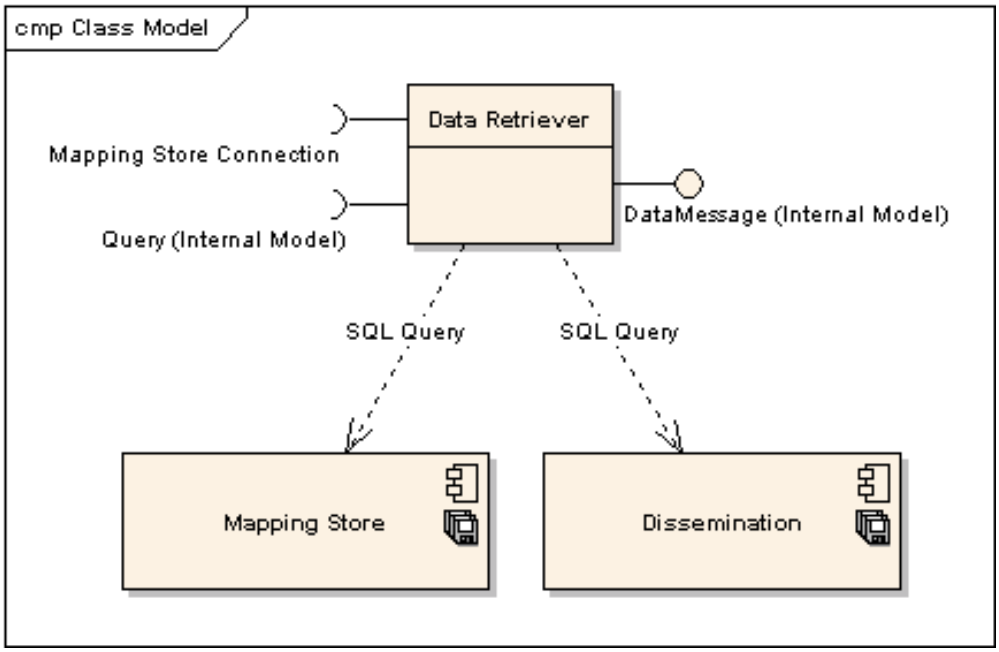

Figure 10 – Data Retriever

'Step 1' The sequence begins with a Data Retriever instance creation by the Web Service Provider (as 'caller') using the 'DataRetriever' method without parameters.

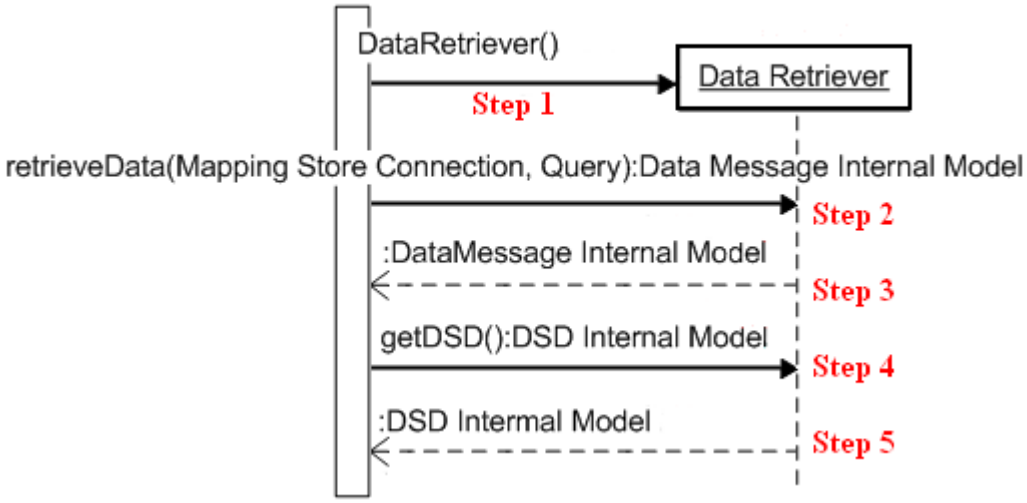

Figure 11 – Data Retriever sequence diagram

'Step 2' The caller passes the query in the 'Internal Model' format to the Data Retriever calling the method 'retrieveData' and passing also the mapping connection as parameter.

'Step 3' The DataMessage is stored in the Internal Model and the reference to this DataMessage is passed back to the caller.

'Step 4' The function 'getDSD' is used by the caller to get the DSD in the 'Internal Model' format.

'Step 5' The DSD is stored in the 'Internal Model' format.

5.4.1 Reusing the Data Retriever

This module is also released as open-source software (OOS) and can be reused as a complete package or the code inside the module can be reused.

It uses two database connections: one to the database in which the mappings are stored, and another to the dissemination database. In case a different design of database connections is required, the source code of the Data Retriever must be modified.

The Data Retriever is the only API dependent on the database. It creates the SQL query and retrieves not only the data, but also some structural metadata information necessary to create the SDMX file.

Input

1. **Internal SDMX Query Message**: Result of the Query Parser API. A representation of the SDMX-IM Query Message inside the Internal SDMX Data Model.
2. **Mapping Store Connection**: String needed to connect to the Mapping Store Data Base, where all the information related to the mappings between the DSD of the Query and the Local Dissemination Database is stored.

Output

The output is an internal representation of an SDMX Data Message inside the internal SDMX Data Model.

Example (SDMX query for STS in Figure 9 and content of the 'QueryBean' object in Table2

The QueryBean object from the QueryParser is used as input for the DataRetriever. The DataRetriever processes the QueryBean object following the mapping rules created to translate the DSD concepts and codes to local database concepts and code. The main output is a 'DataMessage' object containing a 'DataSet' with data retrieved from the local database and already mapped with the DSD concepts. The content is displayed in Table 3[5] below:

[5] The architecture exists both in Java and .Net implementations. However, this example illustrates the .Net architecture.

Class element	Name of property	Parameter of property	Value of property	Parent class element
Group(0)	Id		Sibling	DataSet
	KeyValues()	REF_AREA	IT	
		ADJUSTMENT	W	
		STS_INDICATOR	PROD	
		STS_ACTIVITY	N100DA	
		STS_INSTITUTION	1	
		STS_BASE_YEAR	2000	
Group(1)	Id		Sibling	DataSet
	KeyValues()	REF_AREA	IT	
		ADJUSTMENT	W	
		STS_INDICATOR	PROD	
		STS_ACTIVITY	N100DB	
		STS_INSTITUTION	1	
		STS_BASE_YEAR	2000	
Series(0)	KeyValues()	REF_AREA	IT	DataSet
		ADJUSTMENT	W	
		STS_INDICATOR	PROD	
		STS_ACTIVITY	N100DA	
		STS_INSTITUTION	1	
		STS_BASE_YEAR	2000	
		FREQ	M	
	AttributeValues()	TIME_FORMAT	P1M	
Series(1)	KeyValues()	REF_AREA	IT	DataSet
		ADJUSTMENT	W	
		STS_INDICATOR	PROD	
		STS_ACTIVITY	N100DB	
		STS_INSTITUTION	1	
		STS_BASE_YEAR	2000	
		FREQ	M	
	AttributeValues()	TIME_FORMAT	P1M	
Observation(0)	AttributeValues()	OBS_STATUS	A	Series(0)
	TimeValue		200601	
	Value		101.1	
Observation(1)	AttributeValues()	OBS_STATUS	A	Series(0)
	TimeValue		200602	
	Value		99.5	
Observation(2)	AttributeValues()	OBS_STATUS	A	Series(0)
	TimeValue		200603	
	Value		105.1	
Observation(0)	AttributeValues()	OBS_STATUS	A	Series(1)
	TimeValue		200601	
	Value		77	
Observation(1)	AttributeValues()	OBS_STATUS	A	Series(1)
	TimeValue		200602	
	Value		81.3	
Observation(2)	AttributeValues()	OBS_STATUS	A	Series(1)
	TimeValue		200603	
	Value		84.2	

Table 3 – Content of the 'DataSet' object

Table 3 should be read in the following way:

- Column 1 represents the name of classes that are inside the DataSet object.
- Column 2 represents the name of the properties belonging to the classes indicated in column1.
- Column 3 represents the parameters for the properties.
- Column 4 represents the values of the properties indicated in column2.
- Column 5 represents the parent classes to which the classes indicated in column1 belong.

5.5 The API SDMX Data Generator

This module is responsible for creating SDMX-ML messages given as an input into an SDMX Data Model used for storing SDMX data and metadata. This module concerns the translation of the internal data model in order to build a valid SDMX-ML Dataset.

Within the web service process as a whole, this SDMX-ML Dataset would comply with the incoming SDMX-ML Query which the Web Service Provider took as input.

The UML Development diagram in Figure 12 shows that the input of the 'SDMX-ML Data Generator' is a dataset (DataMessage) represented in the internal SDMX Data Model. Additionally, some metadata are required in order to produce the dataset properly: the Data Structure Definition (DSD) and the SDMX-ML Message Type. The latter provides the format in which the dataset should be written and must be provided as an input. The output is an SDMX-ML Dataset (XML document).

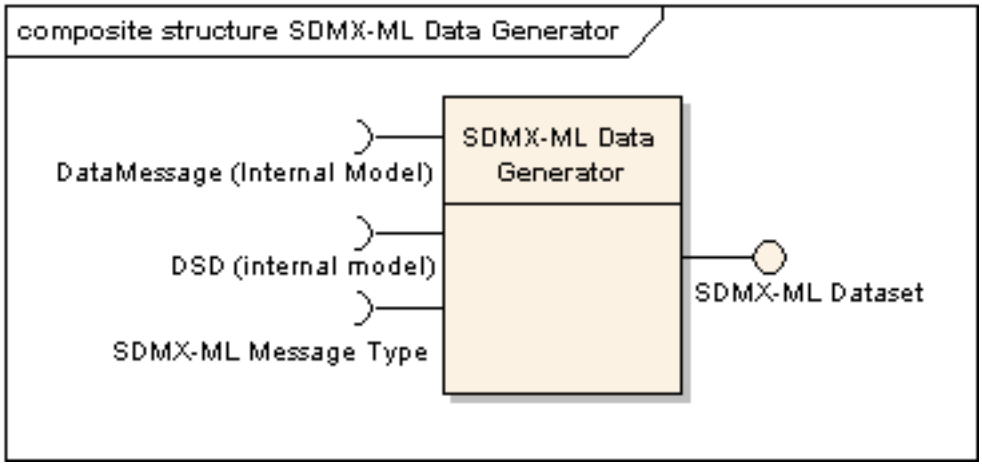

Figure 12 – SDMX-ML Data Generator

'Step 1' The Web Service Provider (as 'caller') initiates a Data Generator instance using the 'CompactDataGenerator' method without parameters.

This sequence is described for a resulting SDMX-ML dataset in 'Compact Data Format'.

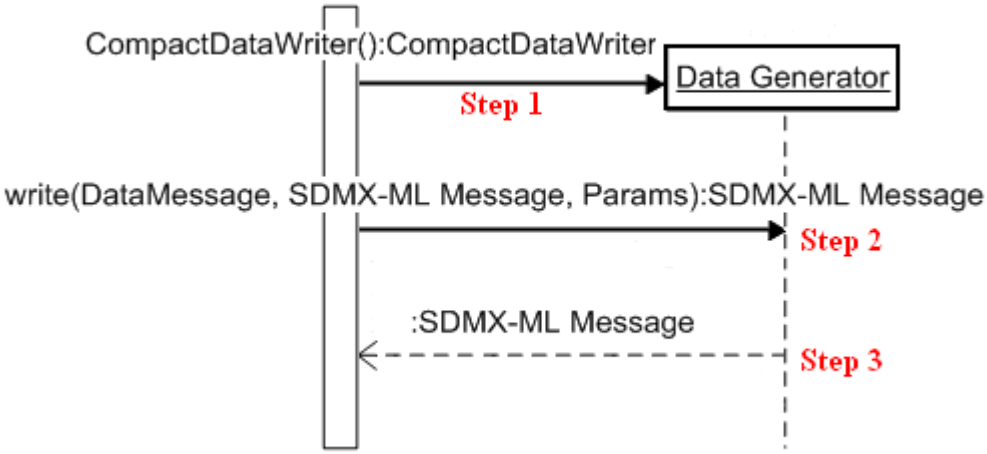

Figure 13 – Data Generator sequence diagram (Compact Data Format)

'Step 2' One DataMessage in Internal Model format is sent to the Data Generator calling the 'write' method, the SDMX-ML Message (XML file where to store the result) and the DSD inside a parameter variable (Params) as parameter.

'Step 3' The SDMX-ML Message is updated and the reference is passed to the caller.

5.5.1 Reusing the Data Generator

As with the previous modules, this one can also be reused as a package or its code can be reused. If it is used as API, it is recommended to use the SDMX Data Model as this is the API's most important input parameter.

The output generated is an XML File.

The SDMX Data Generator is the last API in the process. It is here that the SDMX data file is created using the data retrieved from the dissemination database.

Input

The parameters are:

1. **DataMessage**: Data Message created as output in the previous module and represented inside the Internal SDMX Data Model;
2. **DSD**: DSD related to the SDMX-ML Query Message used to create the corresponding SDMX-ML Data Message;
3. **Target FORMAT**: SDMX-ML message format of the SDMX dataset.

Output

The output is a string data type with the SDMX-ML data file to be serialized as response inside a SOAP message.

The APIs described can be used as a complete solution, or its separate components as APIs, depending on the user's needs.

5.6 SDMX Data Model

SDMX Data Model Library, as a class representation of the SDMX Information Model, is used to represent information internally for the entire SDMX NSI Reference service infrastructure. The SDMX Data Model serves as structure for information stored in the SDMX Queries Messages as well as to produce the SDMX Data Message and its metadata between the building blocks of the SDMX Reference service infrastructure.

It is a reusable component, which simplifies the interfaces for the SDMX Reference service infrastructure modules with readers and writers and represents all SDMX-related information in Java/.NET objects.

6 Annex 1 – Database schemes

The examples in the annex are taken from the Istat SDMX framework. However, it must be stressed that the schemas can be designed and produced in different ways.

6.1 SDMX-Compliant Database

The schema in Figure 14 describes the parts of the database designed for the domains Demography and Census-Hub to store one 'SDMX Compliant Database'. In the case of the Census-Hub, the constraint table and data table can be merged because there are no attributes attached at observation level and no time measure, but only the dimension 'measure'.

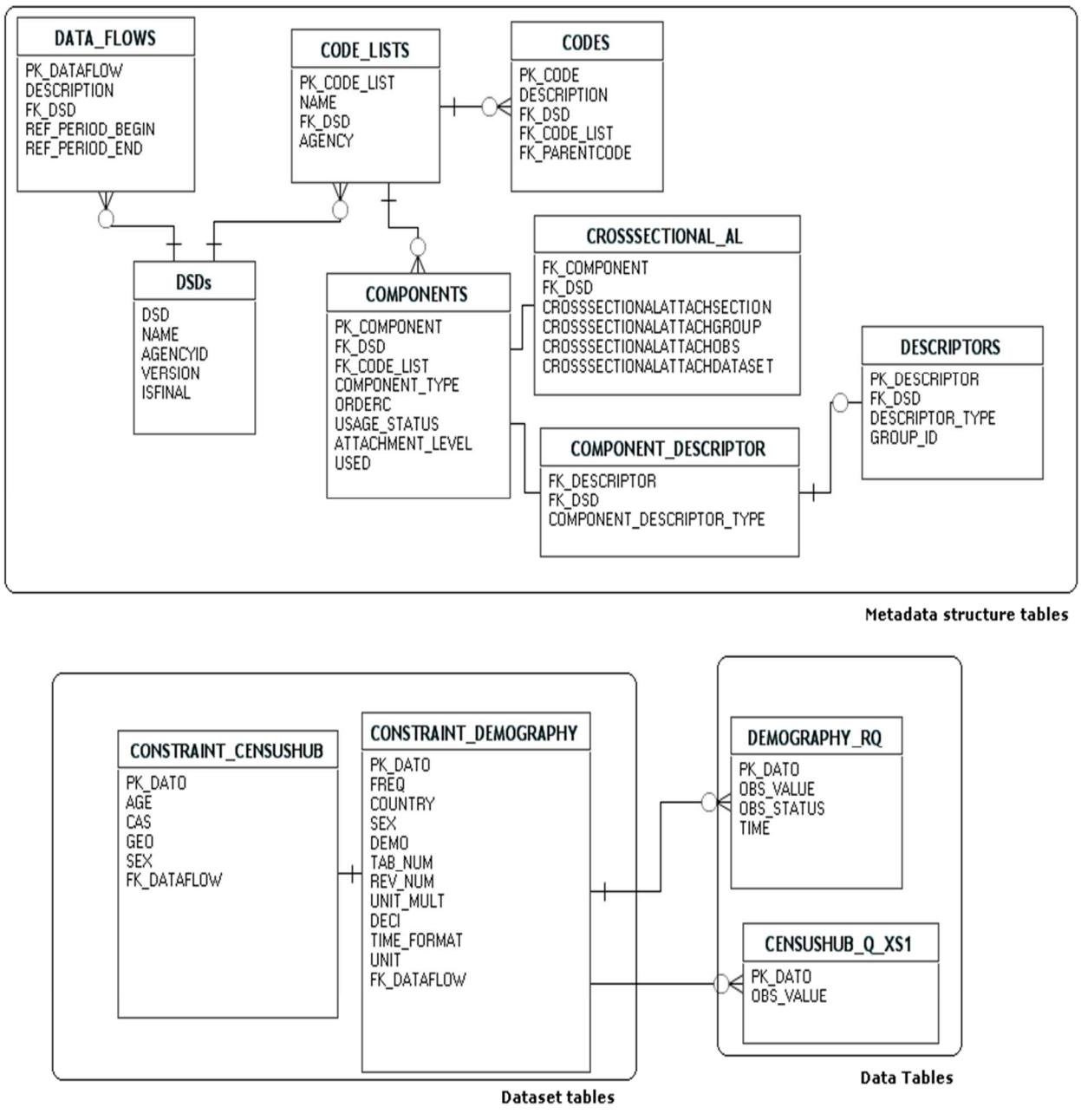

Figure 14 – SDMX-compliant database schema (Demography and Census-Hub)

The design for both domains is separated into 'Metadata structure tables', 'Constraints Mapping Tables' and 'Data tables'.

- Metadata structure tables' area: Tables containing the metadata structure concerning the SDMX-ML Standard.
- Constraints tables area: Tables containing the structure of each DSD. The columns are the concepts belonging to the Dataset, Group and Series or Sections level. It is necessary to create a table for each DSD (because each DSD has different kinds of concepts).
- Data tables area: Tables containing the data. The columns in these tables are the concepts belonging to the observation level and to the primary measure.

6.2 Non-SDMX-Compliant Database

Figure 15 and Figure 16 depict the schema of the local database for Short-Term Statistics. All the tables are ones that already existed in the previous database architecture.

The '*Mapping Constraint Table'* area contains tables of the mapping between the constraints related to the structure metadata and the structure of the local database.

The table 'STS_MAPPING' contains the primary key (without time) of the data table (DOM, S_DOM, CATEGORY, TYPE, VS, and ACTIVITY), the name of the table (TABLE_NAME), the name of the column (COLUMN_NAME) and the name of the database (DATABASE_NAME).

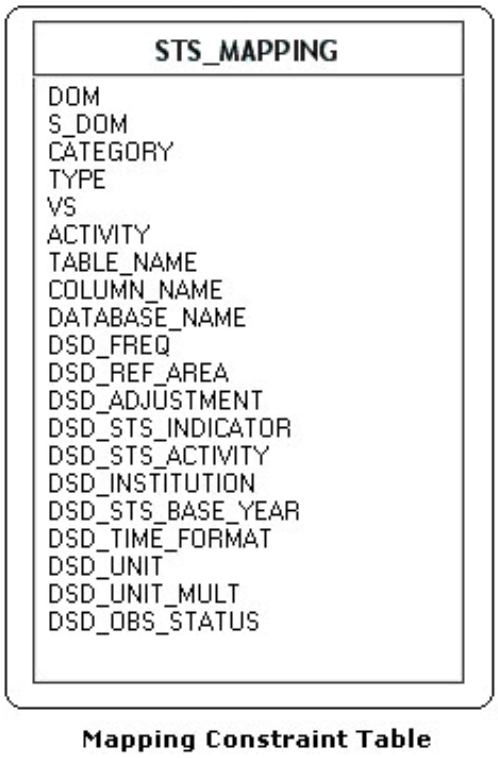

Figure 15 – Database schema (non-SDMX-compliant) (Short-Term Statistics)

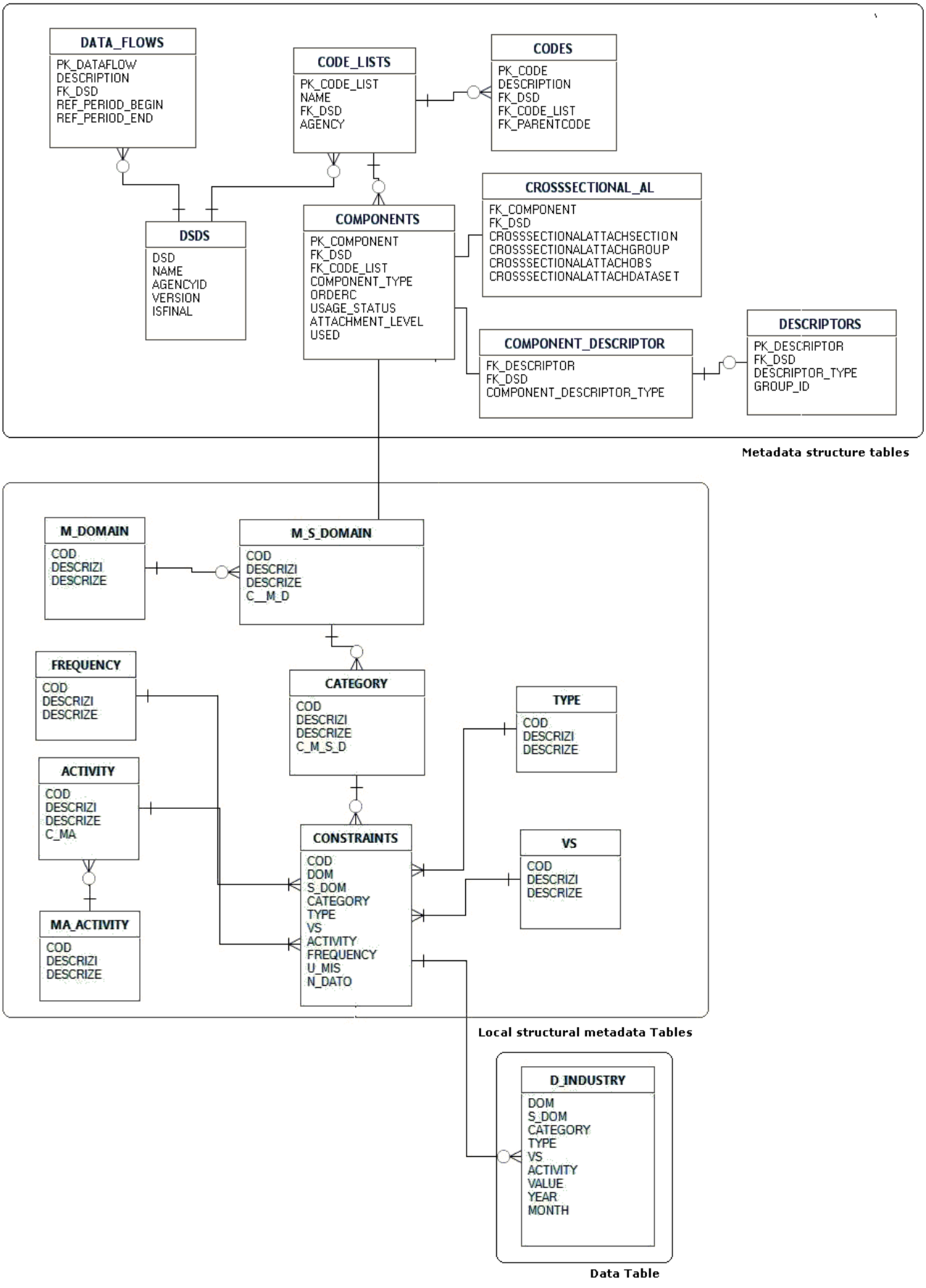

Figure 16 – Database schema (Non-SDMX-compliant) (Short-Term Statistics)

7 Glossary

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

CONCEPT	DEFINITION
CVS	COMMA-SEPARATED VALUES
DSD	DATA STRUCTURE DEFINITION
EDIFACT	ELECTRONIC DATA INTERCHANGE FOR ADMINISTRATION, COMMERCE AND TRANSPORT
ESMS	EURO SDMX METADATA STRUCTURE
GESMES/TS	GESMES TIME SERIES DATA EXCHANGE MESSAGE
HTTP	HYPERTEXT TRANSFER PROTOCOL
ISO	INTERNATIONAL ORGANISATION FOR STANDARDISATION
ISTAT	NATIONAL STATISTICAL INSTITUTE OF ITALY
IT	INFORMATION TECHNOLOGY
MSD	METADATA STRUCTURE DEFINITION
NSI	NATIONAL STATISTICAL INSTITUTION
RSS	REALLY SIMPLE SYNDICATION (*ALSO USED RICH SITE SUMMARY*) - FAMILY OF WEB FEED FORMATS TO PUBLISH FREQUENTLY UPDATED INFORMATION
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 MARKUP LANGUAGE - XML FORMAT FOR THE EXCHANGE OF SDMX-STRUCTURED DATA AND METADATA
SOAP	SIMPLE OBJECT ACCESS PROTOCOL
SQL	SEQUENCE QUERY LANGUAGE
UML	UNIFIED MODELLING LANGUAGE
URL	UNIFORM RESOURCE LOCATOR
XLS	EXCEL WORKSHEET
XML	EXTENSIBLE MARKUP LANGUAGE

Table 4 - Glossary