SDMX 2.1 User Guide

Version 0.1 - 19/09/2012

Objective of this new draft of the User Guide

This new SDMX User Guide aims at providing explanations and guidance to users (and potential users) of the version 2.1 of the Technical Specification released in April 2011. A User guide for version 2.0 already exists and has been released on the web in 2009 (http://sdmx.org/?page_id=38).

As version 2.1 of SDMX contains several innovative parts (such as web services guidelines, new data messages, new code lists and metadata management) this new release intends to document how the new standard can be used to fulfil the most typical use cases and scenarios for data and metadata exchange.

Like in the previous version 2.0, the emphasis, throughout the guide, is on practical use of SDMX, supported by examples. In addition to the examples in the text, some sections of the User Guide refer to example files which are too long to be included in the document. These files, now available in a separate annex, will be available for download with the electronic version of the User Guide on the SDMX web site.

Some parts of the present draft may be considered as quite technical, so that other paragraphs, oriented to a non-technical audience, could be added. Therefore, it is expected that further revised drafts will follow, in the next months, also taking advantage of new examples from real-world usage, and responding to needs expressed by readers.

This version - drafted by Chris Nelson, Jack Gager and Arofan Gregory and discussed within a specific sub-group of the SDMX Technical Working Group coordinated by Marco Pellegrino - is presented to the SDMX Secretariat, to the SDMX Statistical Working Group and to the whole community of SDMX experts.

Comments are welcome and can be sent to the SDMX Secretariat (secretariat@sdmx.org) indicating the version of the User Guide to which comments refer.

The final User Guide – edited and approved by SDMX – will be posted on the sdmx.org website.

Contents

Introduction

1.1 Scope of the User Guide

This Guide covers all of the major use cases for SDMX from simple data reporting to a completely self-updating web dissemination system of data and related reference metadata. The use cases show how best to take advantage of the SDMX Information Model in statistical systems in order to manage data and metadata reporting, data and metadata storage and retrieval, and data and metadata dissemination.

The principal intention is help organizations and individuals to determine how best to use SDMX in order to help them to improve the statistical production process. In order to achieve this objective, examples are taken from real implementation scenarios that enable the reader to understand the scope of the SDMX standards and guidelines in terms of the activities required in order to collect, process, and publish statistical data and reference metadata.

The guide is concerned principally with the reporting/collection, processing, and dissemination of aggregated statistical data. Clearly a large part of the SDMX standard is technical in nature but the audience for this guide is anyone who is involved in the processes that eventually result in the publishing of statistical data either internally within the organization or externally to interested parties. Whether the dissemination is internal or external, this is achieved more and more using web-based technologies, for which SDMX is ideally suited.
The SDMX Information Model is pivotal to implementation of SDMX and systems built with knowledge of this Information Model are easy to maintain, enable data and metadata sharing both between internal systems and with the systems used by the organizations that collect data and metadata, and enable access directly the databases and metadata repositories of data and metadata producers.

There is a growing availability of software tools, many of them free of charge or open source, that support many of the aspects of SDMX. Use of these tools or of the underlying components of the tools can reduce dramatically the cost, internal resources, and the elapsed time of the development of a new system or the integration of SDMX into an existing system.

1.2 Structure of the User Guide

Chapter	Content
1. Introduction	Objective, Scope, and Structure of the Guide
2. What is SDMX	Background, sponsors, users, use cases, industry sectors. Brief overview of the technical and content standards, tools, where to find more information and help.
3. Scenario, Use Cases, and Example	This is based on the SDMX Information Model. The chapter relates the Information Model to the real activities of reporting, processing, and dissemination of statistics.

Chapter	Content
4. Data and Metadata Creation and Reporting	Explanation of the structural components of a Data Structure Definition and a Metadata Structure Definition, and of the Data Set and Metadata Set. How these are used in data and metadata reporting scenarios.
5. Data Bases and SDMX	Explanation of the relationship between the tables in a database and a Data Structure Definition and how the DSD can be used to create these tables. Explanation on how to open a database to SDMX web services.
6. Data and Structure Query	The scope of the data query and the map to the Information Model. Scope of the REST and SOAP queries. Useful tips on what type of queries to support.
7. Metadata Repository and Linking Data to Metadata	Typical requirements for metadata (quality frameworks, linking to disseminated data). Architecture for a metadata repository to enable data and metadata to be combined in a dissemination environment.
8. SDMX Registry	Role of the Registry in statistical data and metadata reporting and dissemination systems. Difference between the content and functions of a Registry and a non-registry based structural metadata repository. Content and role of the Registry in terms of: • Structural metadata maintenance and query • Registration of data and metadata sources • SDMX Registry Services • Web services that can make use of SDMX Registry Services

9. Architecture for an SDMX System	Brings all of the components together in an overall architecture comprising:
	
	• Data and metadata persistence and interfaces
	• Server side middle tier brokering of requests for data and metadata, data loading, validation, transformation
	• Client side tier of:
	  ○ Structural metadata maintenance
	  ○ Data query and visualization
	  ○ Validation, transformation

10. Community Management	The community may be at the level of an organization or in the context of the wider community of organizations. Topics are: • Role of a Global Registry ○ Maintenance agency maintenance

Chapter	Content
	○ Common concepts ○ Community structural metadata ○ Maintenance of Dimension and Attribute roles • Hosting of a shared Registry • Data provider maintenance

2 What is SDMX

2.1 Introduction

This chapter provides some background on the SDMX Initiative, the issues which SDMX addresses, the history of the standards and guidelines which have come out of this initiative, and the areas in which SDMX is playing a role today. This chapter also provides some guidance about how prospective users should think about SDMX as it relates to their own part of the statistical process.

SDMX offers a wide variety of technical tools, and these – like any tools – if used well, they produce positive results.. SDMX also offers a set of guidelines regarding the application of harmonized statistical concepts to data sets, and how these can be represented. Other guidelines address the classification of statistical data and domains, and the harmonization of relevant terminology. Additionally, SDMX represents a framework for the process of harmonization within domains. All of these different aspects are considered here.

2.2 Background – Official Statistics

SDMX comes out of the world of official statistics. If you work for a national or international statistical agency, you already understand “official statistics”, but many people who work with statistical data may not understand this domain, so we will provide a brief, high-level description.

“Official statistics” are the data which is collected and disseminated by a set of governmental and international organizations to provide the factual basis for making policy and supporting research. Some countries have a “national statistical office” (NSO) while others may have several governmental organizations which are charged with collecting statistical data for governmental use. Most countries also have central banks or similar organizations which collect and disseminate financial and economic data. Typically, several national government organizations have a statistical function (ministries of education, justice, labour, etc.)

These national organizations typically report their statistics to a set of supra-national organizations, representing either regions of the globe (examples include Eurostat and the European Central Bank) or domains (examples include the World Health Organization, the Food and Agriculture Organization, UNESCO, and the World Bank). Many of these organizations belong to the UN, or are treaty organizations.

All of these organizations exchange, report, and disseminate data in a chain which can be understood as starting at the lowest level within each country, and resulting in high-level data sets which are “aggregated” as they move through various levels to reach the international level.

The system of official statistics is this network of reported data, according to legal requirements or other types of agreements. There are several important meetings, conferences, and initiatives within this system, so that all organizations adopt similar approaches and techniques, and to coordinate reporting: the Conference of European Statisticians is an important meeting, as is the United Nations Statistical Commission meeting. Ultimately the goal is to measure important phenomenon occurring in the world, and to report the data to policy makers, students, journalists, and other users to help inform their activity. The data is “official” because it comes with the reputation of the world’s governments and international institutions behind it.

2.3 The History of SDMX and its Work Products

In 2001, the heads of seven international statistical institutions came together to form SDMX, with the goal of taking concrete steps towards addressing issues around statistical exchange. These organizations became the sponsors of the SDMX Initiative: the Bank for International Settlements, the European Central Bank, Eurostat, the International Monetary Fund, the Organization for Economic Co-operation and Development, the UN Statistics Division, and the World Bank. They created an initiative with a governing sponsors committee, and a secretariat function to execute the work programme.

The issues can be briefly characterized as follows:

- Statistical collection, processing, and exchange is time-consuming and resource-intensive
- Various international and national organisations have individual approaches for their constituencies
- Uncertainties (in 2001) about how to proceed with new technologies (XML, web services etc.)

The SDMX Initiative stated that it would address these issues:

- *By focusing on business practices in the field of statistical information*
- *By identifying more efficient processes for exchange and sharing of data and metadata using modern technology*

The initial projects of the initiative, based largely on work already on-going among various of the sponsor organizations, were:

- *A practical case study on emerging e-standards for data exchange*
- *Maintaining and advancing existing standards for time series data exchange*
- *Creation of a common vocabulary for statistical metadata*
- *Development of a framework for metadata repositories*

It was further stated that: “New standards should take advantage of the new web-based technologies and the expertise of those working on the business requirements and IT support for the collection, compilation, and dissemination of statistical information.”

Thus, the goals of the SDMX initiative were ones which were broadly agreed across the sponsoring organizations, and within the official statistics community generally. It is important to understand that there were some firm foundations on which SDMX was building:

1. An existing standard for exchanging statistical data, known as GESMES/TS, was already in use among several of the sponsor organizations and their national-level counterparties. This was based not on modern Web technologies such as XML, but used the older UN/EDIFACT syntax.

2. The work on the “metadata common vocabulary” was based on many years of harmonization work within the community, notably Eurostat’s *Concept and Definitions Database (CODED)* and the *OECD Glossary of Terms*.

The formation of the SDMX Initiative can be understood as a recognition by the sponsor organizations that working together to address these issues, and that coordinating business approaches using modern, standards-based technology, was the best way forward. In one sense, SDMX evolved from earlier work, but indicated the increased commitment the sponsors had toward reaching its goals. It also represents a coming-together of efforts around harmonizing statistical content and terminology, and for deploying technology to support statistical processes.

Over time, the work of the SDMX Initiative has expanded, both in terms of content-oriented work products and technical ones. We will describe the evolution of these work products below.

The SDMX Initiative decided early on to position the content-oriented work and the work on technology and standards in a fashion which made these strains of work separate but complimentary. The content-oriented work led to the development of the SDMX Content-Oriented Guidelines, while the technical work resulted in the SDMX Technical Specifications. There were several reasons for taking this approach. It reflected the realization that technical specifications must be very precise and detailed in order to allow for automation of statistical exchanges – the programming of computers relies on having very specific rules about how applications communicate, otherwise the communication fails. The SDMX technical standards in one sense function as exchange protocols for machine-to-machine communications (similar to HTTP, for example, but with a focus on specifically statistical exchanges).

Statistical content and terminology issues are very different – they are the subject to interpretation and analysis by trained statisticians. Thus, the technology specifications formed a basis for supporting work on the content side, but in fact are a very different type of work product. It is easiest to see this in the fact that the SDMX Content-Oriented Guidelines are *guidelines,* to help suggest approaches to people in their statistical work,

while the SDMX Technical Specifications are *specifications* - rules for developing conforming computer applications.

Another reason for this separation is that the technical specifications and content guidelines were expected to be maintained at different rates – once stable, technical specifications tend to be updated less frequently. Also, the reasons for making updates and changes in each area have no dependency between them, so it made sense to separate them. This is reflected in the fact that the technical specifications are submitted and published through the International Standards Organization (ISO), who publish many IT-related standards in various domains, while the content-oriented guidelines are not submitted to ISO, but are maintained by the SDMX Initiative itself. This allows for updates of the content-oriented guidelines on an on-going basis.

A third reason for the separation of the SDMX Technical Standards and the SDMX Content-Oriented Guidelines is that – because they are a technological foundation for exchanging *any* statistics – the technical specifications are applicable outside the domain of official statistics, while the content-oriented guidelines are specifically designed to be useful within that context (although they might also be useful outside that community, possibly).

This coordinated-but-separate positioning of the two threads of work has proven to be very useful, too, because often statisticians and economists do not have deep expertise in IT, and technologists do not have deep expertise in statistics. SDMX helps to define the point where the two sets of expertise need to coordinate, to effectively use IT within statistical exchanges and processes.

Within the content-oriented work, there are a set of work products: *The Content Oriented Guidelines,* and 5 annexes:

1. Cross-Domain Concepts
2. Cross-Domain Codelists
3. Statistical Subject-Matter Domains
4. Metadata Common Vocabulary
5. SDMX-ML for the Content-Oriented Guidelines (Concepts, Code Lists, Category Scheme)

These are discussed in more detail in the first annex to this user guide.

The SDMX Technical Specifications are now in version 2.1, but both version 1.0 and version 2.0 were implemented. The 1.0 version of the specifications have relatively limited coverage – a model for data formats and their structures, along with XML and UN/EDIFACT formats for exchanging these. The UN/EDIFACT format was backward-compatible with GESMES/TS; the XML formats were new. There was also some support provided for SDMX-based Web services: an XML query document, and a set of guidelines about the use of other related Web-services standards (SOAP and WSDL).

The 2.0 version of the technical specifications had a greatly-expanded scope. The model was extended to include “reference metadata” as a way of structuring and formatting

metadata related to data quality frameworks, methodological metadata, and other types of "footnote" metadata. Thus, XML formats for reference metadata were added. Further, a set of standard interfaces in XML for interactions with a SDMX Registry were added, for cataloguing the location of data and reference metadata across the Internet or within an organization, and for maintaining and retrieving structural metadata.

In version 2.1, many features of 2.0 have been improved, and the Web-services recommendations have been expanded to include a RESTful interface, standard functions, and error messages. Now, it is possible to develop generically interoperable applications based on the SDMX standards. Further, the various XML data formats have been simplified based on implementation experience with version 2.0.

For all types of work products, there have been internal reviews within the SDMX community, and also public review of the guidelines and standards.

2.4 The SDMX "Toolkit" Approach

There are many different elements in the SDMX suite of guidelines and specifications, and it may seem daunting to think of implementing them all. It is important to understand the philosophy behind this suite of tools. SDMX has always taken seriously the idea that different organizations will implement at their own speeds, and with their own objectives. As much as possible, they have recognized that investments in legacy systems must be protected, and that existing content and processes should still be supported.

The result of this requirement has been the "toolkit" approach: SDMX offers many different tools, but they need not all be adopted or used together. Indeed, many tools are now built on top of a more fine-grained set of components which themselves can be integrated into an organisation's own systems. The technical specifications outline a number of different types of conformance with the specifications, based on which parts of the specifications are being used.

The following chapter describes the use-cases which SDMX supports, but a basic list of business applications can be given:

- Collection cases
  - SDMX as a "push" reporting format for data and metadata (reporter pushes data to collector)
  - SDMX as a "pull" reporting format for data and metadata (collector pulls data from reporter)
- Dissemination cases
  - SDMX for file downloads
  - SDMX as a queryable data source
  - SDMX to drive website presentation of data and metadata
- Data warehousing cases
  - SDMX for extraction, transformation, and load of data
  - SDMX as a model for the structure of a data warehouse or metadata repository

Different parts of the standard are used for each of these cases (and others), and the specifications are specifically written to allow only the relevant parts of the standard to be used by any given application.

2.5 Uptake of SDMX within Domains

SDMX has become very widely used within the world of official statistics, so much so that it is difficult to form a comprehensive list of users. This section attempts to characterize the current users of SDMX – a group that will likely grow not only in terms of numbers, but also in terms of the breadth of applications. A few possibilities here are suggested at the end of this section.

If we are to look at the most common uses of SDMX, there are two

1. The use of SDMX as a reporting and collection format, which is especially prevalent within the central banking community (as a result of the earlier implementation of GESMES/TS, now SDMX-EDI), and also among the statistical agencies in Europe (also users of GESMES historically, but implementation is now increasingly driven by such projects as Eurostat's Census Hub);

2. Dissemination of statistical data from websites.

The second application is one which we see in a broad range of institutions, including central banks (ECB and European System of Central Banks, BIS, U.S. Federal Reserve Board and New York Federal Reserve, among others), other sponsoring institutions (IMF, World Bank, OECD, etc.), and national statistical agencies (INEGI in Mexico, Statistics New Zealand, Australian Bureau of Statistics, statistics offices in the European Statistical System etc.)

A less-common but growing use of SDMX is as the basis for data warehouses and other forms of data management. Perhaps the best example of this is the European Central Bank, which has created all of its internal data warehouses around the SDMX Information Model, and has realized many benefits from this. They are by no means the only organization looking at this type of implementation, however – many other organizations are using SDMX to manage not only their statistical data, but also to create metadata repositories, and to integrate their metadata and data.

If we look at which statistical domains have been or are becoming major adopters of SDMX, the list would be something like this (in no particular order):

- Census and Demography
- Education
- Financial and Monetary Indicators
- Economic Indicators
- National Accounts
- Labour

- Food and Agriculture including fisheries
- Epidemiology
- Transport
- Data Quality
- Development Indicators

It is easy to see that this is a broad and cross-cutting set of statistical domains – in fact, there are probably very few domains in which SDMX is not being used in some fashion today, and the above list is intended as an indication of the breadth of the uptake..

SDMX was officially endorsed first within the European statistical system, and then by the UN Statistical Committee. These endorsements were powerful incentives for organizations to use SDMX, and the result has been widespread adoption. There are no major competing standards, which has saved the world of statistics from a phenomenon which has slowed the uptake of standards in some other communities.

Additionally, a strong culture of open-source and free tools development has emerged, helping to make the adoption of SDMX easier. This has come both from within the sponsors' community and without, and is supplemented by an increasing number of tools coming from commercial vendors as well.

To learn more about available SDMX tools, the best place is to consult the SDMX Tools Database, a service provided by the SDMX sponsors, linked to from www.sdmx.org.

And support for the standards does not only take the form of tools – The Open Data Foundation hosts the SDMX User Forum in collaboration with the sponsors, providing a place where the community can interact online, and Eurostat's CIRCA website provides many types of resources, from training videos to student guides. Many organizations offer SDMX in-person training for different levels of users. The best single point of entry is of course the SDMX website itself.

Looking forward, SDMX is increasingly coming into use: at the most recent SDMX Global conference held in Washington DC in May 2011, Google showed some interest in SDMX as a source of data for its Data Explorer; there is now an interest in setting up a global registry so that all SDMX data and metadata sources can be easily found. Further, we see the strong possibility that the world of corporate statistics may realize the utility of having a strong standards basis around the vast amounts of data collected today to support business intelligence applications.

3 Use Cases, Scenario, and Example

3.1 Scope of this Chapter

In order to present a common theme throughout this User Guide a common set of use cases, a common scenario, and a common set of examples are used. This chapter specifies the use cases, scenario, and the example SDMX structural metadata used.

3.2 Use Cases

SDMX supports many use cases.

Use Case	System Processing Activity	Structural Metadata Requirements
Data Reporting	Extract data from the source (database, file, spreadsheet)	Access to DSD.
	Write an SDMX-ML or SDMX-EDI data set	Access to DSD and possibly code transformations and aggregations.
	Validate the data set	Access to the DSD or an XML schema derived from it.
Load data into a database	Read an SDMX-ML or SDMX-EDI data set. Validate the data set. Write data to database	Access to DSD and possibly code transformations.
Report reference metadata	Extract metadata from a database	Access to MSD.
	Write metadata set	Access to MSD.
	Validate metadata set	Access to MSD or schema derived from it.
Load reference metadata into a database (often called a metadata repository)	Read SDMX-ML metadata.	Access to MSD and possibly code transformations.
	Write metadata to database	Access to MSD.
Report data by means of the “pull” method	Extract data Write data set Validate data set	Access to DSD
	Place the file at a URL	

Use Case	System Processing Activity	Structural Metadata Requirements
	location	
	Publish the existence of the dataset by means of an SDMX Registration	Provision Agreement in an SDMX Registry.
Database Administration (automatic generation of database tables)	Create database tables	Access to DSD
	Load database	Access to DSD
Enable database to be compatible with SDMX Web Services	Accept and process and SDMX structure query	Access to SDMX structural metadata such as DSD, MSD, Dataflow, Metadata flow, Data Provider, Provision Agreement, Constraint, Category Scheme.
	Accept and process an SDMX data query	Access to DSD.
	Write SDMX data set	Access to DSD.
Data Discovery	Locate data source	Category Scheme and links to Dataflow, Provision Agreement and data Registration
Data and Metadata Query and Visualisation	Query an SDMX structural repository.	Access to SDMX structural metadata.
	Create an SDMX query from the user selections	Access to DSD.
	Access a metadata repository to extract referential metadata pertaining to the data.	Access to code lists and concepts.
	Transform the SDMX data and referential metadata into tables, graphs, charts etc	Access to DSD and metadata code lists and concepts.

3.3 Scenario

3.3.1 Web Dissemination Use Case

The use of web dissemination is the prime use case for this User Guide and the chapters in this guide follow this process flow. However, the topic of each chapter is relevant to more than this single use case of SDMX. For instance, a Data Structure Definition is relevant to all use cases concerning data, and an SDMX Registry is relevant to virtually all use cases. The reason for choosing web dissemination of data and related metadata is twofold:

- this single use case uses many of the SDMX constructs
- it is an increasingly popular use of SDMX as it enables organizations to build a self-updating dissemination system

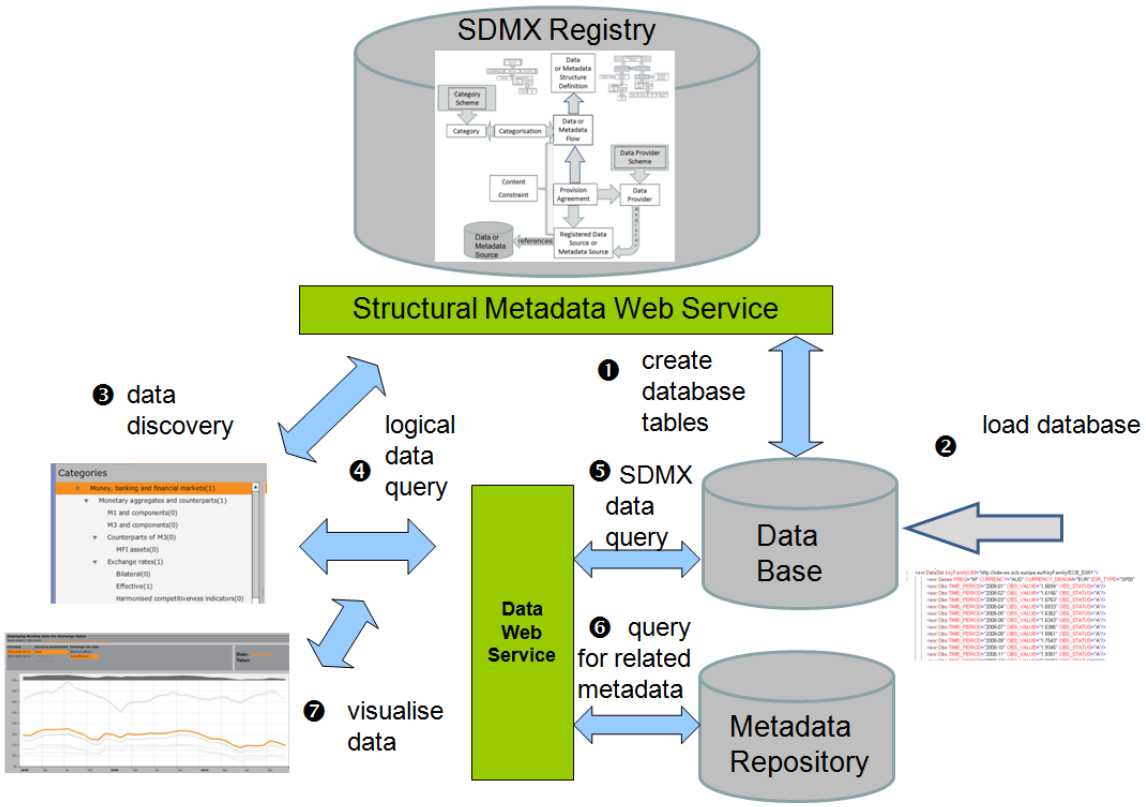


Figure 1: Process flow of an SDMX Web Data Dissemination System

Process	Description
❶	Retrieve the DSD from a structural metadata source (e.g. an SDMX Registry), and create database tables.
❷	Read an SDMX data set file and load the data into the database
❸	Data discovery system continually synchronises its metadata with the structural metadata source. A user makes a data selection from choices built from the information held in an SDMX Registry (structural metadata such as category scheme, dataflow, DSD, data provider, provision agreements and data registration)
❹	These choices are logical choices, built from the dimension selections.
❺	The logical choice is formatted as an SDMX data query. This is passed to the Data Base which responds with an SDMX data set.
❻	Reference metadata relevant to the data returned is retrieved from a metadata repository.
❼	The data and metadata are passed to a visualization tool to display the data

Process	Description
	in tables, charts, graphs, maps etc. Often a download is offered in various formats. The download options often include also the DSD or MSD.

3.3.2 Structural Metadata

The following structural metadata and provisioning metadata is used in the scenario.

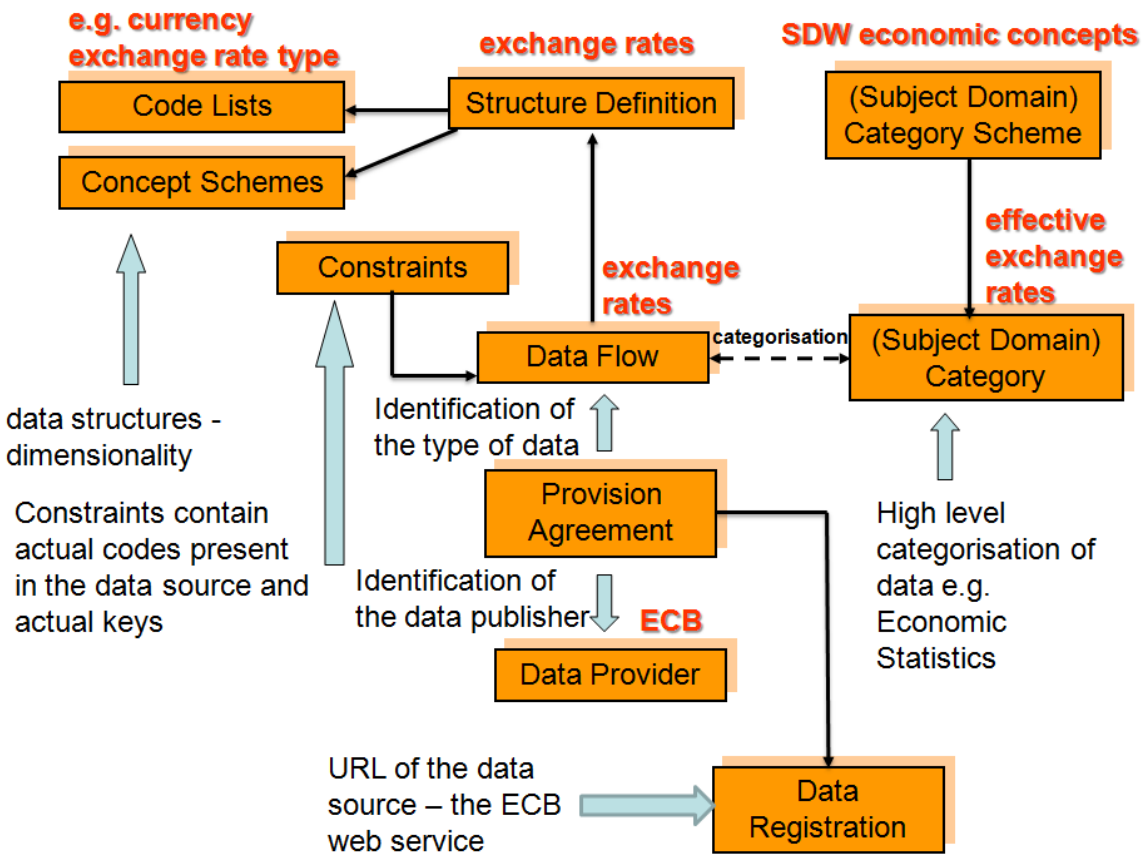


Figure 2: Structural and Provisioning Metadata Used in the Scenario

The content of these metadata is described in the chapters that follow.

4 Data and Metadata Creation and Reporting

4.1 Scope of this Chapter

Use Case	Relevant
Data Reporting	☑
Load data into a database	☑
Report reference metadata	☑
Load metadata into a database (often called a metadata repository)	☑
Report data by means of the "pull" method	☑
Database Administration (automatic generation of database tables)	☑
Enable database to be compatible with SDMX Web Services	
Data Discovery	☑
Data and Metadata Query and Visualisation	☑

This chapter covers the creation of data and metadata for reporting purposes, starting with the definition of the structures which the data and metadata are to be reported against. This chapter references two sets of samples, the details of which are provided in Annex 3 – Data and Metadata Samples. This section focuses on the details of the fundamental features of SDMX whereas the samples referenced in the Annex demonstrate the syntactical representation of these features in SDMX-ML.

4.2 Basics

Fundamental to defining a structure definition is defining the concepts which describe the information contained in the data or metadata and the code lists which provide specific values for these concepts.

4.2.1 Defining Concepts

Any component of a data or metadata structure definition must take its semantic from a concept. This concept is important in that it;

1. Provides a detailed definition of the component which describes the structure of the data or metadata.

2. Can allow data and metadata for different structures to be comparable when concepts are reused.

It is important when defining concepts to first consider concepts which are already defined within a given community, whether it be the SDMX community as a whole or a smaller community of users in a particular sector. As general rule, one should not define a new concept when an existing concept will suffice. By reusing an existing concept, data and metadata are more easily understood in a wider range of applications. This leads to greater interoperability and comparability of data/metadata.

Assuming new concepts are to be defined, the first step is to determine an appropriate grouping for concepts. In SDMX, all concepts are defined in schemes. These schemes serve to group similar concepts into groups which can be useful for maintenance purposes. One possible distinction to be used to determine the grouping of concepts is their intended usage. If concepts are only to be used for metadata, it may be best to group these into a single scheme. Similarly, if concepts are only to be used for data, these too may be grouped into a single scheme. It is important to consider that concepts themselves are not versioned, rather the schemes in which they are defined are versioned. This means that if any property of a concept is to change, the version of the scheme in which it is defined must change if the scheme is marked as “final”. This in effect versions all concepts defined in that scheme. Therefore, when grouping concepts into schemes it is important to consider the stability of the concept definitions.

A concept itself consists of three main components;

1. Its identification, which must be unique within the scheme.
2. Its name.
3. Its description.

Note that the description is not mandatory, but is highly recommended in order to provide a more complete definition of a concept. When you consider that these concepts are the building blocks which are constructed to define the structure of all data and metadata in SDMX, it should be apparent why complete definitions are important.

In addition to the basic definition properties of the concept, a concept can reference another concept from within the same scheme as its parent concept. The exact nature of this parent child relationship is not strictly enforced by the standard, but it is typically used to denote the child concept is a specialization of the parent. For example, one may have a concept which defines a "reference area" concept as a "geographic area to which a measured statistical phenomenon relates". In order to allow for more specific types of references areas, one might define concepts for countries as well as groups of geographically similar countries (e.g. continents) and political countries (e.g. military or economic alliances). These child concepts could reference the "reference area" concept as a parent in order to note that they are specializations of a "reference area". An example of this can be seen in the Common Structures sample set. In this sample the confidentiality status references the confidentiality concept as a parent. The status is a specialization of a general confidentiality concept.

< note the test for whether something is a specialization is to ask the question “is xxx a type of yyy” In the Cross Domain Concepts the question would be “Is confidentiality status a type of confidentiality”. The answer is “no”. The problem here is that the cross domain concepts are grouped, often in a structural and not semantic sense. This, unfortunately, is carried over in the SDMX-ML and needs to be rectified.>

When a component in a structure definition takes its semantic from a concept, it always is provided a value in the data or metadata reported against the structure. A concept can define the default nature of these values. This is done with a definition of a *core representation*. The core representation can either be a un-coded text format, in which a data type is defined along with facets which serve to further restrict that data type, or a coded enumeration in which a code list can be referenced. In the case where an enumerated core representation is defined, the nature of the enumeration values can be described in the same manner that the un-coded text format could be. Note that if a concept is not provided a core representation, it is assumed to have an unbounded value set (i.e. it could be represented by a variety of code lists or un-enumerated formats when used in a DSD or MSD). However, as with any core representation, a usage of the concept in a structure definition can always provide a local representation which overrides the core representation of the concept.

Finally, in order to provide a more complete conceptual definition, the concept can reference an ISO 11179 concept. The intention of this reference is to reference the ISO 11179 concept definition from which this SDMX concept definition is derived.

4.2.2 Defining Code Lists

Any component in a structure definition can specify an enumeration of possible values. These enumerations are defined as code lists. A code list contains a value set of enumerations and the names and descriptions of what is represented by the coded value.

The code list itself is provided an identity, name, and description. The codes comprising the code list have the same properties. Note that the identification of the code is its code value which will be used in any data/metadata sets.

Codes can also be arranged into simple hierarchies, where any code can reference another code within the same list as its parent. Note that SDMX does not formally define the exact nature of this relationship (e.g. whether the child code is additive to the parent and if so, whether there is a weight associated with the code). None the less, it is often useful to capture the fact that some formal relationship does exist, if onlyto allow for more detailed descriptions of these code lists to be accessed in order to properly understand the data/metadata.

4.3 Data

Fundamental to SDMX is the exchange of data, or more specifically multi-dimensional data against a known structure. Data in SDMX starts with the structure which defines it. This section details the data structure and its relationship to data sets.

4.3.1 Defining Data Structures

A data structure is a collection of components which define what is being measured and what additional properties (metadata) can be transmitted alongside the actual observed data values.

4.3.1.1 Data Structure Components

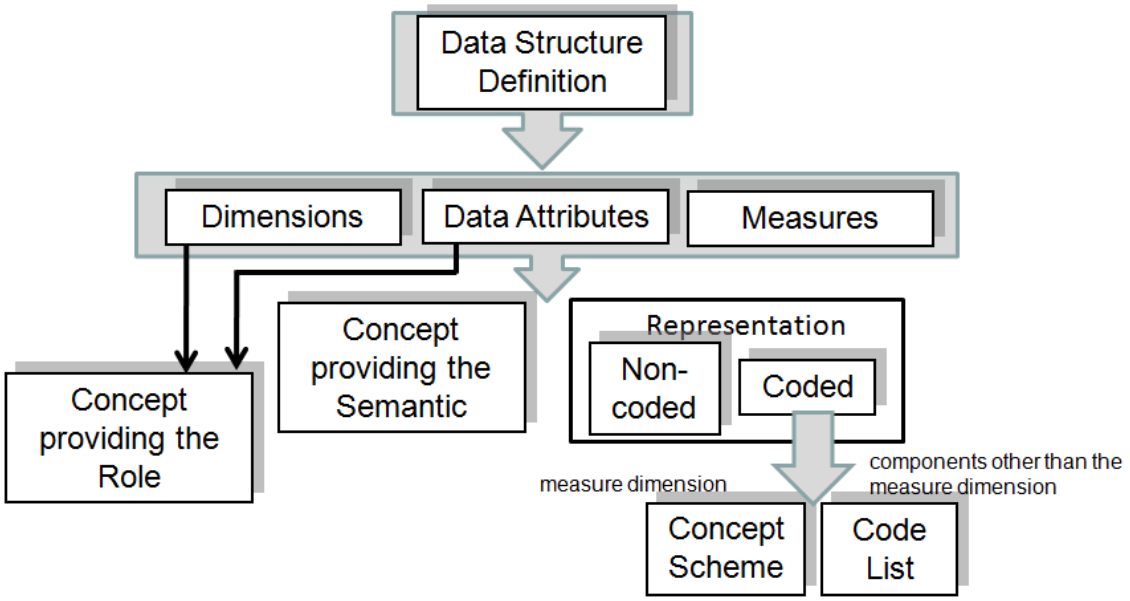


Figure 3: Schematic of the Data Structure Definition

Each component must reference a concept from which it takes its semantic. This concept defines the meaning of the component. Typically, the component will also take its identity from this concept, although it is possible for a (data) structure-specific identity to be created. Regardless of whether the concept identity is used or a local identifier is assigned, the identifier of a component must be unique within the scope of all data structure components.

A component can also inherit the representation of the concept from which it takes its semantic, or it can provide a data structure specific representation for the concept. This representation can be either code or un-coded, although some component types have more restrictive representations.

A data structure component can also reference a collection of concepts for the purpose of identifying specific roles that the concept serves in the data structure. For example, a user community may define a collection of concepts which are intended to note special roles for data structure components (e.g. the unit of measure). A component can reference any of these concepts in order to specify that the role identified by the concept is being served by the component. Note that a component is implied to serve the role of the concept from which it takes its identity, therefore it does not need to reference this concept again.

4.3.1.1.1 Primary Measure

Every data structure must define a primary measure. This component always has a fixed identifier (OBS_VALUE), and its purpose is to given a consistent data structure artefact where the observed value can be found. This makes the processing of data messages much more consistent, as the measured value is always readily found. The primary measure can define a representation, or it can inherit the representation from the concept from which it takes its semantic (this concept can have, but need not have, and Id of OBS_VALUE).

4.3.1.1.2 Dimensions

The identification of the phenomenon being measured is achieved through the dimension descriptor. Any given data structure must have at least one dimension. The dimensions (in some models these are known as “classificatory variables”) reference concepts which define identifying properties of the phenomenon being measured. In a dataset, each dimension is given a value. This set of dimension values is often referred to as a key. In any given data set, there can only be one observed value and collection of data attribute (metadata) values per key. Therefore, a key uniquely identities any observed phenomenon.

There are two specialized dimensions that can be defined only once for a data structure. The first is the time dimension. The time dimension represents the point in time at which the phenomenon was observed. If the explicit time dimension is used, then it must use any or all of the allowable time representations defined in SDMX. These representations are:

1. A Gregorian calendar period (which can be any or all of a year, a month and year, or a date)
2. A standard reporting period (which can be any or all of a year, a semester, a trimester, a quarter, a week, or a day). Each reporting period exists in the context of a reporting year which is defined by a start day (expressed as a month and day). This start day is communicated in a specialized data attribute which will be discussed later in this section.
3. A distinct duration, which consists of a start date and time and a duration
4. A distinct point in time (i.e. a full time stamp)

Each of theses time representation allows for a time zone offset, so that the exact period of time encompassed by the value can be expressed with absolute precision.

The time dimension has a fixed identification which allows for strictly time series formats to be created. These explicit time series formats will be discussed in the Data Sets section.

The other specialized dimension is the measure dimension. A measure can refer to a collection of properties for which phenomena is being measured for an entity classified by the other dimensions of the data structure (e.g. the demographic measures in the demography data structure). A measure dimension might also refer to the different

means in which a phenomenum may be measured (e.g. weight, volume, and price for a commodity).

A measure dimension must always take its representation from a concept scheme. This concept scheme must define a collection of concepts which define the value set of the measure dimension concept. An example of this can be seen in the Eurostat Demography data structure. For example, the demography measures concept scheme (DEMO_MEASURES) contains only concepts which define demographic measurements.

An advantage of defining a measure dimension is that it also allows for a more explicit definition of the representation of the observed value for a given measure. This is achieved by defining core representations for the measure dimension concepts. Note that it is necessary that the primary measure representation in this case be a superset of all possible representations of the measure concepts. This is demonstrated in the Eurostat Demography data structure. An analysis of the demography measures concept scheme shows that the number of deaths in a year is measured as an integer, whereas the life expectancy is measured as a decimal with only one decimal digit. This representation is carried over to the structure specific schema when the measure dimension is used as the observation dimension, and explicit measures are used. This can be seen in the data structure specific schema for the demography data structure.

By defining a measure dimension, a user of the data reported against the data structure will be able to better understand the relationships that exist between the observed values. Another advantage of using a measure dimension is that the data structure can be specific to the representation of observed value because it relates to a specific measure (i.e. it is a concept in a the concept scheme referenced from the measure dimension).

4.3.1.1.3 Attributes

A data structure can also define additional components which serve to hold additional information (metadata) about the data. This information can be presentational in nature (e.g. a series title) or be critical to understanding the data (e.g. the unit of measure). The ECB Exchange Rates data structure contains both of these attributes.

There is one specialized attribute in a data structure, and this is the reporting year start day. If the time dimension of a data structure is allows for reporting periods in its representation then it is strongly recommended that this attribute be used. It has a fixed identifier and representation (a month and day). The purpose of this attribute is to communicate the reference point for a reporting year. This reference point allows the exact calendar period for a reporting period to be determined. If this is not present, then the basis for all reporting periods will be assumed to be January 1.

In addition to the aforementioned component properties, an attribute must define its relationship to the other components of the data structure (i.e. the dimensions or the primary measure). This relationship states how the value of the attribute varies with the value of other components. In the Eurostat DemographyEurostat Demography data structure, the unit of measure attribute (UNIT_MEASURE) specifies an attribute

relationship with the demographic measure dimension (DEMO). This should be intuitive, since the unit of measure differs if one is measuring a count, such as the number of live births in a year, or a rate, such as the fertility rate.

A data structure can define groups for the purpose of specifying attribute values. The advantage of defining groups is to avoid repetition of attribute values in a data set. Each group consists of a unique subset of the dimensions. Attributes can either explicitly specify a relationship with the group, or they can specify relationships with specific dimensions yet still reference a group for attachment purposes (although the dimensions with which the attribute have a relationship must all be part of the group dimensions).

4.3.2 Data Sets

Every data set in SDMX must conform to a data structure definition. As described above, the data structure definition organizes concept definitions into various components which identify and supplement the data. When processing data, it is critical to be able to retrieved and fully understand the data structure definition. A clear understanding the data structure is dependent on well defined concepts. Therefore, useful data starts with well defined concepts. Ultimately, understanding is dependent upon understanding the concepts which define what is being measured.

In terms of processing data, the data structure definition provides enough information so that the data can be validated and understood. However, the data structure does not dictate the exact orientation of the data. The orientation of the data is defined by the data set itself. However, SDMX only allows for two basic orientations:

1. A flat list of observations in which the full key is provided for each observed value.
2. A collection of observations in series where all but one dimension has the same value and each observation is distinguished by the other dimension (e.g. a time series in which a series has a key and each observation in the series has a distinct time value). This dimension if known as the observation dimension.

A data set may also contain groups, if the data structure defines them. Each group will have a unique set of key values and provide the attribute values associated with the key set.

Data sets note the orientation by defining the observation dimension. In the case of the flat orientation, the observation dimension is actually all dimensions. In the second case, it is a specific dimension from the data structure (this must be the same dimension for the entire data set).

This observation dimension dictates where attributes should be present, based on the attribute relationships defined in the data structure. Any attribute which has a relationship with the observation dimension (i.e. the dimension is a part of a group or a set of dimensions with which the attribute has an “attribute relationship”) must exist at the observation level. This also holds true if an attribute has a relationship with the primary measure. If an attribute has a relationship with no data structure components, then only one value is provided per data set (i.e. the attribute exists at the data set level).

If an attribute has a relationship with a group, or specifies a group for attachment, the attribute will be communicated at the group level. In all other cases the attribute will exist at the series level.

Data can be expressed in one of four formats;

1. Generic
2. Time series generic
3. Structure specific
4. Time series structure specific

The time series formats are actually equivalent in content of their more generalized counterparts when they specify the time dimension at the observation level. Therefore, with the exception of the root element name, a generic data set with time at the observation level will be the same as a time series generic data set. The difference in these formats is that the time series only allows for time to be the dimension at the observation level.

Note that regardless of the organisation or the format, the data expressed is always the same. This can be seen by examining the various data messages for the ECB Exchange Rates data. Regardless of the organisation or the format, the data expressed is always the same. In fact, this even holds true for the attribute values even though they are expressed at different levels depending on their relationships. This serves to show that it is critical that the attribute relationship be specified correctly, otherwise the correct value cannot be expressed in a data message.

4.4 Metadata

Reference metadata enables additional information to be attached to data or structural metadata. The design of the reference metadata model allows informational metadata to be;

1. Structured and validated
2. Late bound to the structural metadata or data to which it applies

Consider contact information for a data set. This information could be contained is a single data attribute which could be carried in the data message. However, the nature of data attributes is that they have no sub structure. Therefore, there would be no means of separating the name of the contact person from the phone number (outside of creating many attributes to hold this information). In addition, this contact information is probably subject to change. It would not make much sense to update the data set when the data itself is unchanged simply to specify a new contact.

This is where reference metadata is useful. The structure of contact information can be clearly specified, and attached dynamically to the data. This same dynamic applies to structural metadata as well. Although all structural metadata components contain annotations, these often do not allow for the structure that is necessary to communicate the desired information.

A major use case for reference metadata is in the support of quality frameworks, where the metadata are not concerned with a data set but with the processes, regulations, and policies of data collection and dissemination.

4.4.1 Defining Metadata Structures

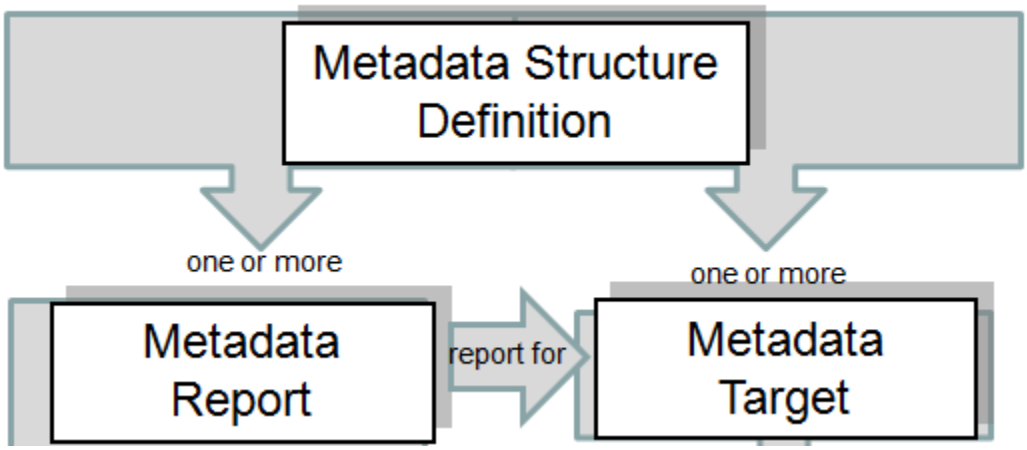


Figure 4: Schematic of the Basic Structure of a Metadata Structure Definition

A metadata structure defines two types of component lists. This first type of component list, the metadata target, serves to identity the types of objects to which the metadata described by this structure can be attached. The second type of component list, the report structure, identifies the content of the metadata reports which can be attached to the target objects.

This is, in a sense, similar to a data structure. In a data structure the dimension list describes how one identifies what is being measured, and the primary measure and attributes describe the details of that measurement. Similarly, the metadata structure uses the metadata target to describe how one identifies what the report pertains to, and the report structure defines the nature of the report in terms of what is to be reported (metadata attributes) . The fundamental difference between the metadata structure and the data structure is that whereas a data structure only has one set of dimensions, attributes, and primary measure, a metadata structure can define multiple targets and report structures, and has no measures.

4.4.1.1 Metadata Target

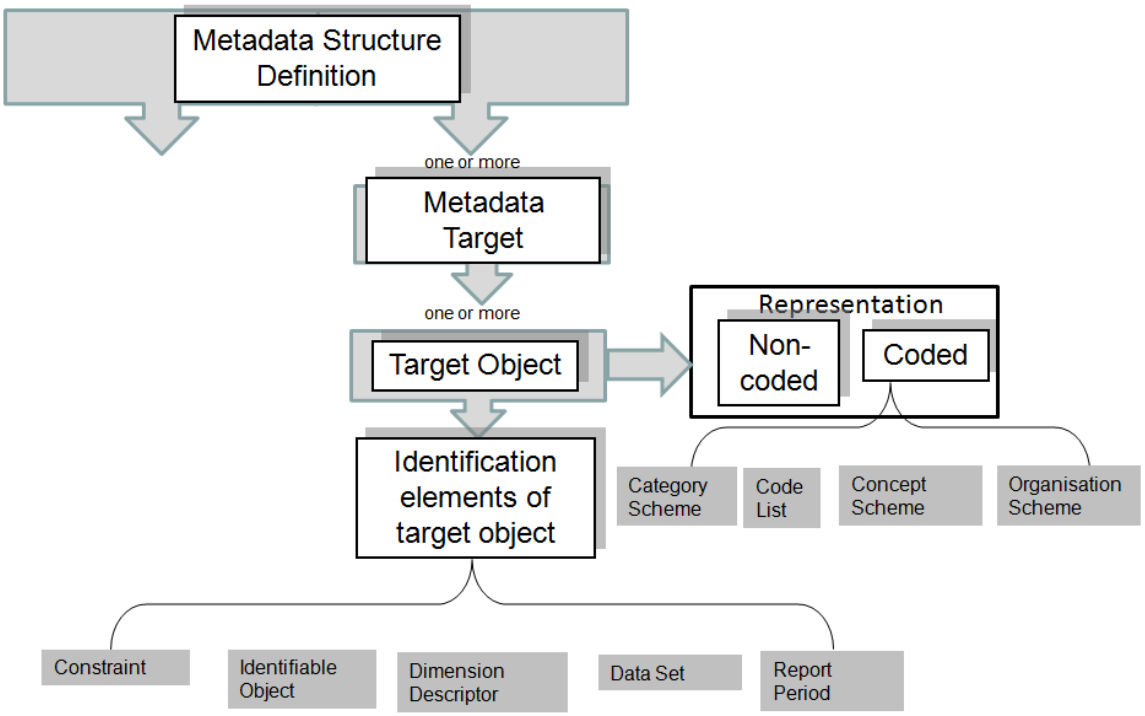


Figure 5: Identification of Targets in a Metadata Structure Definition

A metadata target defines what is expected in the metadata set in order to identify the object to which the metadata pertains. It is given a unique identifier within the metadata structure in which it is defined. It consists of one or more target object descriptors, each of which themselves have a unique identifier within the metadata target.

There are 5 types of target objects that one can use, each of which serves to uniquely identify an object within the SDMX information model.

4.4.1.1.1 Data Set Target

The data set target is used to attach metadata to a specific data set, which is identified by the identification of the data provider and the provider assigned identification of the data set. This target object has a fixed identifier and representation, so the purpose of defining this in a metadata target is to simply state that the data set reference is part of the metadata target value set. Only one data set target can occur within a metadata target.

4.4.1.1.2 Identifiable Object Target

The identifiable target object is used to attach metadata to any identifiable object in the SDMX information model. This type of target object can be repeated and each instance is assigned a unique identifier within the metadata target. Each instance identifies the type of object which is reference by this target. The identification of the target object is always a complete reference. If the target identifiable object type is an item from within

an item scheme, the representation of the target object can reference a scheme for the purposes of limiting the items to which metadata can be attached.

4.4.1.1.3 Dimension Descriptor Values Target

The key descriptor values target is used to attach metadata to data by identifying full or partial data "keys" (a collection of dimension identifier/value pairs). By itself, this target object is typically not descriptive enough as it does not identify the type of data to which the keys apply. Therefore, this is typically used with other target objects, such as the dataflow or data structure, which can identify these data. This target object has a fixed identifier and representation, therefore the metadata target is simply stating that it contains a key descriptor value set. Only one key descriptor values target can occur within a metadata target.

4.4.1.1.4 Constraint Content Target

The constraint content target is used to attach metadata to data by referencing an attachment constraint. This attachment constraint in turn defines the data set(s) and keys to which the metadata applies. This is equivalent to using the data set or identifiable object target along with the key descriptor values target. The difference is that the attachment constraint allows the target set to be defined once, and be reused by multiple reports, whereas the former method would require that the data set (or equivalent data structure, dataflow, or provision agreement) reference and key descriptor value set be repeated for each report. Only one constraint content target can occur within a metadata target.

4.4.1.1.5 Report Period Target

The report period target is used to state the reporting period for which metadata is applicable. This effectively allows the metadata to change over time while persisting the historicity of the changes. This target object has a fixed identifier, but its representation can specify the specific type of date that can be used. By default, this is the least restrictive date format. Only one report period target can occur within a metadata target.

4.4.1.1.6 Composing Targets

A metadata target consists of one or more of the target objects described above. It is the sum of these targets which identify the actual target for the metadata. For example, one might attach metadata to a specific portion of data in a given data set. In this case, the data set target object and the key descriptor values target object would be used. In the metadata set, the data set target would identify the data set and the key descriptor values target would identify the portions of the data to which the metadata applies. In another example, one might wish to be able to attach metadata to portions of data for all data sets reported against a given data structure. Rather than repeating the metadata for each data set, a metadata target would be defined which contains an identifiable target object which references a data structure and a key descriptor values target object. When designing a metadata target one must consider how generally the metadata which is reported against the target might be applied.

4.4.1.2 Report Structure

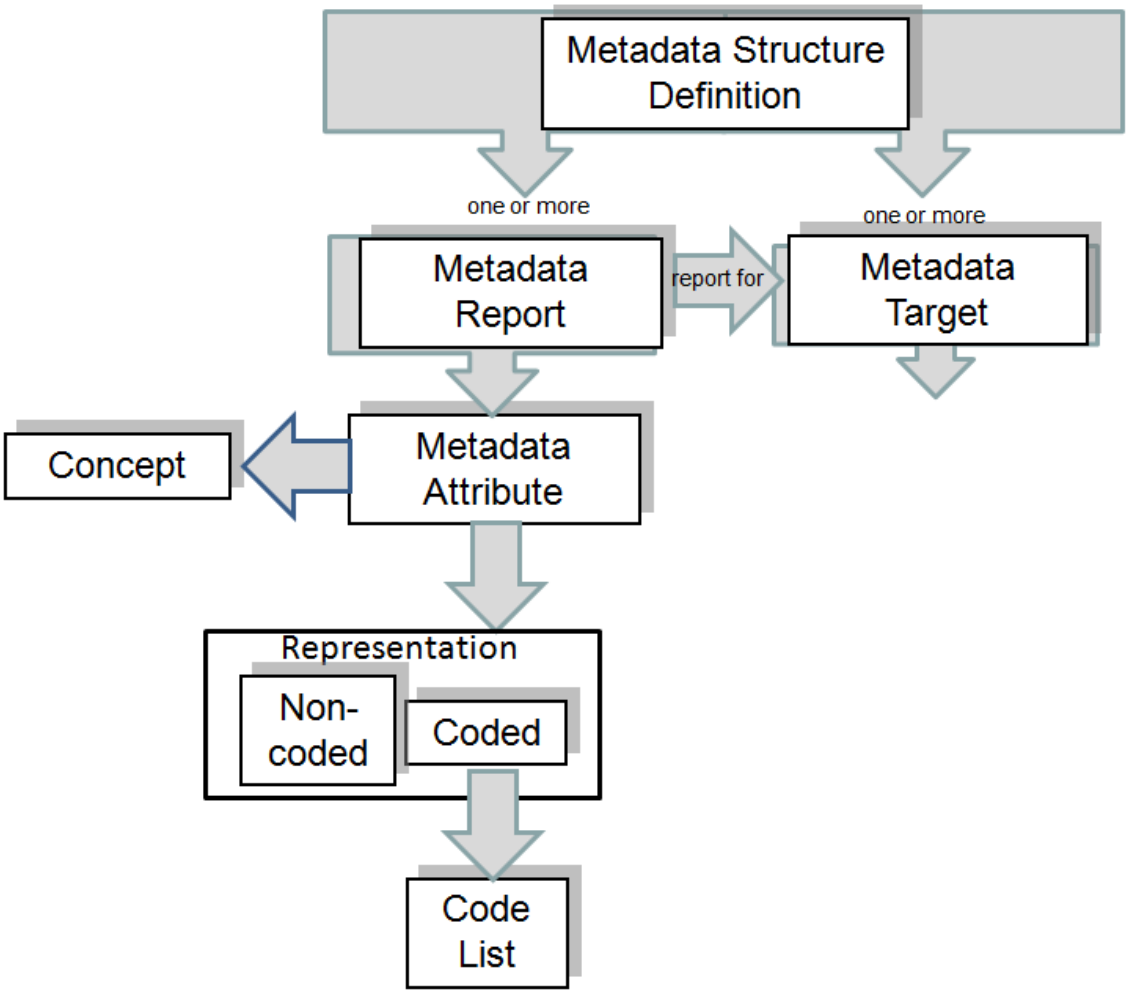


Figure 6: Schematic of the Report Structure in a Metadata Structure Definition

The metadata structure defines one or more report structures which define the content of its metadata reports. Each report structure is assigned a unique identifier within the metadata structure. The report structure defines the metadata attributes which make up its content and references the metadata targets from within the metadata structure that define where the report can be attached. Since a metadata set can only contain metadata reports for a single metadata structure, one must consider whether having reports in the same metadata set would be useful. Note that all reports in a metadata set must be defined in the same metadata structure.

4.4.1.2.1 Metadata Attribute

A metadata attribute is component of the content of a metadata report. Similar to a data attribute in a data structure, the metadata attribute takes its semantic from a concept. This concept serves to define the meaning of the information contained in the report. As

with data structure components, it is important to make use of common concepts whenever possible, as this makes the metadata relatable to other metadata reports.

The content of a metadata attribute can be a value and/or other metadata attributes. The value of a metadata attribute can serve a number of purposes. First, as with a data attribute it can be an enumerated value from a SDMX code list. It can also be a non-coded value of any given data type. Where the metadata attribute value differs from that of a data attribute is that if the value is textual, it can be represented in parallel language values, and if necessary be structured using XHTML. It is not necessary that a metadata attribute has a value, as it might only serve to contain other metadata attributes for the purpose of organising metadata reports (in which case the metadata attribute is designated as “isPresentational” indicating that no value is expected to be reported for the metadata attribute in a report).l.

Metadata attributes are not reusable across various levels of a report structure. If an attribute is intended to occur at multiple levels, it must be redefined at each level. However, the identification of a metadata attribute is only required to be unique within the scope within which it is defined. Therefore, if the intention is to reuse a metadata attribute at different levels, it is recommended that the same identification be used to convey this intention. It should also be noted that within a scope, a metadata attribute can have cardinality (minimum and maximum number of occurrences). This allows metadata attributes to be repeated at various levels, as well as giving the report structure the ability to enforce content requirements.

4.4.2 Metadata Sets

A metadata set is a collection of metadata reports from the same metadata structure definition. Each report is based on a report structure defined within the metadata structure on which the set is based. More than one report for a given report structure can exist within a set, so long as their targets are unique, which is to say that for any given target there should only be one instance of a report for a given report structure.

The manner in which a metadata report is related to an object is through the specification of its target. The metadata structure defines the possible types of targets for a given report structure, and an instance of a report uses one of these target types to identify the actual target of the metadata. The target is essentially a collection of references to data or structural metadata and possibly a period to which the report applies. It is assumed that any system processing metadata reports will be able to resolve these references, or perhaps more appropriately, any system working with data or structural metadata will be able to process the related metadata reports.

The actual content of the report is always contained in an attribute set. This attribute set is the collection of metadata attributes for which values are provided. As with data, the key to a useful metadata report is understanding what is being reported. This comes down to effective concept usage. The content of any report is essentially made up of values reported against concepts. In order for systems to understand the meaning of the metadata, they must understand the concepts.

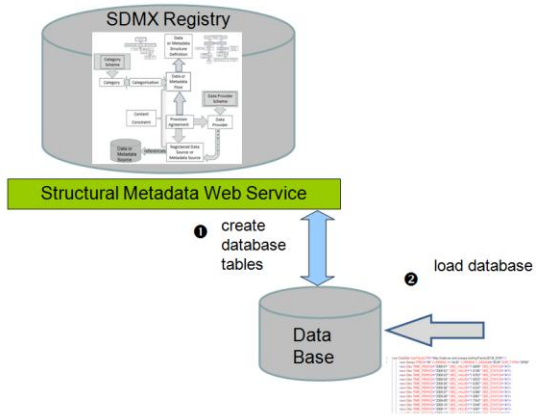
With the metadata structure definition, the content of any report can be evaluated for completeness and validity. From the metadata structure, one can determine if all necessary metadata attributes are present for a given report structure and if the values assigned to them are allowable based on the attribute definition.

In the Eurostat DemographyEurostat Demography metadata samples, one can see the similarities between the generic and the structure specific messages. The content of the reported metadata does not change with the format used. The structure specific metadata simply provides more validation of the content.

5 Data Bases and SDMX

5.1 Scope of this Chapter

Use Case	Relevant
Data Reporting	☑
Load data into a database	☑
Report reference metadata	
Load metadata into a database (often called a metadata repository)	
Report data by means of the "pull" method	
Database Administration (automatic generation of database tables)	☑
Enable database to be compatible with SDMX Web Services	☑
Data Discovery	
Data and Metadata Query and Visualisation	



5.2 Introduction

So far this guide has concentrated on the various SDMX constructs – structural metadata, data and metadata sets – which enable applications to understand and process the data and reference metadata. The next few chapters explain how these are used in practical situations. This starts with the database: the database holds the data that are to be reported, collected, or disseminated. The database is at the kernel of any statistical system.

This Chapter explains ways that you can read and write SDMX formatted data files to and from a database, and how you can process an SDMX REST query for data. It also describes how you can use a Data Structure Definition to create database tables.

5.3 Database and DSD Mapping

Note that for the data reporting use case it is probable that the coding system used in the database of the data reporter is not the same as that defined in the DSD (which is usually that used by the data collector), and the database column names are not the same as the Dimension and Attribute Ids in the DSD used for data reporting. Even for

the data collector it may be that the coding system in DSD is not the same as that used in the collector's database. In these cases there is a need for a generic mapping mechanism. Depending on the chosen method for reading and writing data to/from the database, this mapping can be performed from within the database application or external to it (i.e. before passing the data to the database application or after the database application has written the data).

<this is taken directly from the current user guide>

This mechanism provides a generic metadata-driven way for the database application to map the local structural metadata present in the data providers' system and those provided with the DSD.

In order to better explain this process the following real life example will be used.
The Eurostat SODI (SDMX Open Data Interchange) project deals with certain of the set of STS (short-term statistics) indicators defined by EU statistical legislation. This project implements a data-sharing architecture using the pull mode (although the push mode is also supported). Generally the majority of the involved data producers have their data already stored in a database and described using different local structural metadata. This is the case for the Italian National Institute of Statistics (ISTAT), which disseminates those data through its short-term statistical databank ConIstat[1].
Inside ConIstat, data are stored in a database using local structure metadata. A simplified snapshot of the database schema is provided in **Error! Reference source not found.**.

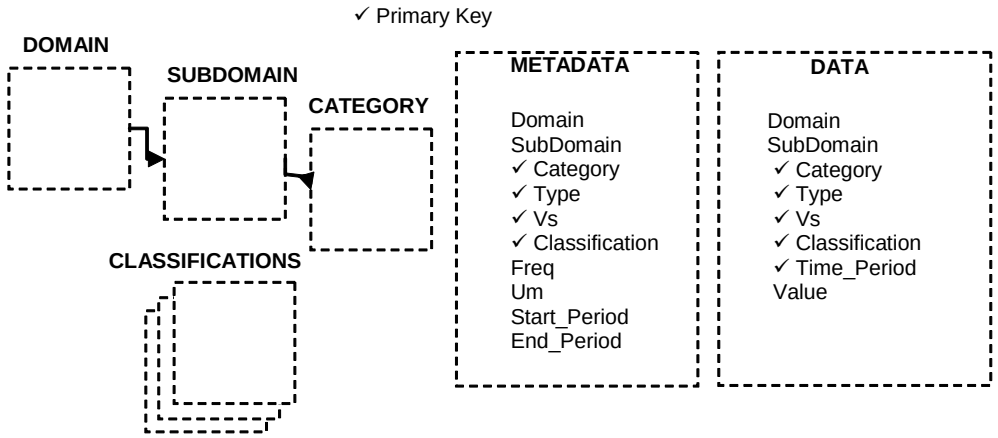


Figure 7: Database schema for ConIstat

The schema is mainly based on two database tables: METADATA and DATA. The others tables can be considered as lookup tables useful to store code lists. Moreover two tables, respectively named DOMAIN and SUBDOMAIN, allow categorizing data in statistical subject-matter domains.

[1] ConIstat: http://con.istat.it/

The main concepts used in order to describe each time-series are the following:
Category: short term statistical indicator
Type: adjustment indicator
Vs: stock/flow
Classification: NACE, SEC95, other classifications
Freq: frequency
Um: Unit of measure + Unit multiply + Base year
Start_Period: start period of the time-series
End_Period: last available period of the time-series
Each time-series is identified through a row in the METADATA table, and each field in that table has a correspondence in a particular lookup table representing a code list.
So the time-series *Monthly, neither seasonally or working day adjusted, Production in industry index base 2000, Mining and quarrying* is described in the following way:
Category: 11 (index of industrial production)
Type: g (neither seasonally or working day adjusted)
Vs: R (flow)
Classification: C (Mining and quarrying)
Freq: 12 (monthly)
Um: PE (index number - base 2000)
Start_Period: 01_1990
End_Period: 08_2008
The mapping process can be achieved by storing the resulting information in a special repository outside or inside the native database. In the case of the reported example it was chosen to use a repository inside the native database but without changing anything in the original tables. For this purpose, the following tables were added to the already existing schema:

- STS_METADATA: used to describe STS time-series (in order to describe other domains it would be necessary to add other tables, for example ESA_METADATA for National Accounts and so on);

- Some lookup tables useful to store within the local database some SDMX artefacts from the related DSD (for example: labels or even descriptions for concepts, code lists and dataflows)

The table STS_METADATA represents the place where the mapping process stores the mapping information. In fact, it inherits the base structure from METADATA, and some fields were added in order to cover all the concepts expressed in the SDMX DSD.
The resulting database schema after adding the new tables useful for the mapping process is shown in Figure 5.3.8.

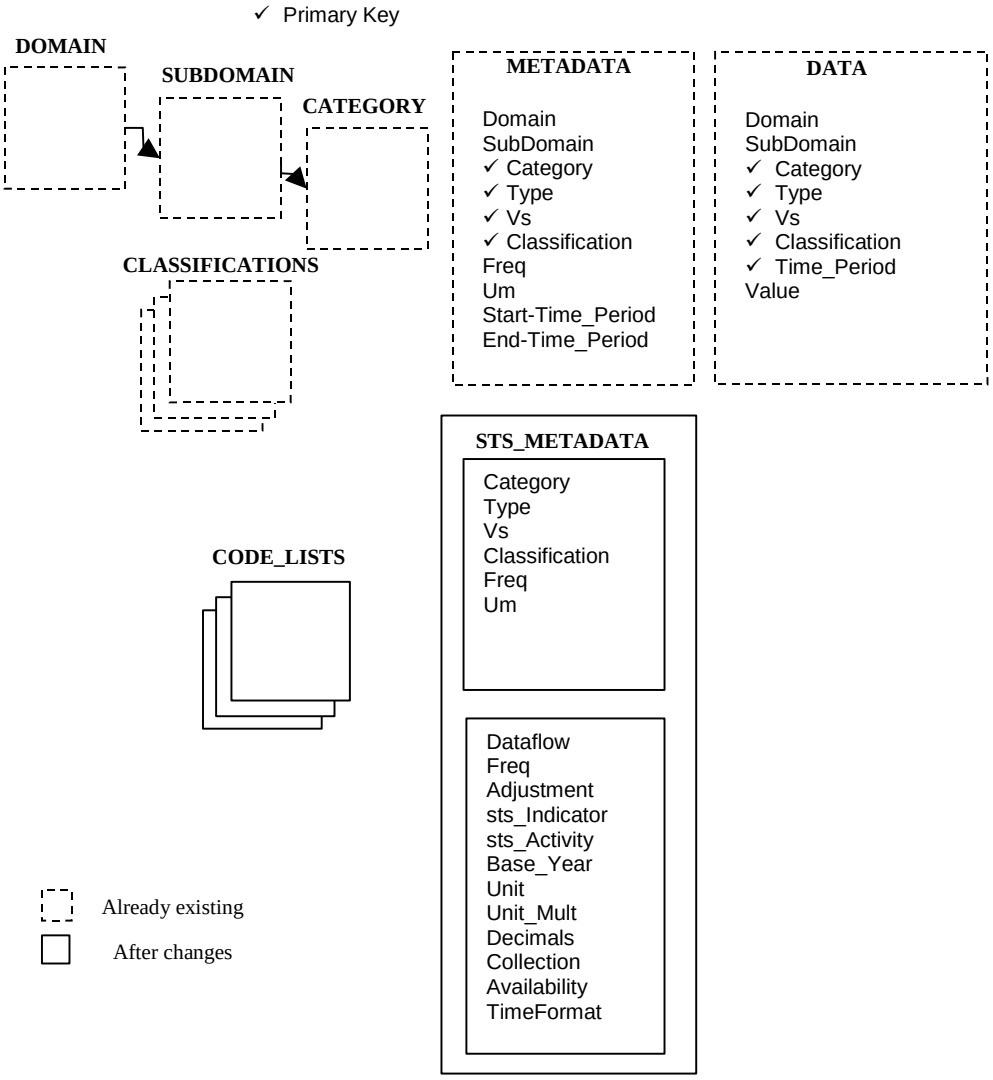


Figure 5.3.8: Database schema with additional tables for mapping

In order to perform the mapping process correctly, it is necessary to consider different types of mapping: mapping of concepts and mapping of codes[2].

5.3.1.1 Mapping of concepts

The first step is to identify all the statistical concepts involved in the exercise. The following circumstances can occur:

1) one concept in the DSD can be linked up with a single local concept. A typical example is the *measured value* in the data provider database that corresponds to the *Primary measure* in the STS DSD used for SODI;

[2] For further explanations of the usage of concepts and codes, see chapters XXXXXX.

2) one local concept must be linked up with two or more concepts in the DSD. For example in the local concept named Um there is an element as follows: “one million of Euro”. In the related STS DSD it corresponds to two concepts: Unit (Euro) and Unit multiple (one million);
3) one concept in the DSD is not directly linked up with any local concept. This could be the case of the concept “Reference area”, in fact that concept is generally not used in a National Organisation because it is the default (Italy);
4) one concept in DSD is linked up with two or more local concepts. For example the DSD concept “Adjustment” has no 1-to-1 correspondence with any single local concept; it is split into two different concepts “DAYADJ” (calendar adjusted) and “SEASADJ” (adjusted for periodical variations during the measurement period), which each has a Boolean value (true/false).

5.3.1.2 Mapping of codes

The second step is the mapping of the codes. Often a concept within a DSD can assume a code enumerated in a code list or a free value. The same thing can happen for a local concept. Assuming the concept used in the DSD and the local concept, used in the data provider's database, are both described using code lists, it may be possible to map one code in the first code list with a code in the second code list. The following example shows two such code lists:

- code list associated with the frequency local concept

CODE	DESCRIPTION
1	Annual
12	Monthly
365	Daily
4	Quarterly
52	Weekly

- code list associated with the frequency concept in the code list used by the STS DSD

CODE	DESCRIPTION
A	Annual
M	Monthly
D	Daily
Q	Quarterly
W	Weekly
H	Half-yearly
B	Business

The mapping process will produce the following result:

DSD CODE	Local CODE	DESCRIPTION
A	1	Annual
M	12	Monthly
D	365	Daily
Q	4	Quarterly
W	52	Weekly
H		Half-yearly
B		Business

Often the map processing can be helped by some rules. For example, consider the CL_STS_ACTIVITY code list and the NACE Rev 1.1 classification. The rule is: remove all dots from the NACE code and add as many zeros as necessary in order to reach four digits. Then add the prefix N1, or NS in case of special codes.
After applying the above steps, the result of the mapping process in ConIstat can be set out as in Table 5.3.1, in which columns represent both DSD concepts and local concepts, while rows represent a combination of their codes. The scheme shown here reflects the way in which the mapping tables are set up at ISTAT, which was chosen for performance reasons; the mapping table could be organised in other ways.

Table 5.3.1: Mapping result example

CATEGORY	TYPE	CLASSIFICATION	FREQ	UM	DATAFLOW	STS_INDICATOR	STS_ACTIVITY	UNIT	BASE_YEAR	ADJUSTMENT	FREQUENCY
18	G	DL300	12	PE	SSTSIND_ORD_M	ORDT	N13000	PURE_NUMB	2000	N	M
18	G	DL31	12	PE	SSTSIND_ORD_M	ORDT	N13100	PURE_NUMB	2000	N	M
18	G	DL311	12	PE	SSTSIND_ORD_M	ORDT	N13110	PURE_NUMB	2000	N	M

For example:

- the concept named CATEGORY that assumes the code 18 (Index of total orders), from the related local code list, is mapped with the concept named STS_INDICATOR that in the STS code list is represented by the code ORDT;

- the concept named TYPE that assumes the code G (neither seasonally or working day adjusted), from the related local code list, is mapped with the concept named ADJUSTMENT that in the STS code list is represented by the code N;

- the concept named FREQ that assumes the code 12 (Monthly), from the related local code list, is mapped with the concept named FREQUENCY that in the STS code list is represented by the code M;

- the concept named UM that assumes the code PE (index base=2000), from the related local code list, is mapped with the two concepts: UNIT that in the SDMX code list is represented by the code PURE_NUMB and BASE_YEAR that in the STS code list is represented by the code 2000;

- the concept named CLASSIFICATION that assumes the code DL300 (Manufacture of office machinery and computers), from the related local code list, is mapped with the concept STS_ACTIVITY that in the STS code list is represented by the code N13000.

5.4 Reading and Writing SDMX to and from a Database

5.4.1 Mechanisms

Database applications may need to read or write different versions of SDMX data. This can, of course, be solved in many ways. Two basic ways are:

- the application reads or writes a specific format of data (which could be a specific format of SDMX) which is pre or post processed by a transformation tool that transforms the input/output to the desired format
- the desired output format is read or written directly to/from the application

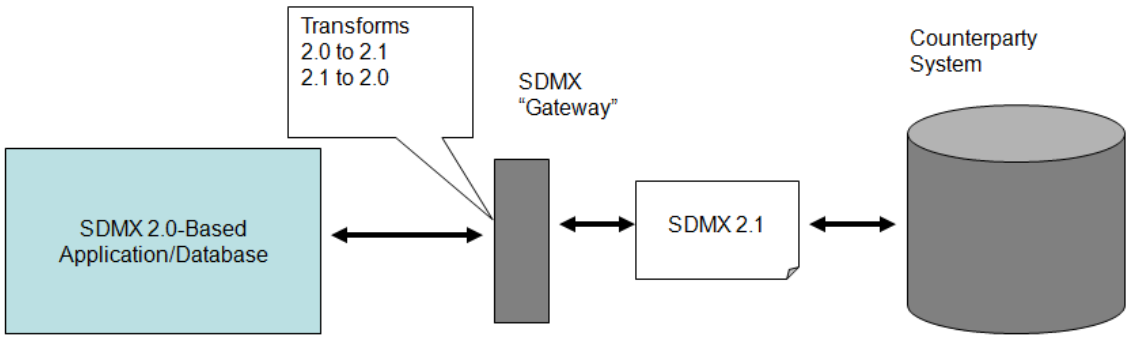


Figure 9: External Transformation of SDMX Formats

If the application reads/writes a specific form of SDMX, then there are transformation tools readily available that will convert the data to/from different SDMX formats. Some may rely on reading the entire file into memory to undertake the transformation and so this may be a limiting factor on the practicality of this approach (certainly if performance is an issue). Whist a separate transformation process is a simple approach which is well understood (and as such is not discussed any more in this Chapter), it does mean reading or writing the data twice.

The mechanism discussed here is concerned with what may seem to be a more complex approach, but it has two advantages over the separate transformation approach:

1. The database application need have no knowledge of the SDMX data set syntax.
2. The data set is read or written only once, and can be streamed directly to/from the database application and the SDMX read/write application which means there is no size limiting factor.

5.4.2 SDMX Information Model

The SDMX Information Model for data recognizes the following fundamental structures for data:

- Group Key - comprising Dimensions
- Series Key - comprising Dimensions
- Observation – possibly including time
- Attribute

It is practical to read or write an SDMX data file without the need for the database read/write application to know anything about SDMX. This can be done in two main ways:

- By means of a data reader and data writer software component
- By means of a data and structure mapping tool

5.5 Data Reader and Data Writer

5.5.1 Schematic

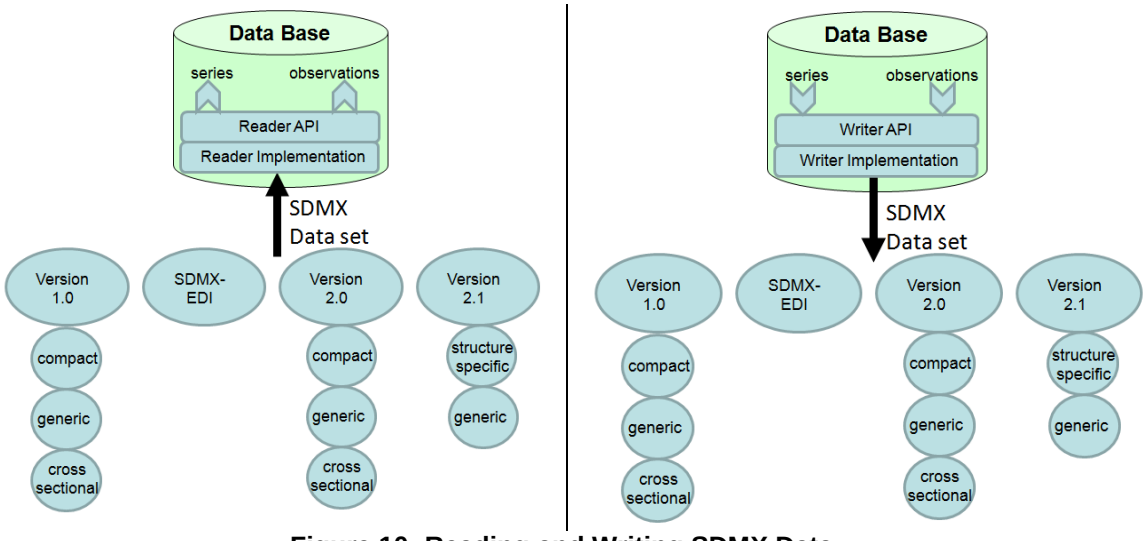


Figure 10: Reading and Writing SDMX Data

5.5.2 SDMX Data Writer Interface

An example of interfaces that will enable an application to read and write any type of SDMX data file is shown in Annex 3.

5.5.3 Data Mapping Tool

<it is suggested that Eurostat add some content here>

5.6 Data Base Table Structure

The Data Structure Definition can be used to create a relational table structure in a database. The simplest type of structure is shown below:

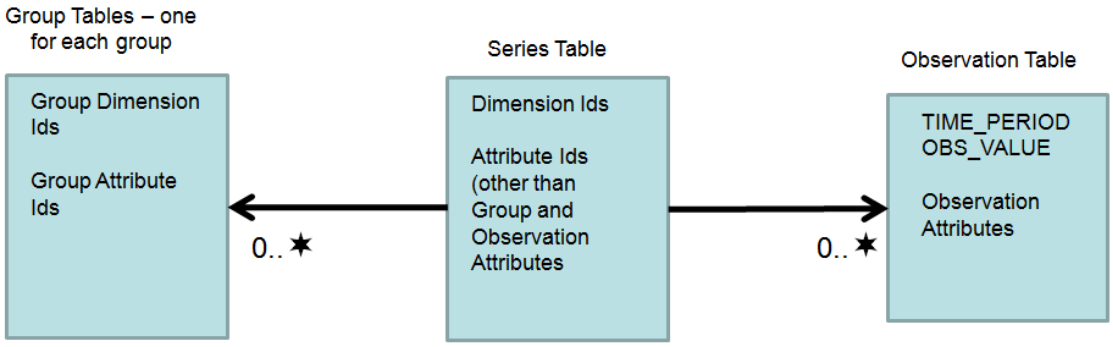


Figure 11: Schematic of a Database Schema Derived from a DSD

With this structure it is easy to implement the SDMX REST web services for data. Note that this type of structure does not store any of the structural metadata and this must be made available from another web service, such as an SDMX Registry, or from additional (structural metadata) structures in the database

The structural metadata from the ECB-EXR1 Data Structure Definition that is relevant to the database is shown below.

Dimensions

Id	Concept
FREQ	Frequency
CURRENCY	Currency
CURRENCY_DENOM	Currency denominator
EXR_TYPE	Exchange rate type
EXR_SUFFIX	Series variation - EXR context
TIME_PERIOD	Time period or range

Attributes

Id	Concept	Codelist	Attribute Relationship
COMPILATION	Compilation	-	Group (GROUP)
COVERAGE	Coverage	-	Group (GROUP)
DECIMALS	Decimals	Decimals code list(1.0)	Group (GROUP)
NAT_TITLE	National language title	-	Group (GROUP)
SOURCE_AGENCY	Source agency	Organisation code list(1.0)	Group (GROUP)
SOURCE_PUB	Publication source	-	Group (GROUP)
TITLE	Title	-	Group (GROUP)
TITLE_COMPL	Title complement	-	Group (GROUP)
UNIT	Unit	Unit code list(1.0)	Group (GROUP)
UNIT_MULT	Unit multiplier	Unit multiplier code list(1.0)	Group (GROUP)
BREAKS	Breaks	-	Dimension Group
COLLECTION	Collection indicator	Collection indicator code list(1.0)	Dimension Group
DOM_SER_IDS	Domestic series ids	-	Dimension Group
PUBL_ECB	Source publication (ECB only)	-	Dimension Group
PUBL_MU	Source publication (Euro area only	-	Dimension Group
PUBL_PUBLIC	Source publication (public)	-	Dimension Group
TIME_FORMAT	Time format code	-	Dimension Group
UNIT_INDEX_BASE	Unit index base	-	Dimension Group
OBS_COM	Observation comment	-	Observation
OBS_CONF	Observation confidentiality	Observation confidentiality code list(1.0)	Observation
OBS_PRE_BREAK	Pre-break observation value	-	Observation
OBS_STATUS	Observation status	Observation status code list(1.0)	Observation

Groups

Group Ids
Group

Dimension Reference
CURRENCY
CURRENCY_DENOM
EXR_TYPE
EXR_SUFFIX

Primary Measure

Id	Concept Agency
OBS_VALUE	ECB

An example set of database tables from the ECB-EXR1 Data Structure Definition is shown below.

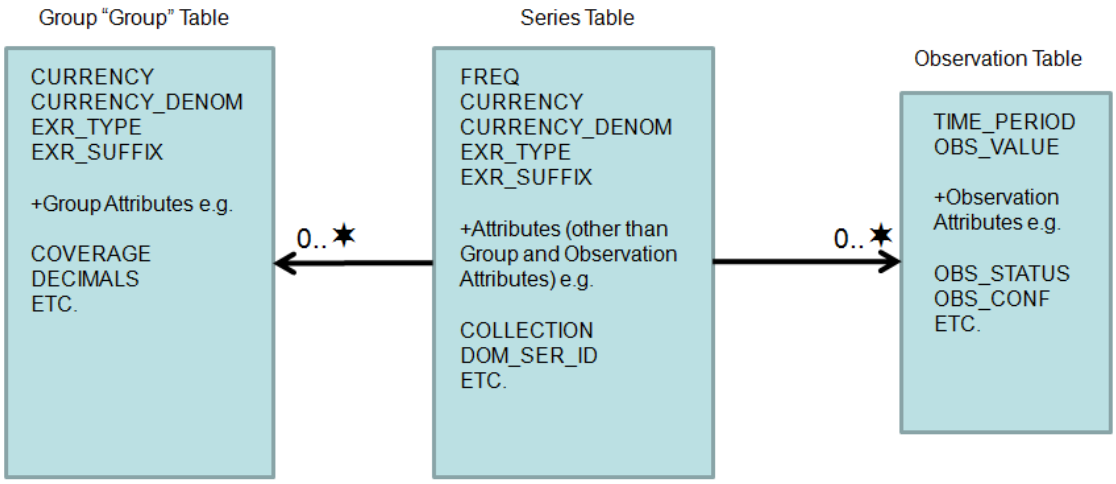


Figure 12: Schematic of a Database Schema Derived from the ECB_EXR1 DSD

5.7 Processing Queries

The SDMX Information Model for data recognizes the following constructs that are relevant to a database system for reading or writing SDMX, and for processing SDMX REST data queries:

For structure

- Dimension
- Time Dimension
- Observation
- Data Attribute
- Group

For data

- Series Key
- Observation

A database can be made "SDMX Web Services enabled" in a similar way to the Data Reader and Data Writer described in

Annex 4 – Data Reader and Data Writer Functions. This is shown schematically in the diagram below.

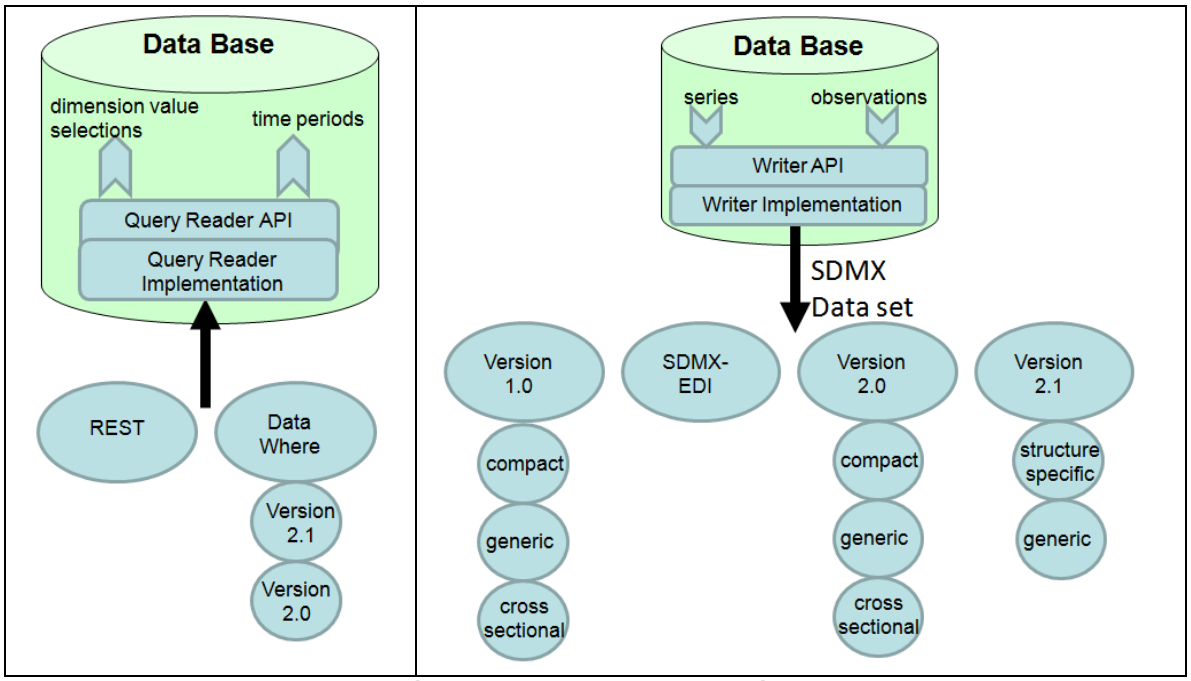


Figure 13: SDMX Query Reader

An example interface for the Query Reader API is shown in Annex 3. As for the Data Reader and Data Writer it is the interface that is the important asset here, and this is structured using the constructs of the SDMX Information Model, which are implemented in some way by any of the actual query formats (see Annex 3), the database application need not be concerned with this.

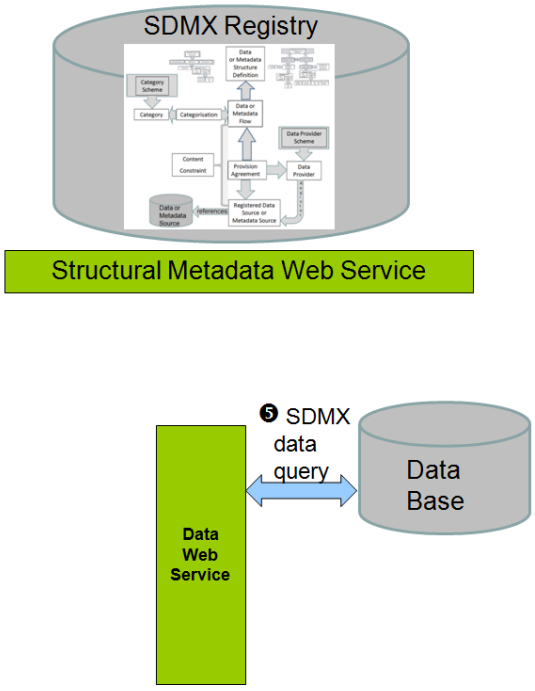
It will be seen from Chapter 6, that this interface presents the content of the SDMX REST data query in a way that is easy for the database application to process without the need to know the syntax of the REST query (or any of the other possible query formats).

The database result set is output to SDMX using the relevant implementation of the Data Writer Interface. This actual implementation that the database application uses will depend upon the query response format requested by the user. However, again this technicality is hidden from the database application and it is concerned solely with the methods of the Data Writer interface.

6 Data and Structure Query

6.1 Scope of this Chapter

Use Case	Relevant
Data Reporting	☑
Load data into a database	
Report reference metadata	
Load metadata into a database (often called a metadata repository)	
Report data by means of the "pull" method	
Database Administration (automatic generation of database tables)	
Enable database to be compatible with SDMX Web Services	☑
Data Discovery	☑
Data and Metadata Query and Visualisation	☑



This chapter describes how SDMX opens data by standardising the means in which it can be discovered. By using SDMX, existing data can easily be exposed regardless of its current storage. The **Error! Reference source not found.** chapter has already detailed how existing data stores can be related to SDMX. This chapter will describe the further considerations of exposing such a data store with SDMX query interfaces. While reading this section, it will be useful to reference Annex 5 - Query Samples for detailed examples of both the REST and SOAP syntaxes.

6.2 Query Types

Part 7 of the SDMX standard defines the web services standards for querying data and metadata. There are two standard protocols for querying; REST and SOAP. Because the specific query interfaces are standardised in each of these protocols, a user can access any SDMX enable data source without knowing any details of it, outside of its existence. All of the information needed to query for the data can be found using standard SDMX query mechanisms.

The full details of the REST and SOAP protocols are described in detail in Part 7 of the SDMX standards documentation. However, the following section will briefly discuss the

differences between the protocols and when it may be more appropriate to use one over the other.

6.2.1 REST and SOAP

The REST and SOAP protocols complement each other. They are effectively different means to the same end. Anything which is discoverable using SOAP is discoverable using REST. Both protocols share the same basic design principle, which is to have a predictable query mechanism which closely aligns to the SDMX information model. Therefore, if one understands the information model, one should be able to easily understand the query mechanisms. However, it should be noted that the focus of the REST protocol is simplicity. As such it only implements the core facets of the information model, which include identification and referencing. The SOAP query supports more detailed searches, allowing for a search based on all facets of the information model. For example, it is possible in SOAP to make a query based on the name of a structural metadata object, whereas REST does not support this.

6.3 Querying Structures

SDMX must always conform to a data structure definition, and as such the first step in exposing existing data as SDMX it to define the data structures for the available data. The Defining Data Structures chapter discusses the specific considerations of creating the data structure, but there are other considerations that are more specific to exposing existing data through SDMX queries.

When exposing data as SDMX it is important to consider how users might discover the data which is being sought. In most cases, users will first want to see which data structures are available from a given source and then get the specific data for a given structure. Therefore, the way in which these data structures are defined is very critical to making data easily discoverable.

One means in which users often find data is through the classification of the nature of data. For example, one might be seeking exchange rate data as detailed in the sample scenario in Annex 5 - Query Samples. In SDMX, this is achieved via categorisation of data sets, data structures, data flows, or provision agreements. A user will typically find the category which matches the type of data being sought, and then search for data structures which a categorised against this category. Therefore, it is often useful to categorise your data and structural metadata.

Annex 6: Worked Use Case shows data discovery using this “drill down” approach.

Another means in which users might discover data is through known concepts or code lists. This further emphasises the point made in Chapter 4, that standard concepts and code lists should be used whenever possible. For example, suppose a user seeking demographic data knows that they are looking for data which uses a life birth concept as a measure. The SDMX SOAP query allows for such queries. But this will only result in data being returned from the query if the standard concept is used.

6.4 Querying Data

Once a user has discovered the structural metadata on which data is based more specific queries can be made to get the specific data being sought. It is often true that a user may not wish to receive an entire data set, but rather data for particular keys within the data set. The structural metadata gives the user enough detail to inform how to query for data, but it does not make any guarantees about the presence of data. It is quite possible that the data source is a sparse cube - that is to say that there may not be data for every possible key permutation. To help users with this issue, there are two possible approaches.

The first approach is to describe the available content of the data source as a content constraint. A content constraint is structural metadata which details the key sets for which data is present. The constraint is attached to a particular data set, provision agreement, data flow, or data structure. Therefore, if the user application knows the structural metadata which the data is based on, it can query for the content constraint in order to determine which data is present.

The second approach is to support the data query mechanism which allows for only key values to be returned. In this approach, the user queries for data in the usual way, but specifies that the detail returned should contain no data or data attributes. This information is effectively equivalent to a content constraint but does not require the extra bit of structural metadata be maintained. These keys can be stored locally or cached by the application and processes to ensure that the user is presented with dimension value choices that will return data.

Finally, it is important to note that Part 7 of the specification is very clear in the fact that a query web service does not need to implement every feature of the various query protocols. It simply states that the query be accepted even if the result is an error stating that the given feature is not implemented. Therefore, the implementer need not be overwhelmed by all of the finer details of the queries.

7 Metadata Repository and Linking Data to Metadata

7.1 Scope of this Chapter

Use Case	Relevant
Data Reporting	
Load data into a database	
Report reference metadata	☑
Load metadata into a database (often called a metadata repository)	☑
Report data by means of the "pull" method	
Database Administration (automatic generation of database tables)	☑
Enable database to be compatible with SDMX Web Services	
Data Discovery	
Data and Metadata Query and Visualisation	☑

The storage, retrieval and linkage to data of Reference Metadata (often known as "footnote" metadata) is becoming more and more important in the dissemination of statistics. All organizations that compile statistical data have reference metadata and it is often useful to data consumers for the data providers to disseminate this with the data. This can be done in a simple way by disseminating all such metadata regardless of whether it pertains to the data, or it can be done in a more intelligent way, and therefore more useful to the data consumer, by disseminating only that metadata which pertains to the data being viewed or downloaded.

This Chapter discusses the role of the metadata repository in the context of:

- its logical structure
- loading metadata
- retrieving metadata

7.2 Structure of the Metadata Repository

In SDMX all metadata is characterized as:

- Metadata Attribute
  - Attached to a (SDMX) object

Examples are shown below. The “Metadata Attribute” is taken from the SDMX Cross Domain Concepts and the “Attached to” is a typical object to which it would be attached in a data dissemination scenario.

Metadata Attribute	Description	Attached To
COMPILING_ORG	The organisation compiling the data being reported	Data Key – this can be a full or partial key. In thie specific example presented here it would probably be attached to the Reference Area dimension.
COVERAGE_SECTOR	Main economic or other sectors covered by the statistics	CODE
SOURCE_TYPE	Characteristics and components of the raw statistical data used for compiling statistical aggregates	DATAFLOW
UNIT_MEASURE		SERIES KEY
COMMENT	Supplementary descriptive text which can be attached to data or metadata	CONCEPT (if applicable to all uses of the Concept) or DIMENSION (if applicable to the use of a Concept in a specific DSD)
OBS_STATUS	Information on the quality of a value or an unusual or missing value	OBSERVATION

Therefore the conceptual structure of a Metadata Repository is:

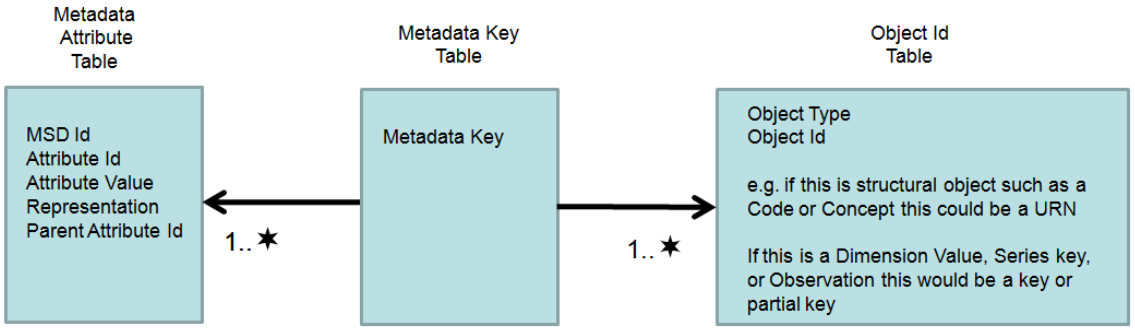


Figure 14: Conceptual Structure of a Metadata Repository

This is only a conceptual structure and is purposefully over-simplistic as it is drawn to show conceptually how such a metadata repository may work.

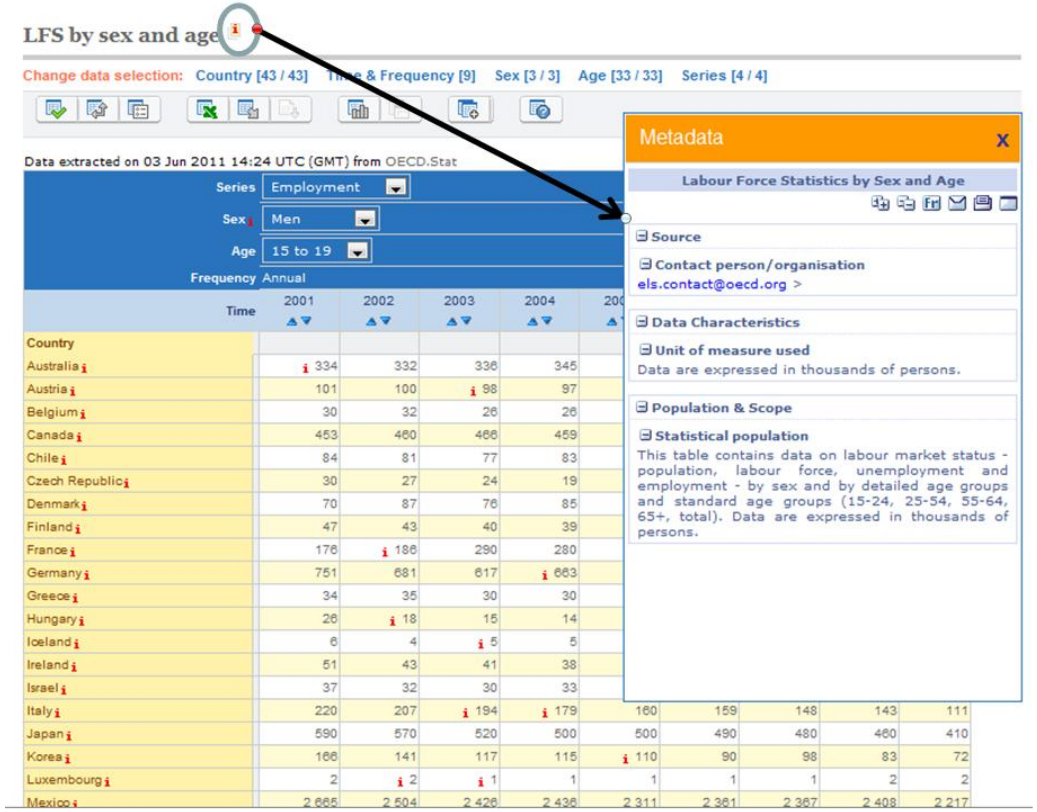
The Metadata Key is comprised of one or more Object Type/Object Id. For instance, for a Code or a Concept it would be the Id of the Code or Concept such as the URN. For a specific key or partial key there would be two Objects, the DSD and the Data Key. Clearly, this is a conceptual structure as it is probable that the Metadata Key would combine the value of all of the individual Object Ids, as this would be necessary in order to find metadata quickly.

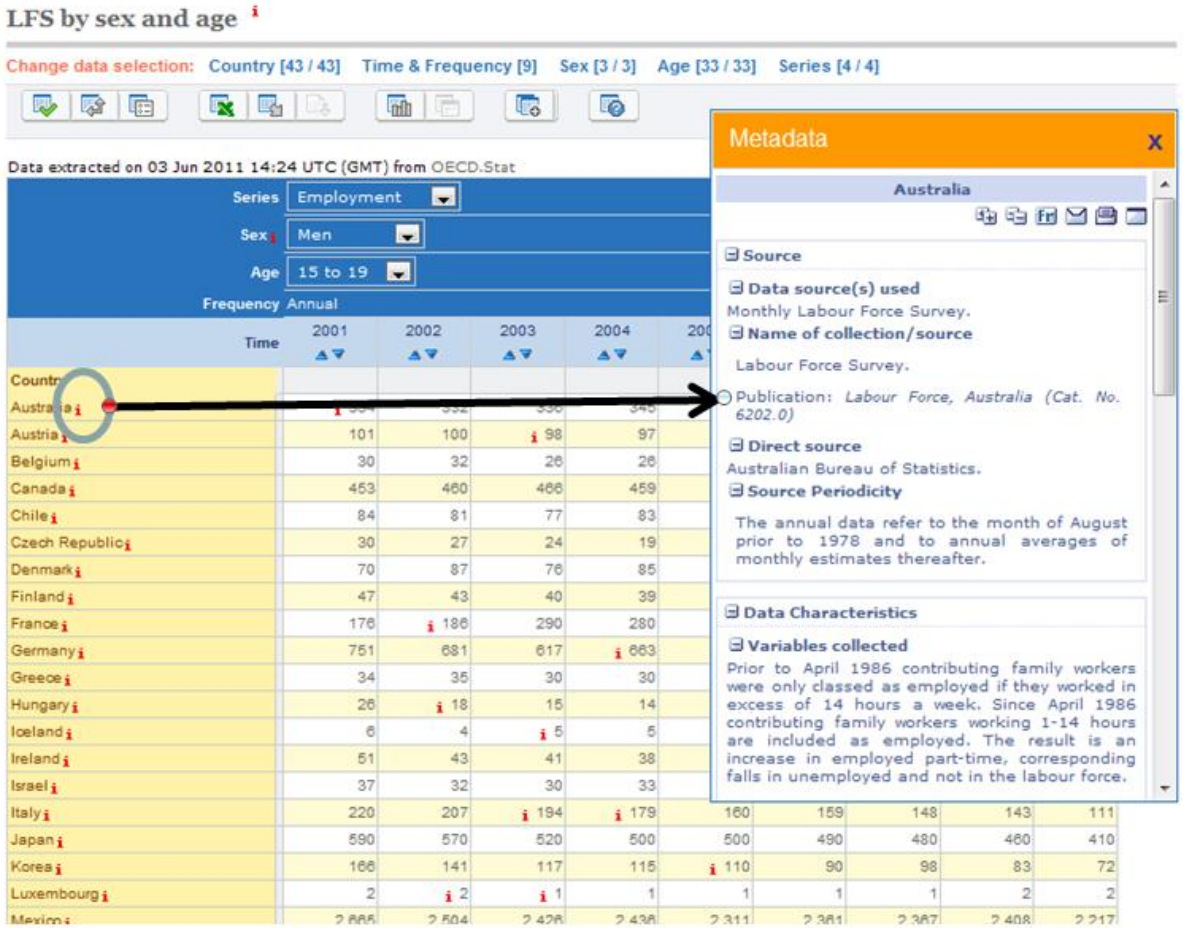
The Metadata Attribute Table may, of course, contain additional information depending on the actual requirements. For instance, it may not be necessary to reference the Metadata Structure Definition (MSD) providing the Concept name and Description are contained in the table. Also, it may be useful to add a “Short Name” to be used by applications such as web sites. Note that the metadata attribute structure can be hierarchical (i.e. a Metadata Attribute can have child Metadata Attributes). As each Metadata Attribute can have only one parent this structure is achieved by the Parent Attribute Id.

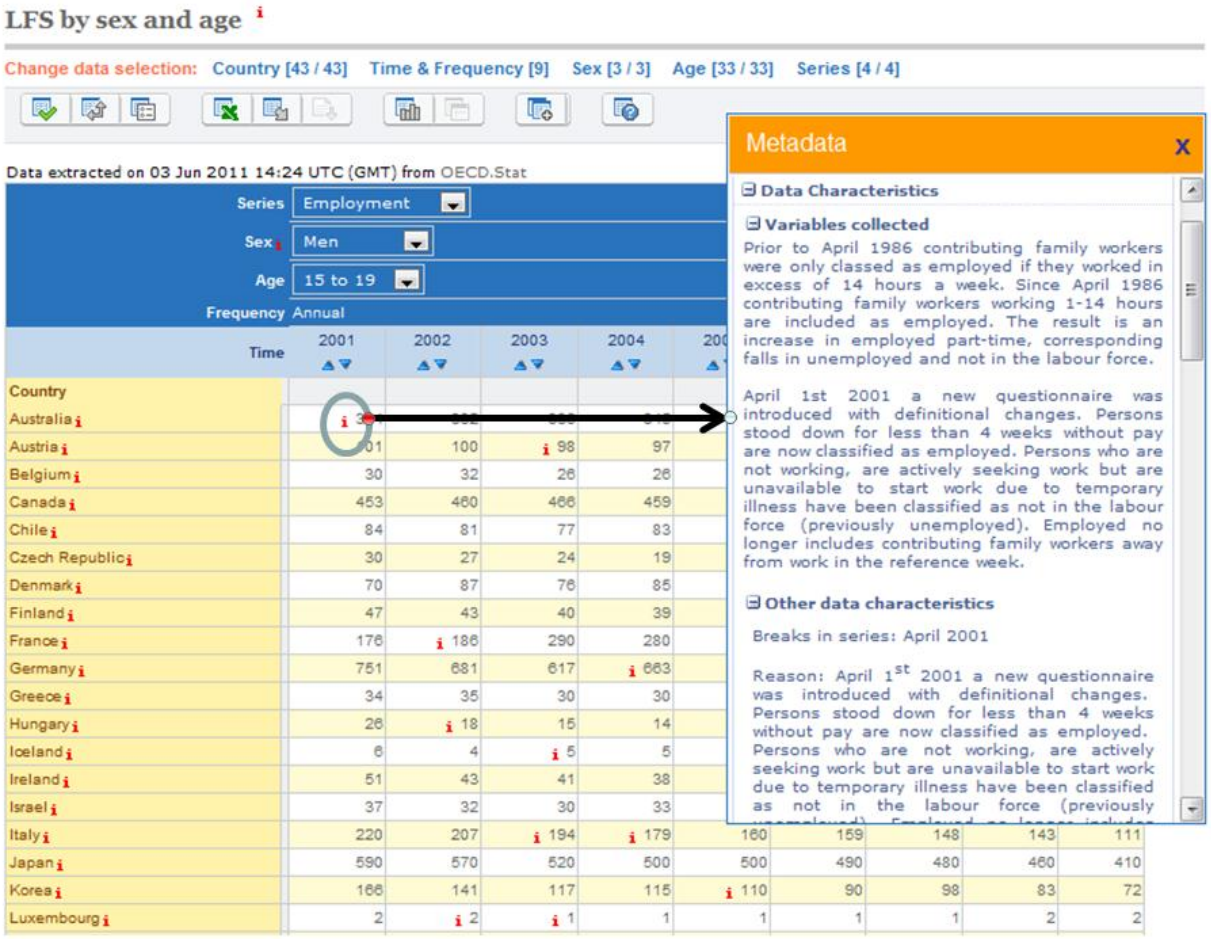
The way such a repository would support the linking of data to metadata is shown below.

7.3 Linking Data to Metadata

7.3.1 Examples

Selected Metadata	Key
 Source: OECD	**Object Type** = Dataflow Object Id = LFS_AGE_SEX **Metadata Attributes:** MSD Id = OECD_42 Id = CONTACT Value = els.contact@oecd.org Representation = text Parent Id = SOURCE MSD Id = OECD_42 Id = UNIT_MEASURE Value = data are expressed in thousands of persons Representation = text Parent Id = DATA_CHARACTERISTICS ETC.

Selected Metadata	Key
 Source: OECD	**Object Type** =Dataflow Object Id = LFS_AGE_SEX (note that this could be the DSD but the dataflow references the DSD) **Object Type** =Data Key Object Id = REF_AREA=AU **Metadata Attributes:** MSD Id = OECD_42 Id = SOURCE_TYPE Value = Monthly Labour Force Survey Representation = text Parent Id = SOURCE MSD Id = OECD_42 Id = SOURCE_NAME Value = Labour Force Survey Publication*: Labour Force, Australia (Cat. No. 6202.0)* Representation = HTML Parent Id = SOURCE MSD Id = OECD_42 Id = DIRECT_SOURCE Value = Australian Bureau of Statistics Representation = text Parent Id = SOURCE ETC.

Selected Metadata	Key
	**Object Type** =Dataflow Object Id = LFS_AGE_SEX **Object Type** =Data Key Object Id = REF_AREA=AU, SERIES=EMPLY, SEX=M, AGE=15_19, TIME_PERIOD=2001-M4 **Metadata Attributes:** MSD Id = OECD_42 Id = VARIABLES_COLLECTED Value = Prior to April 1986 contributing family workers were only classed as employed if they worked in excess of 14 hours a week. Since April 1986 contributing family workers working 1-14 hours are included as employed. The result is an increase in employed part-time, corresponding falls in unemployed and not in the labour force. April 1st 2001 a new questionnaire was introduced with definitional changes. ETC. Representation = text Parent Id = CHARACTERISTICS

Source: OECD

7.3.2 Metadata Structure Definition

7.3.2.1 MSD Visualised in a Tool

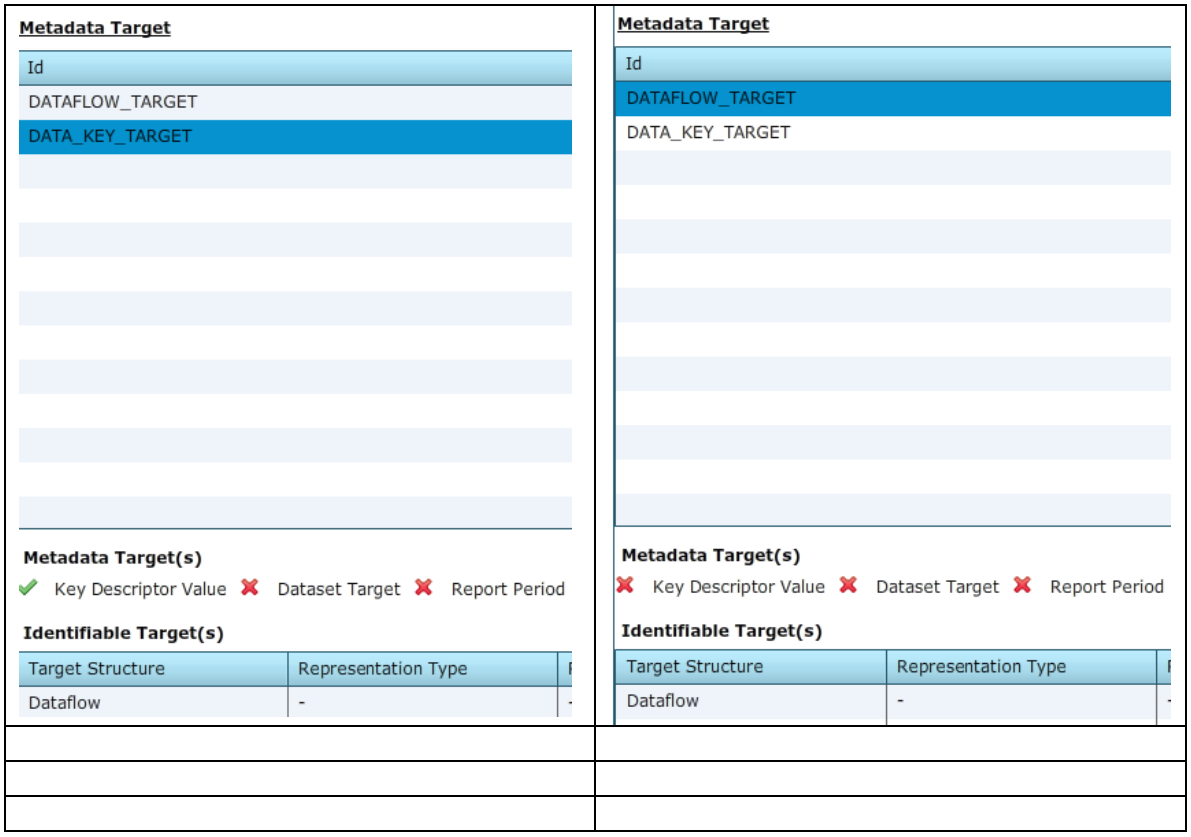


Figure 15: MSD - Metadata Targets

Notes
The DATA_KEY_TARGET comprises two components:

1. A **Key Descriptor** (dimension descriptor) – this specifies that a key (one or more dimension values) must be specified in the Metadata Set
2. An **Identifiable Object** which must be a Dataflow – this specifies that a Dataflow must be specified in the Metadata Set. If required, the valid list of Dataflows can be specified in an Item Scheme such as a Category Scheme.

The combination of these two components specifies a data key in the context of a specific Dataflow.

The DATAFLOW_TARGET comprises just one component:

1. An **Identifiable Object** which must be a Dataflow

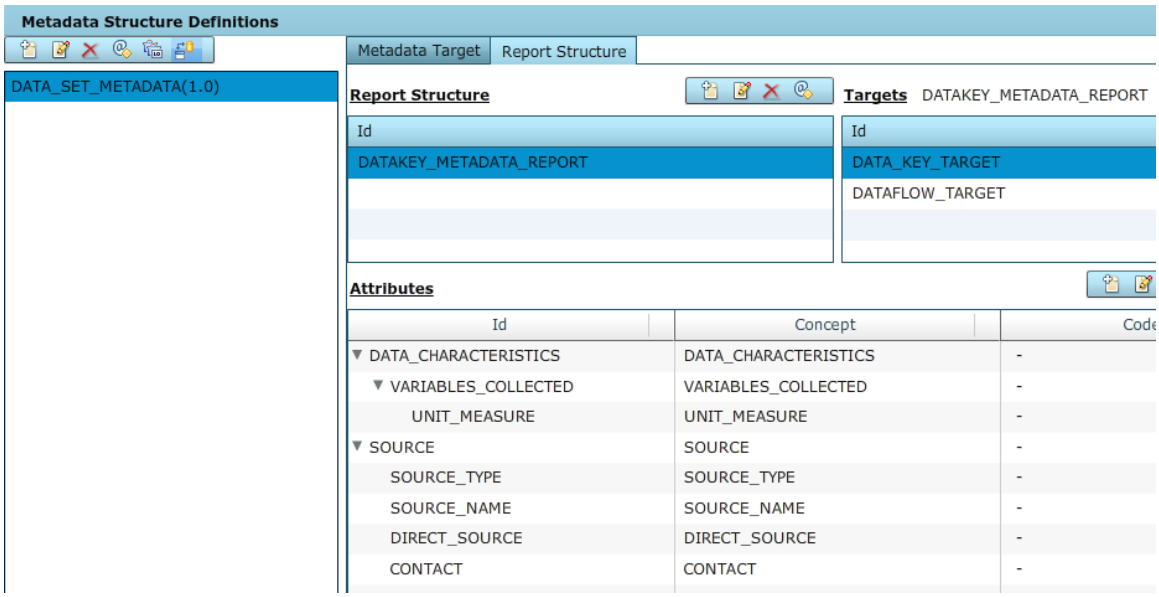


Figure 16: MSD - Report Structure

Notes

1. There is only one Report Structure. The same structure is used for both Metadata Targets (Dataflow and Data Key).
2. It is possible to have many Report Structure in an MSD, each referencing a different Metadata Target.

7.3.2.2 MSD SDMX-ML

```
<str:MetadataStructure id="OECD_42" agencyID="OECD" version="1.0">
  <com:Name xml:lang="en">MSD for data set metadata</com:Name>
  <str:MetadataStructureComponents>
  <str:MetadataTarget id="DATAFLOW_TARGET">
      <str:IdentifiableObjectTarget id="DATAFLOW" objectType="Dataflow">
        <str:LocalRepresentation>
          <str:TextFormat textType="IdentifiableReference"/>
        </str:LocalRepresentation>
      </str:IdentifiableObjectTarget>
      </str:MetadataTarget>
  <str:MetadataTarget id="DATA_KEY_TARGET">
      <str:KeyDescriptorValuesTarget>
        <str:LocalRepresentation>
          <str:TextFormat textType="KeyValues"/>
        </str:LocalRepresentation>
      </str:KeyDescriptorValuesTarget>
      <str:IdentifiableObjectTarget id="DATAFLOW" objectType="Dataflow">
        <str:LocalRepresentation>
          <str:TextFormat textType="IdentifiableReference"/>
        </str:LocalRepresentation>
      </str:IdentifiableObjectTarget>
      </str:MetadataTarget>
  <str:ReportStructure id="DATAKEY_METADATA_REPORT">
  </str:MetadataStructureComponents>
</str:MetadataStructure>
```

Figure 17: MSD - Metadata Targets

```
<str:MetadataStructure id="OECD_42" agencyID="OECD" version="1.0">
  <com:Name xml:lang="en">MSD for data set metadata</com:Name>
  <str:MetadataStructureComponents>
  <str:MetadataTarget id="DATAFLOW_TARGET">
  <str:MetadataTarget id="DATA_KEY_TARGET">
  <str:ReportStructure id="DATAKEY_METADATA_REPORT">
      <str:MetadataAttribute id="DATA_CHARACTERISTICS" minOccurs="0" maxOccurs="1">
        <str:ConceptIdentity>
          <Ref id="DATA_CHARACTERISTICS" maintainableParentID="METADATA_CONCEPTS"
          maintainableParentVersion="1.0" agencyID="OECD" package="conceptscheme" class="Concept"/>
        </str:ConceptIdentity>
        <str:LocalRepresentation>
          <str:TextFormat textType="String"/>
        </str:LocalRepresentation>
        <str:MetadataAttribute id="VARIABLES_COLLECTED" minOccurs="0" maxOccurs="1">
      </str:MetadataAttribute>
      <str:MetadataAttribute id="SOURCE"  minOccurs="0" maxOccurs="1">
        <str:ConceptIdentity>
        <str:LocalRepresentation>
        <str:MetadataAttribute id="SOURCE_TYPE"  minOccurs="0" maxOccurs="1">
        <str:MetadataAttribute id="SOURCE_NAME"  minOccurs="0" maxOccurs="1">
          <str:MetadataAttribute id="DIRECT_SOURCE"  minOccurs="0" maxOccurs="1">
        <str:MetadataAttribute id="CONTACT"  minOccurs="0" maxOccurs="1">
        <str:ConceptIdentity>
        <str:LocalRepresentation>
        </str:MetadataAttribute>
  </str:MetadataAttribute>
    <str:MetadataTarget>
        <Ref id="DATA_KEY_TARGET"></Ref>
      </str:MetadataTarget>
    <str:MetadataTarget>
        <Ref id="DATAFLOW_TARGET"></Ref>
      </str:MetadataTarget>
    </str:ReportStructure>
  </str:MetadataStructureComponents>
</str:MetadataStructure>
```

Figure 18: MSD - Metadata Report

Note that the reasons of brevity content of many of the Metadata Attributes has been collapsed e.g. each has a Concept identity and Local Representation.

7.3.3 Metadata Set

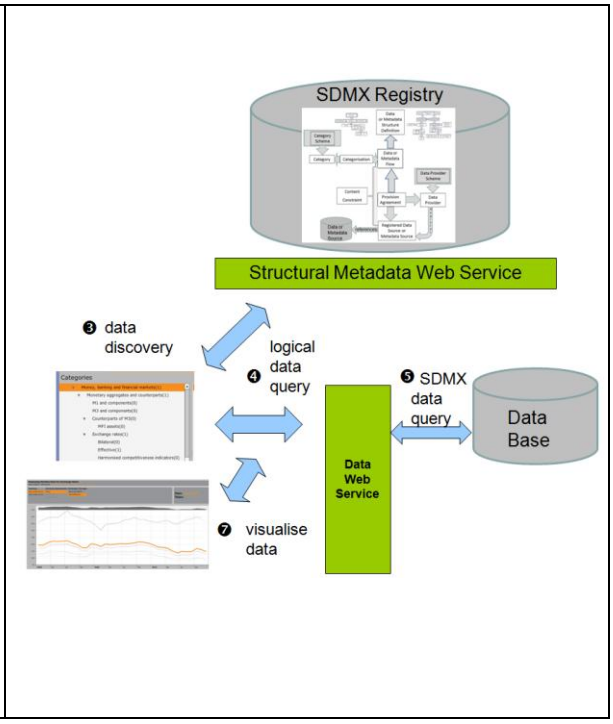
```
<message:MetadataSet structureRef="OECD_42">
  <metadata:Report id="REPORT1">
    <metadata:Target id="TARGET1">
      <metadata:ReferenceValue id="DATA_KEY_TARGET">
        <metadata:ObjectReference>
          <Ref class="Dataflow" id="LFS_AGE_SEX" package="datastructure" agencyID="OECD"/>
        </metadata:ObjectReference>
      </metadata:ReferenceValue>
    </metadata:Target>
    <metadata:AttributeSet>
      <metadata:ReportedAttribute id="DATA_CHARACTERISTICS">
      <metadata:AttributeSet>
        <metadata:ReportedAttribute id="CONTACT" value="els.contact@oecd.org"/>
        </metadata:AttributeSet>
    </metadata:ReportedAttribute>
      </metadata:AttributeSet>
  </metadata:Report>

  <metadata:Report id="REPORT2">
    <metadata:Target id="TARGET2">
      <metadata:ReferenceValue id="DATA_KEY_TARGET">
        <metadata:ObjectReference>
          <Ref class="Dataflow" id="LFS_AGE_SEX" package="datastructure" agencyID="OECD"/>
        </metadata:ObjectReference>
      </metadata:ReferenceValue>
      <metadata:ReferenceValue id="KEY_DESCRIPTOR">
        <metadata:DataKey>
          <common:KeyValue id="REF_AREA">
            <common:Value>AU</common:Value>
          </common:KeyValue>
        </metadata:DataKey>
      </metadata:ReferenceValue>
    </metadata:Target>
    <metadata:AttributeSet>
      <metadata:ReportedAttribute id="SOURCE">
      <metadata:AttributeSet>
        <metadata:ReportedAttribute id="SOURCE_TYPE" value="Monthly Labour Force Survey"/>
          <metadata:ReportedAttribute id="SOURCE_NAME"
          value="= Labour Force Survey Publication: Labour Force, Australia (Cat. No. 6202.0)"/>
          <metadata:ReportedAttribute id="DIRECT_SOURCE"
          value="Australian Bureau of Statistics"/>
          </metadata:AttributeSet>
    </metadata:ReportedAttribute>
    </metadata:AttributeSet>
  </metadata:Report>
```

```
<metadata:Report id="REPORT3">
  <metadata:Target id="TARGET3">
    <metadata:ReferenceValue id="DATA_KEY_TARGET">
      <metadata:ObjectReference>
        <Ref class="Dataflow" id="LFS_AGE_SEX" package="datastructure" agencyID="OECD"/>
      </metadata:ObjectReference>
    </metadata:ReferenceValue>
    <metadata:ReferenceValue id="KEY_DESCRIPTOR">
      <metadata:DataKey>
        <common:KeyValue id="REF_AREA">
          <common:Value>AU</common:Value>
        </common:KeyValue>
        <common:KeyValue id="SERIES">
          <common:Value>EMPLY</common:Value>
        </common:KeyValue>
        <common:KeyValue id="SEX">
          <common:Value>M</common:Value>
        </common:KeyValue>
          <common:KeyValue id="AGE">
          <common:Value>15_19</common:Value>
        </common:KeyValue>
      <common:KeyValue id="TIME_PERIOD">
          <common:Value>2001-M4</common:Value>
        </common:KeyValue>
      </metadata:DataKey>
    </metadata:ReferenceValue>
  </metadata:Target>
  <metadata:AttributeSet>
    <metadata:ReportedAttribute id="DATA_CHARACTERISTICS">
    <metadata:AttributeSet>
        <metadata:ReportedAttribute id="VARIABLES_COLLECTED" value="= Prior to April 1986 contributing family workers
        were only classed as employed if they worked in excess of 14 hours a week. Since April 1986 contributing family workers
        working 1-14 hours are included as employed. The result is an increase in employed part-time, corresponding falls in
        unemployed and not in the labour force.">
  </metadata:ReportedAttribute>
      </metadata:AttributeSet>
  </metadata:ReportedAttribute>
  </metadata:AttributeSet>
</metadata:Report>
```

8 SDMX Registry/Repository

8.1 Scope of this Chapter

Use Case	Relevant
Data Reporting	☑
Load data into a database	☑
Report reference metadata	☑
Load metadata into a database (often called a metadata repository)	☑
Report data by means of the "pull" method	☑
Database Administration (automatic generation of database tables)	☑
Enable database to be compatible with SDMX Web Services	
Data Discovery	☑
Data and Metadata Query and Visualisation	☑



In this guide it has been assumed that the structural metadata (e.g. DSD, MSD, Code List, Concept Scheme) is available to processing applications such as a data extract function of a database or a data visualization function of a website. For many SDMX processing applications these structural metadata must be made available "on demand".

This Chapter explains the role and functions of an SDMX Registry and how the Registry is used both as a storage, retrieval, and maintenance repository for SDMX structural metadata, and as a mechanism for discovering data and reference metadata including the automated "pull" scenario of an automated data reporting system .

8.2 The Need for an SDMX Registry

First, it should be stated that it is not necessary to operate or have access to an SDMX Registry in order to use SDMX. Many use cases do, however, require access to structural metadata but these can be retrieved from any web service that can respond to a query for structural metadata (either the REST query or the "Structure Where" query) - this service does not need to be an SDMX Registry.

Whilst both an SDMX Registry and a non-Registry web service use exactly the same query mechanism, a Registry differs from non-Registry web service in three important areas:

1. The Registry offers a maintenance service for structural metadata – this is the "Repository" function of the Registry.
2. The Registry offers a data and metadata Registration service
3. The Registry offers a subscription and notification service

The functions of these services are covered in this Chapter.

Software tools that support SDMX Registry services are becoming more functional and have reached a point where it is extremely easy to install and operate an SDMX Registry. This can be used to support a community of users or be used solely within a single organization in order to have a central repository for SDMX structural metadata which can be maintained and retrieved.

8.3 Objective of a Registry

The objective of the SDMX registry/repository is, in broad terms, to allow organisations to publish statistical data and reference metadata in known formats such that interested third parties can discover these data and interpret them accurately and correctly. The mechanism for doing this is twofold:

1. To maintain and publish structural metadata that describes the structure and valid content of data and reference metadata sources such as databases, metadata repositories, data sets, metadata sets. This structural metadata enables software applications to understand and to interpret the data and reference metadata in these sources.

2. To enable applications, organisations, and individuals to share and to discover data and reference metadata. This facilitates data and reference metadata dissemination by implementing the data sharing vision of SDMX.

8.4 SDMX Registry/Repository Architecture

8.4.1 Architectural Schematic

The architecture of the SDMX registry/repository is a layered architecture that is founded by a structural metadata repository which supports a provisioning metadata repository which supports the registry services. These are all supported by the SDMX-ML schemas. Applications can be built on top of these services which support the reporting, storage, retrieval, and dissemination aspects of the statistical lifecycle as well as the maintenance of the structural metadata required to drive these applications.

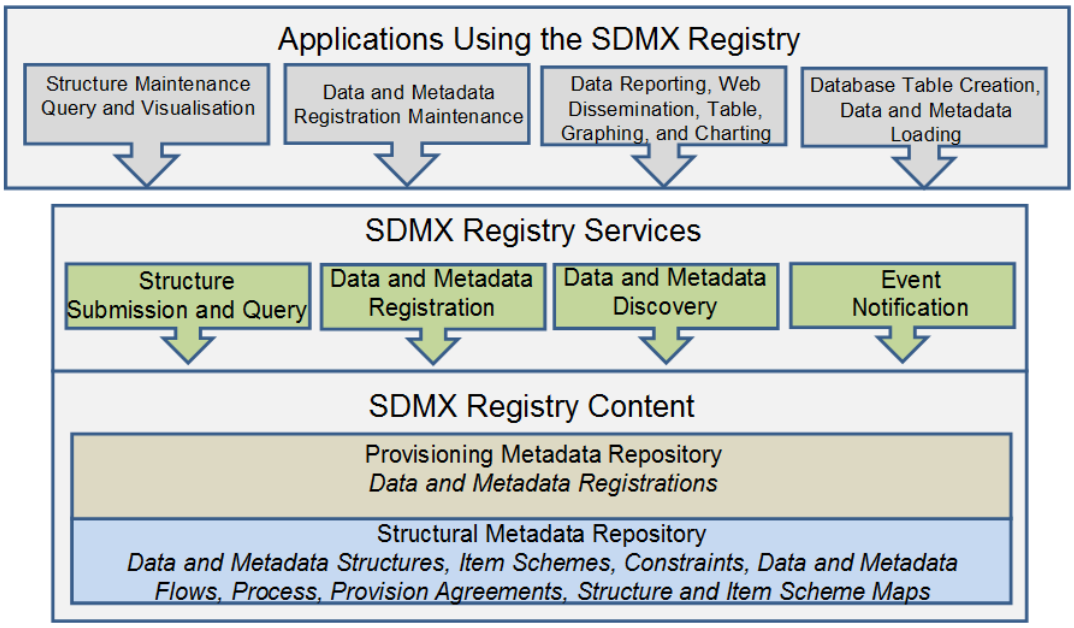


Figure 19: Schematic of the Registry Content and Services

8.4.2 Structural Metadata Repository

The basic layer is that of a structural metadata service which supports the lifecycle of SDMX structural metadata artefacts such as Maintenance Agencies, Data Structure Definitions, Metadata Structure Definitions, Provision Agreements, Processes etc. This layer is supported by the Structure Maintenance and Query Service.

Note that the SDMX-ML Submit Structure Request message supports all of the SDMX structural artefacts. The only Registry artefacts that are not supported by the SDMX-ML Submit Structure Request are:

- Registration of data and metadata sources
- Subscription and Notification

Separate registry-based messages are defined to support these artefacts.

8.4.3 Provisioning Metadata Repository

The function of this repository is to support the registration of various types of data-store which model SDMX-conformant databases or files, and to link to these data and reference metadata sources. These links can be specified for a data provider, for a specific data or metadata flow. In the SDMX model this is called the Provision Agreement.

This layer is supported by the Data and Metadata Registration Service.

8.5 Services of an SDMX Registry/Repository

8.5.1 Structure Maintenance and Query Service

A Registry offers a web service for the maintenance of structural metadata. Whilst this service must comply with the SDMX-ML Registry Interface, it is common for such registries to also provide a graphical user interface for the submission and maintenance of SDMX structures.

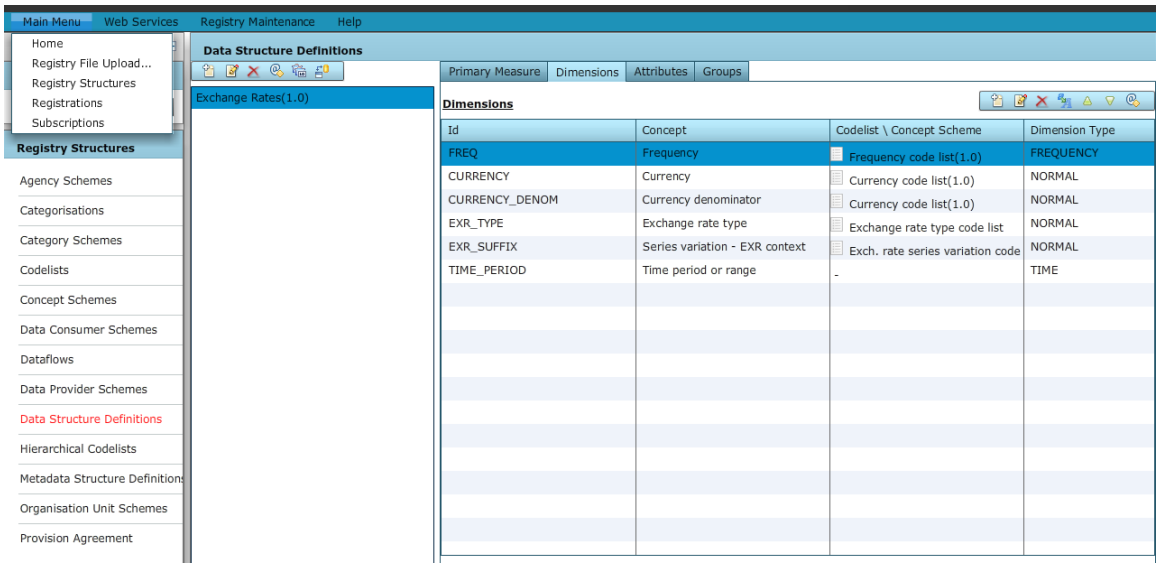


Figure 20: Example Graphical User Interface to an SDMX Registry

The Registry must also offer a query service for structural metadata.

8.5.2 Registration Service

The Registration service allows a data or metadata publisher (in SDMX terminology this is a Data Provider) to publish that data or metadata exist. As the Registration is connected to a Provision Agreement which itself is connected to a Data Provider and Data or Metadata Flow, the exact type of data or metadata registered is known, and also its structure (as the Data or Metadata Flow references the DSD or MSD).

The Registration must include a URL of either a file of SDMX-ML containing a data or metadata set, or a URL of a web service which can be queried to obtain the data.

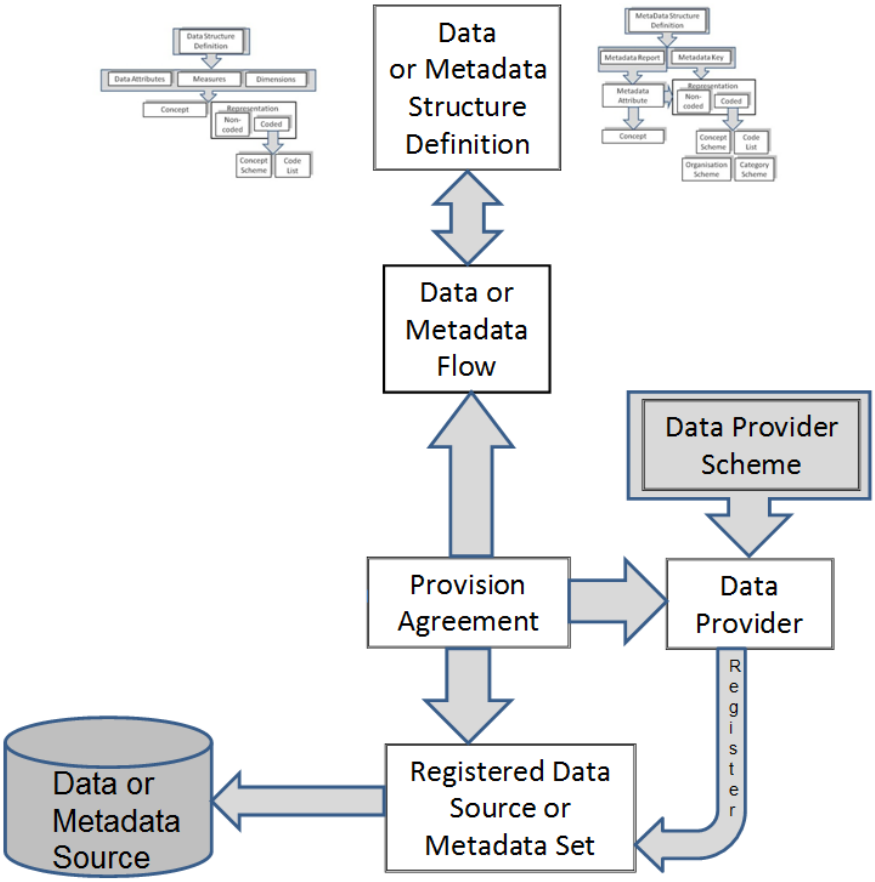


Figure 21: Schematic of Registered Data and Metadata Sources in the SDMX-IM

The Registration can be used both for data consumers to find data (this is described below in “Data and Metadata Discovery”) and for automated data reporting systems to harvest the data (the “pull” scenario).

8.5.3 Subscription and Notification Service

This subscription service allows a user to monitor Registry events. An event is something that has changed in the Registry. The subscription can be specified for specific objects such as a new registration of data or a change to a code list, or it can be specified for many objects such as any change to a specific type of structure or all registrations. A subscription can ask for notification by e-mail or notification to a URL (this will be a web service that can act upon the notification such as the automated update of structural metadata supporting a web dissemination service or the “pull” mechanism for data reporting).

When a monitored event is triggered the notification service notifies interested parties of the Registry content that has changed. The notification can be user (email) or application (URL).

8.6 Data and Metadata Discovery

8.6.1 Choreography

Discovering published data and reference metadata involves interaction with the Registry to:

- optionally (but usually) browsing a subject matter domain category scheme to find Dataflow Definitions (and hence Data Structure Definitions) and Metadataflows (and hence Metadata Structure Definitions) which structure the type of data and/or reference metadata being sought
- build a query, in terms of the selected Data Structure Definition or Metadata Structure Definition, which specifies what data are required and submitting this to a service that can query an SDMX Registry which will return a list of (URLs of) data and reference metadata files and databases which satisfy the query
- processing the query result set and retrieving data and/or reference metadata from the supplied URLs

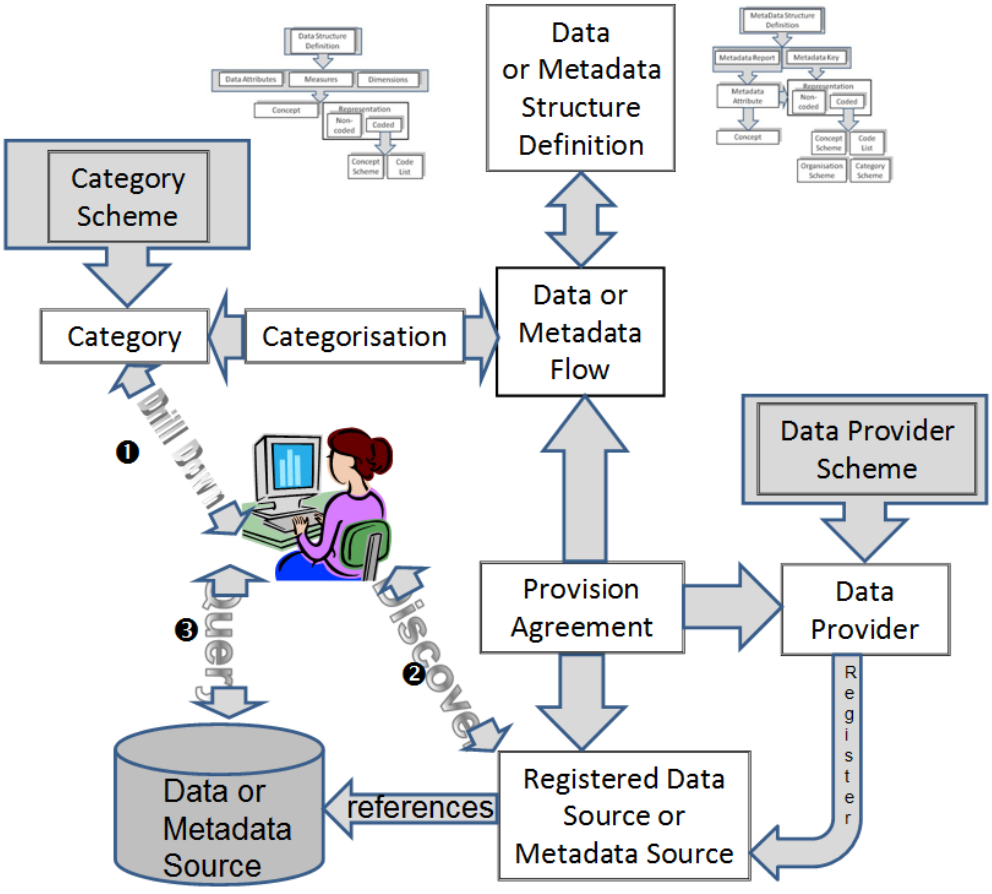


Figure 22: Schematic of Data and Metadata Discovery and Query in the SDMX-IM

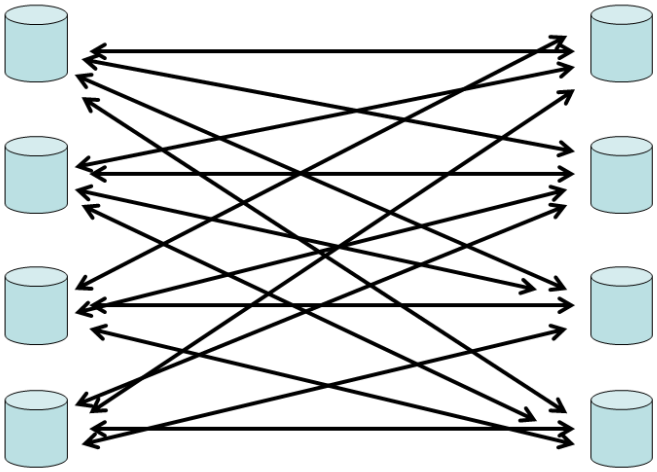
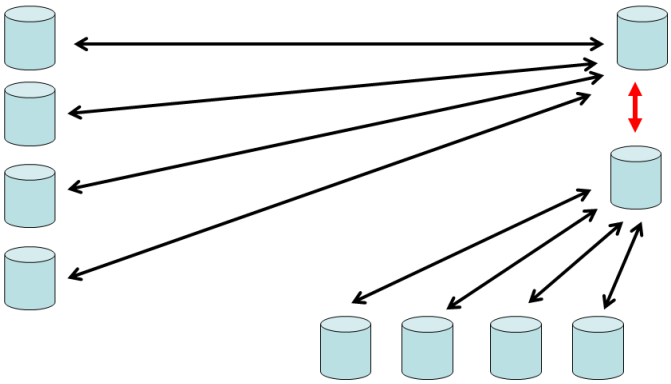
8.6.2 Example

A worked example of this scenario can be found in

Annex 4 – Data Reader and Data Writer Functions.

8.7 Patterns for Data and Metadata Exchange

SDMX identifies three basic process patterns and two modes (push and pull) regarding the exchange of statistical data and metadata.

Bilateral • Often person to person • Often many formats used for one type of data • Problem is "too many arrows" • Uses the "push" mode	
Gateway • Data or reference metadata collection shared between different organisations • Each organisation collects form its own community • Collecting organisations share the data • Can use the "pull" or "push" mode • Requires agreed common structural metadata (DSD and MSD)	

Data Sharing

- Reporters/Publishers register the existence of data/metadata
- Collectors/Consumers discover existence of data/metadata
- Collectors/Consumers retrieve data/metadata directly from the web service or file at a URL
- Uses the “pull” mode
- Requires agreed common structural metadata (DSD and MSD)

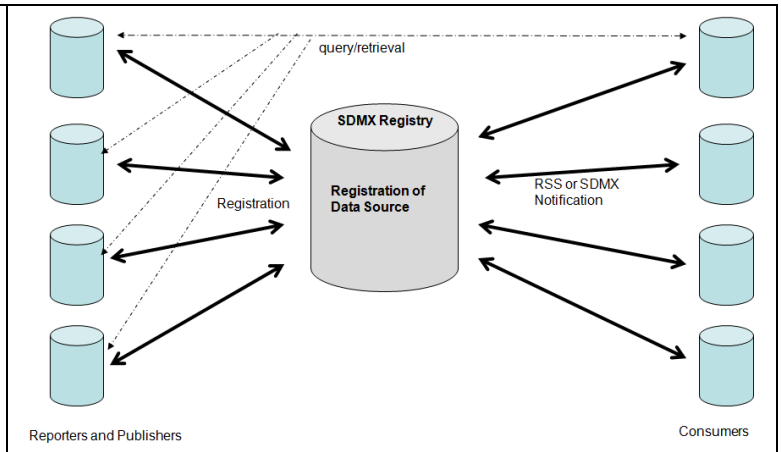


Figure 23: The Three Basic Patterns of Data and Reference Metadata Exchange

In the *push* mode the data/metadata are sent by the reporter to the collector.

In the *pull* mode the data consumer retrieves the data from the reporter's web server. The data may be made available for download in an SDMX-conformant file, or they may be retrieved from a database in response to an SDMX-conformant query, via a web service running on the reporter's server. In both cases, the data are made available to any organisation requiring them, in formats which ensure that data are consistently described by appropriate metadata, whose meaning is common to all parties in the exchange.

Data sharing using the pull mode is well adapted to the database-driven and data hub architectures. Both architectures provide the best benefits for the data producers because they can lessen the burden of publishing the data to multiple counterparties.

In both architectures, it is necessary to implement a notification mechanism, providing provisioning metadata in order to alert collecting organisations that data and metadata sets are made available by data providers, details about the online mechanism for getting data (for example, a queryable online database or a simple URL) and constraints regarding the allowable content of the data sets that will be provided.

At the heart of a data-sharing architecture there is often an SDMX Registry. This is a central location where structural and provisioning metadata can be found. In fact all the users/applications that need to access data can query the registry in order to know what data sets and metadata sets are available from data providers, and how to access them.

8.7.1 The Database-driven Architecture

The database-driven architecture is implemented by those collecting organisations that periodically need to fetch the data and to load them in their database. In general a batch

process is used in order to automate the flow in which a whole or a partial dataset, including incremental updating, is used.
From the data management point of view, the pull approach within a database-driven architecture includes the following steps:

1) when new data are available, the data provider should:
   a) create an SDMX-ML file containing the new data set

   or
   b) provide a web service (WS) that builds SDMX-ML messages upon request.

   In both cases a provision agreement must be in place.

2) the data collector Pull Requestor is notified of the new registration. Note that it is common for a Registry to provide an RSS mechanism, though this is nit a formal part of the SDMX standard. The data collector then:
   a) retrieves the SDMX-ML file from the specified URL, if it resides in a URL,

   or
   b) uses the Query Message or REST Query included in the feed to query the data provider web service, if the data are prepared by the data provider web service.

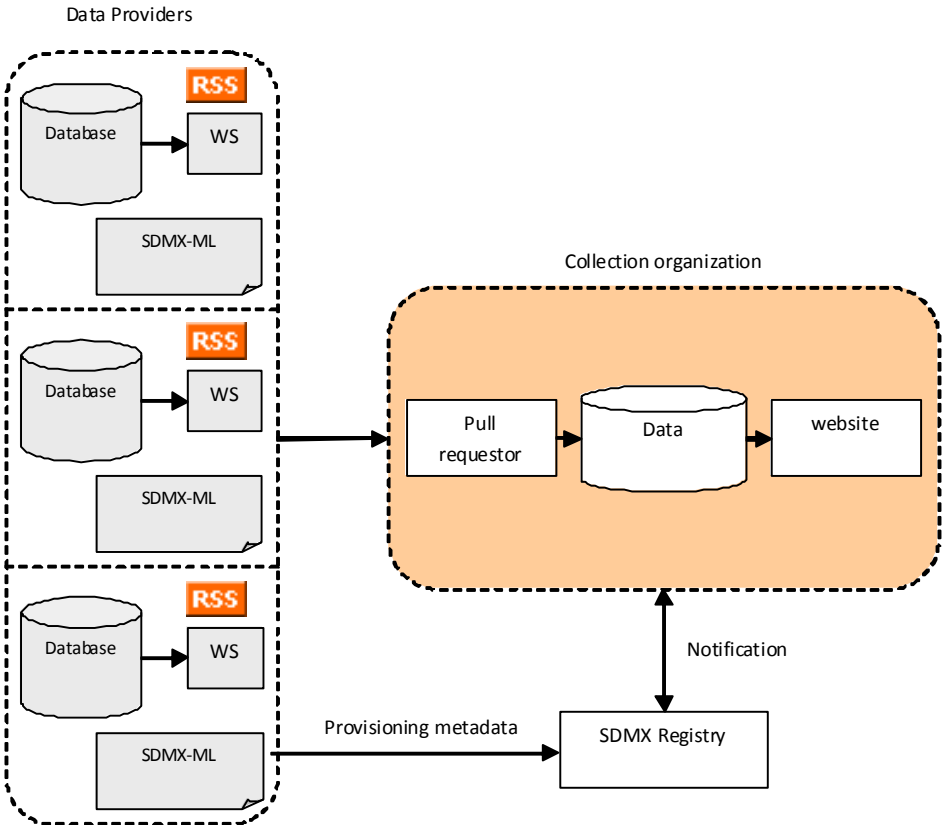


Figure 24: Database-driven Architecture

8.7.2 The Data Hub Architecture

The data hub architecture consists of an accessible system providing involved actors with the following services:

- data providers can:
    - notify the hub of new sets of data and corresponding structural metadata (measures, dimension, code lists, etc.);
    - make data available directly from their systems through a querying system.
- data users can:
    - browse the hub to define a dataset of interest via the above structural metadata;
    - retrieve the dataset from the data providers.

From the data management point of view, the hub is also based on agreed hypercubes or datasets, but here the hypercubes or datasets are not sent to the central system. Instead the following process operates:

1) a user identifies a dataset through the web interface of the central hub using the structural metadata, and requests it;
2) the central hub translates the user request in one or more queries and sends them to the related data providers' systems;
3) data providers' systems process the query and send the result to the central hub in a standard format;
4) the central hub puts together all the results originated by all interested data providers' systems and presents them in a human readable format.

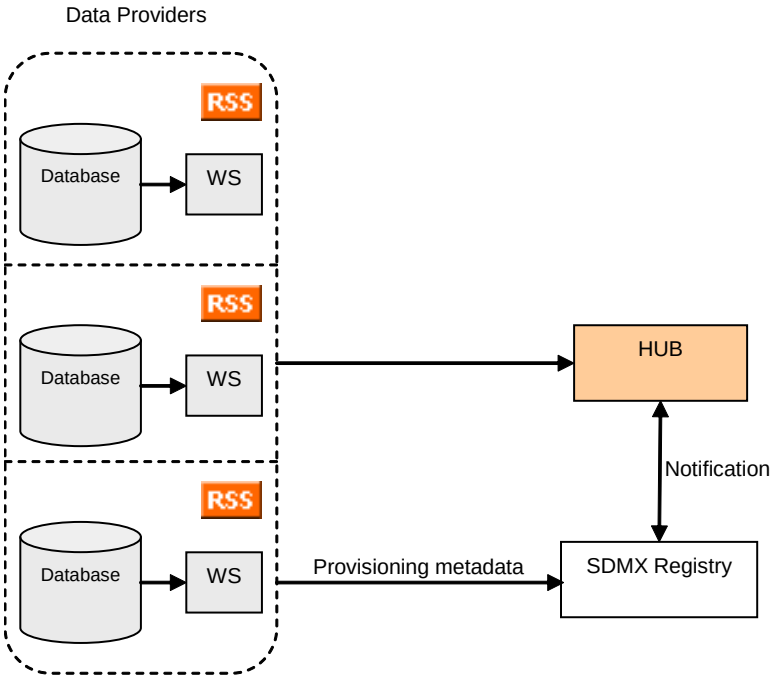


Figure 25: Data Hub Architecture

8.7.3 Data Producer Architectures

In order to implement an SDMX IT architecture for data-sharing using the pull mode, several steps must be accomplished by a data producer and several questions must be considered:

1. which statistical domains are involved and where are the data currently stored?
2. which structural metadata are involved, and where are they currently stored?
3. what is the business process behind the data flow involved in the exercise?
4. will the SDMX data producer architecture be part of a data warehouse architecture, of a data hub architecture or of both?

Generally data and structural metadata that will be involved in the new SDMX information system are stored either in databases or in files. The two cases lead to different architectural approaches:

a. data and structural metadata are already stored in a database and it is necessary to build suitable software interfaces in order to make the system “SDMX-compliant”.

b. a separate special-purpose database is set up to store data and structural metadata. This database will be designed with the main aim of being part of an SDMX-compliant system. In this case the database can be modelled using the SDMX Information Model. An SDMX Registry is, of course, such a database.

Both cases make it possible to:

- extract SDMX-ML files from the database that will be made available to be pulled by data collectors;

- allow the database to be queried directly through a web service.

Whichever type of data producer architecture is involved, a mapping process between structural metadata may be necessary, as explained in 5.3.

8.8 Registry Interfaces

8.8.1 Registry Interfaces

The Registry Interfaces are:

- Notify Registry Event
- Submit Subscription Request
- Submit Subscription Response
- Submit Registration Request
- Submit Registration Response
- Query Registration Request
- Query Registration Response
- Query Subscription Request
- Query Subscription Response
- Submit Structure Request
- Submit Structure Response

Applications communicate with the Registry using either the SDMX-ML Registry Interface message (which contains an XML element that defines the actual interface) or individual messages – one for each interface.

In more technical terms the Registry interfaces are invoked in one of two ways:

1. The interface is the name of the root node of the SDMX-ML document.
2. The interface is invoked as a child element of the RegistryInterface message where the RegistryInterface is the root node of the SDMX-ML document.

In addition to these interfaces the Registry must support a mechanism for querying for structural metadata.

All these interactions with the Registry – with the exception of Notify Registry Event – are designed in pairs. The first document (the one which invokes the SDMX Registry Interface) is a “Request” document. The message returned by the interface is a “Response” document.

It should be noted that all interactions are assumed to be synchronous, with the exception of Notify Registry Event. This document is sent by the SDMX-Registry to all subscribers whenever an even occurs to which any users have subscribed. Thus, it does not conform to the request-response pattern, because it is inherently asynchronous.

9 Architecture for an SDMX System

9.1 Scope of this Chapter

Use Case	Relevant
Data Reporting	☑
Load data into a database	☑
Report reference metadata	☑
Load metadata into a database (often called a metadata repository)	☑
Report data by means of the "pull" method	☑
Database Administration (automatic generation of database tables)	☑
Enable database to be compatible with SDMX Web Services	☑
Data Discovery	☑
Data and Metadata Query and Visualisation	☑

This Chapter brings together the different SDMX constructs described in this User Guide into a coherent architecture that supports the typical SDMX use cases shown above. The architecture described here shows how applications need not be concerned with the actual XML (or SDMX-EDI) or version of SDMX, but are concerned with interfaces offered by components that work with objects constructed according to the SDMX Information Model (called SDMX Beans in the diagrams) which are themselves built from the actual XML (or SDMX-EDI). For data and metadata sets the actual XML is read directly into or written directly out of the SDMX Component.

Such an architecture, known as a Component Architecture, enables applications to be built quickly and with minimal resource. As each Component is agnostic to the context in which it is used it can therefore be used by many applications (e.g. a Data Writer Component may be used in a database application, or an application reading a spreadsheet), thus reducing considerably the development effort and resulting in more robust and more easily maintained systems.

9.2 Architecture

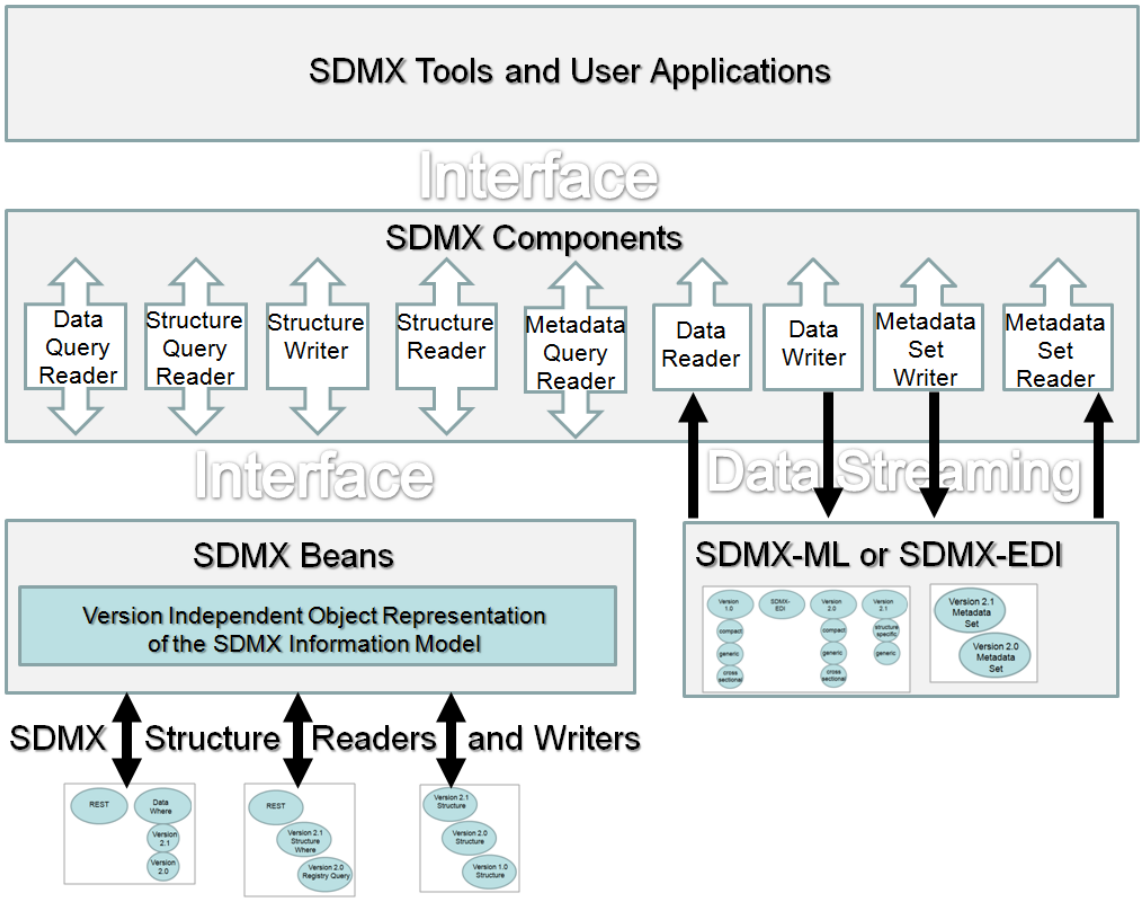


Figure 26: Schematic of a Component Architecture

The SDMX Beans underpin the framework and are what may be referred to as 'domain objects', 'data transfer objects', or 'business objects'. The SDMX Beans are an Object representation of the information being exchanged. The information may have been exchanged in SDMX, EDI, or even CSV, it does not matter, as the SDMX Beans are not coupled to a particular syntax or version of the standard. The SDMX Beans are format and version independent allowing the software that uses them to be decoupled from the format in which the data/metadata are exchanged.

Whilst "bean" objects can be created for a data set or metadata set, this does not scale well for large volumes of data/metadata. Therefore, the architecture shows a data streaming approach where the data/metadata are read or written directory to/from an input or output stream – this could be a local file or streamed over the internet.

The "structure" components use the SDMX Beans by means of the object interfaces, thus they do not need to be concerned with the external format of the structural metadata – this is done by structure readers and writers.

For data sets and metadata sets there are no “beans” but rather the data/metadata are input to/output from the data and metadata readers and writers.

9.3 Using the Architecture

9.3.1 Tools and Applications

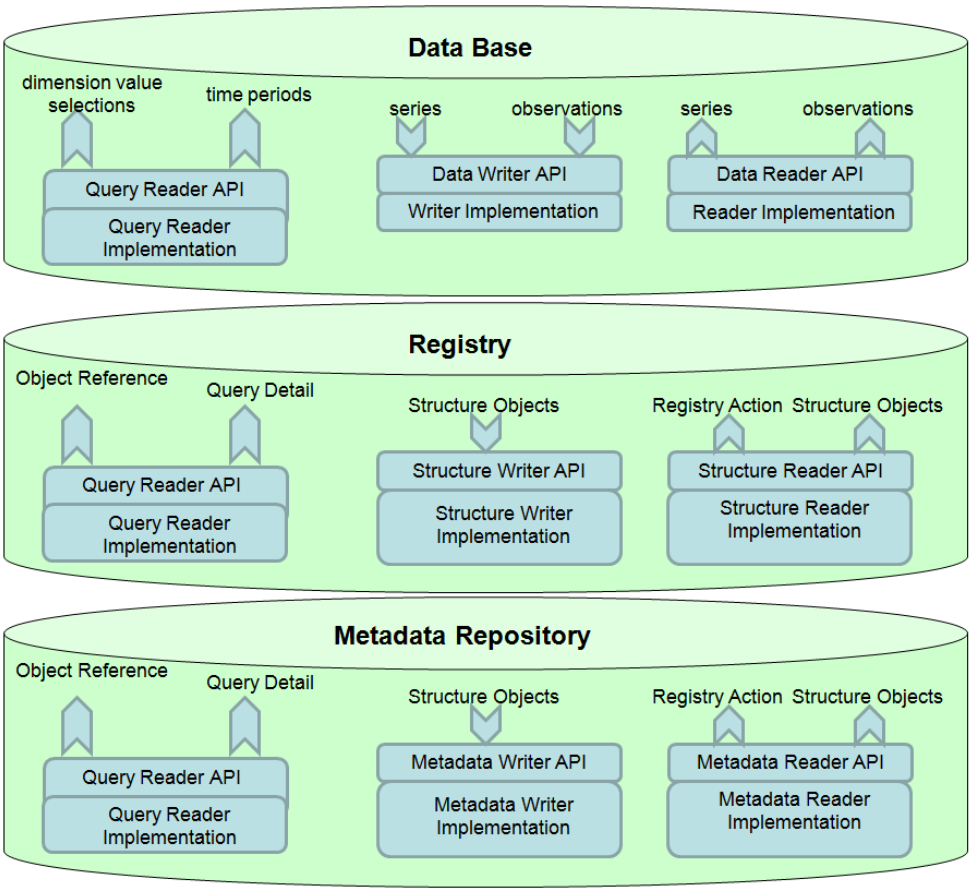


Figure 27: Applications Supporting Use Cases

These three basic tools/applications support the requirements of many of the use cases listed in 3.2.

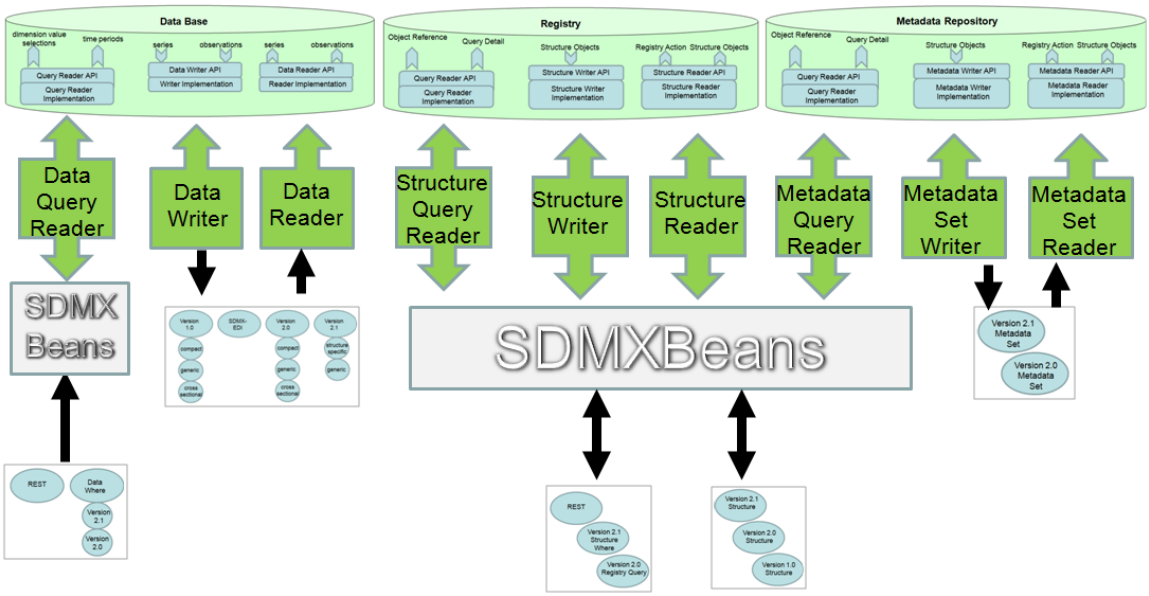


Figure 28: Applications Integrated with a Component Architecture

A Component Architecture enables the components to be shared by many applications thus reducing development and maintenance effort and resulting in a more robust system.

9.4 Availability of Component Architectures

Component Architectures based on the schematic depicted above are beginning to emerge as freely available components or open source. The SDMX Tools Database (accessible from the SDMX website at www.sdmx.org) contains details of many SDMX tools including complete architectures.

10 Community Management

10.1 Scope of this Chapter

Many organizations have a community of data reporters, data sharing agencies, or data consumers. This places responsibilities upon the organization in terms of publishing and dissemination of structural metadata, setting up hubs for data sharing, administration of maintenance agencies, usernames etc.

These aspects are dealt with in this Chapter.

10.2 Maintenance Agency Maintenance

All structural metadata in SDMX is owned and maintained by a maintenance agency (Agency identified by `agencyID` in the schemas). It is vital to the integrity of the structural metadata that there are no conflicts in `agencyID`.

The maintenance of maintenance agencies in SDMX is a devolved function. Any organization registered as a maintenance agency can itself set up and maintain its own maintenance agencies. However, in order for this devolved system to work there must be a "top-level" maintenance agency list (called an Agency Scheme in SDMX) which is itself maintained by the recognized top-level Agency. This Agency is SDMX. Any organization registered in the SDMX Agency Scheme can itself maintain its own Agency Scheme of sub-agencies. Only one such Agency Scheme is allowed for any recognized Agency. With the exception of the "SDMX" Agency, a recognized Agency must itself be registered in a "parent" Agency Scheme.

The following rules are a more formal definition of the way the SDMX agency system works:

1. Agencies are maintained in an Agency Scheme.
2. The maintenance agency of the Agency Scheme must be registered as an Agency in a (different) Agency Scheme (the "parent" Agency Scheme).
3. The "top-level" agency is SDMX and this agency scheme is maintained by SDMX.
4. Agencies registered in the top-level scheme can themselves maintain a single Agency Scheme. SDMX is an agency in the SDMX agency scheme, thus allowing SDMX be a maintenance agency. Agencies in this (child) scheme can themselves maintain a single Agency Scheme and so on.
5. The Agency Scheme cannot be versioned and so takes a default version number of 1.0, and it cannot be made "final".
6. There can be only one Agency Scheme maintained by any one Agency. It has a fixed Id of `AGENCIES`.
7. The format of the agency identifier is `agencyId.agencyID` etc. The top-level agency in this identification mechanism is the agency registered in the SDMX

agency scheme. In other words, SDMX is not a part of the hierarchical ID structure for agencies.

8. SDMX is, itself, a maintenance agency.

This supports a hierarchical structure of `agencyID`.

An example is shown below.

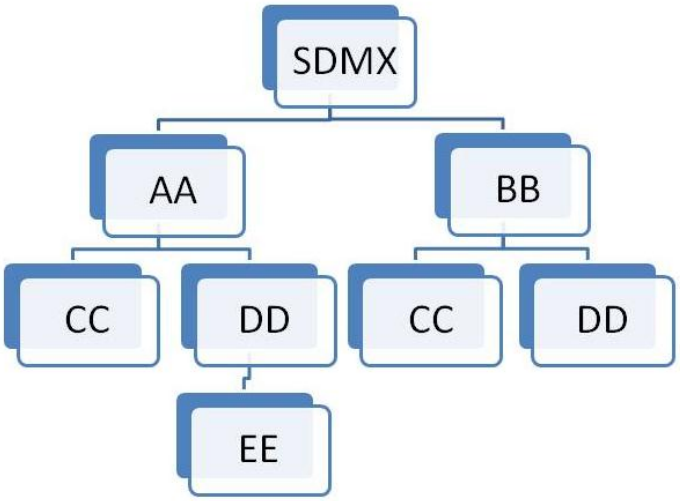


Figure 29:Example of Hierarchic Structure of Agencies

Each agency is identified by its full hierarchy excluding SDMX.

The XML representing this structure is shown below.

```
<structure:OrganisationSchemes>
  <structure:AgencyScheme agencyID="SDMX" id="AGENCIES">
    <common:Name>SDMX Agency Scheme</common:Name>
    <structure:Agency id="AA">
      <common:Name>AA Name</common:Name>
    </structure:Agency>
    <structure:Agency id="BB">
      <common:Name>BB Name</common:Name>
    </structure:Agency>
  </structure:AgencyScheme>
  <structure:AgencyScheme agencyID="AA" id="AGENCIES">
    <common:Name>AA Agency Scheme</common:Name>
    <structure:Agency id="CC">
      <common:Name>CC Name</common:Name>
    </structure:Agency>
    <structure:Agency id="DD">
      <common:Name>DD Name</common:Name>
    </structure:Agency>
  </structure:AgencyScheme>
  <structure:AgencyScheme agencyID="BB" id="AGENCIES">
    <common:Name>BB Agency Scheme</common:Name>
    <structure:Agency id="CC">
      <common:Name>CC Name</common:Name>
    </structure:Agency>
    <structure:Agency id="DD">
      <common:Name>DD Name</common:Name>
    </structure:Agency>
  </structure:AgencyScheme>
  <structure:AgencyScheme agencyID="AA.CC" id="AGENCIES">
    <common:Name>AA.CC Agency Scheme</common:Name>
    <structure:Agency id="EE">
      <common:Name>EE Name</common:Name>
    </structure:Agency>
  </structure:AgencyScheme>
</structure:OrganisationSchemes>
```

Figure 30: Example Agency Schemes Showing a Hierarchy

Example of Structure Definitions:

```
<structure:Codelists>
   <structure:Codelist id="CL_BOP" agencyID="SDMX" version="1.0">
      <common:Name>name</common:Name>
   </structure:Codelist>
   <structure:Codelist id="CL_BOP" agencyID="AA" version="1.0">
      <common:Name>name</common:Name>
   </structure:Codelist>
   <structure:Codelist id="CL_BOP" agencyID="AA.CC" version="1.0">
      <common:Name>name</common:Name>
   </structure:Codelist>
   <structure:Codelist id="CL_BOP" agencyID="BB.CC" version="1.0">
      <common:Name>name</common:Name>
   </structure:Codelist>
</structure:Codelists>
```

Figure 31: Example Showing Use of Agency Identifiers

Each of these maintenance agencies has an identical Codelist with the Id CL_BOP. However, each is uniquely identified by means of the hierarchic agency structure.

Clearly, in order for such a system to work there must be a mechanism that enables a user or organisation to discover the full "list" of maintenance agencies or at least the Agency Scheme in which an organisation is registered. In order for this to be possible all Agency Schemes must be made known to a Global SDMX Registry and either maintained in that Registry or maintained and made available from a metadata source that is referenced from an entry in the Global Registry. For example, the Agency Schemes shown in Figure 30 could be "registered" in the Global Registry as follows:

```
<structure:AgencyScheme agencyID="AA" id="AGENCIES" isExternalReference="true"
structureURL="http://www.AA.org/RegistryQuery/AgencyScheme/AA/AGENCIES">
   <common:Name>AA Agency Scheme</common:Name>
   </structure:AgencyScheme>
<structure:AgencyScheme agencyID="BB" id="AGENCIES" isExternalReference="true"
structureURL="http://www.BB.org/RegistryQuery/AgencyScheme/BB/AGENCIES">
   <common:Name>BB Agency Scheme</common:Name>
</structure:AgencyScheme>
<structure:AgencyScheme agencyID="AA.CC" id="AGENCIES" isExternalReference="true"
structureURL="http://www.AACC.org/RegistryQuery/AgencyScheme/AA.CC/AGENCIES">
   <common:Name>AA.CC Agency Scheme</common:Name>
</structure:AgencyScheme>
```

Figure 32: Example XML Showing External References

10.3 Dissemination of Structural Metadata

When structural metadata are disseminated or exchanged it is important that any consuming application has access to all of the structural metadata that is referenced from structures such as a DSD or MSD (i.e. in these cases the Concept Schemes, Code Lists, Category Schemes etc. that are "used" in the DSD or the MSD). These structures can be embedded as complete structures in the file that is disseminated, or they can be embedded as "stubs" that contain the reference from where the structures can be retrieved. This referencing mechanism is achieved using one of the SDMX-ML attributes:

- `structureURL`
- `serviceURL`

These attributes are available on every structure that is maintained e.g. it is at the level of the "maintained object" such as Code List, Data Structure Definition. In addition to the use of these attributes the attribute `isExternalReference` should be set to "true" indicating that the full definition of the structure is not available and must be retrieved from one of the reference attributes `structureURL` or `serviceURL`.

It is therefore the responsibility of the agency maintaining these structures that the structures can be accessed using the URL. If the URL cannot be used to retrieve the structure then it is possible that applications using the structures will not be able to process the information properly. It follows that the content of `structureURL` and `serviceURL` should be deemed to be resolvable in the long term.

The use of or a link to a shared community SDMX Registry is recommended for organizations that wish to store such structures. For example, the shared community registry can be used as the repository for the referenced structures.

Note that any structural metadata submitted to an SDMX Registry (SubmitStructureRequest) must have resolvable references to all of the structures cross referenced in the submitted structure. If this is not the case then the submission may be rejected.

10.4 Maintenance of Community Concept Roles

10.4.1 Overview

The DSD Components of Dimension and Attribute can play a specific role in the DSD and it is important to some applications that this role is specified. For instance, the following roles are some examples:

Frequency – in a data set the content of this Component contains information on the frequency of the observation values
Geography - in a data set the content of this Component contains information on the geographic location of the observation values
Unit of Measure - in a data set the content of this Component contains information on the unit of measure of the observation values

In order for these roles to be extensible and also to enable user communities to maintain community-specific roles, the roles are maintained in a controlled vocabulary which is implemented in SDMX as Concepts in a Concept Scheme.

It is possible to specify zero or more concept roles for a Dimension, Measure Dimension and Data Attribute (but not the ReportingYearStartDay). The Time Dimension, Primary Measure, and the Attribute ReportingYearStartDay have explicitly defined roles and cannot be further specified with additional concept roles.

10.4.2 Maintaining and Using Concept Roles

The mechanism for maintaining and using concept roles is described in the SDMX Standards Section 6: Technical Notes. It is the responsibility of Agencies to ensure their community knows which concepts in which concept schemes play a “role” and the significance and interpretation of this role. In other words, such concepts must be known by applications, there is no technical mechanism that can inform an application on how to process such a “role”.

Clearly, Agencies defining DSDs can use Concepts from any Concept Scheme, even a Concept Scheme not maintained by this Agency. It is therefore important that any Concept Scheme that is referenced from the DSD must be available from a known URL to any application that needs to process the DSD, or be embedded in the file of structural metadata (this is no different from any other cross referenced structure as described in 10.3 above). Therefore, if an Agency decides to maintain its own Concept Scheme of concept roles it needs to consider whether this scheme is “public” (i.e. it is required in the DSDs disseminated by the organisation) and, if so, ensure that it is made available.

10.5 Hosting of a shared registry

It is clear from the responsibilities of an Agency that there is a need to ensure structural metadata maintained by that Agency, and which is to be used outside of the organization that is the Agency, is made available. As more and more organizations use SDMX and as more Agencies are created, it is clear that a central repository of structural metadata will benefit the “community” of the Agency. Many of these communities will exist already, as they will pre-date the existence of. The SDMX standard formalizes the way a community exchanges and shares data and reference metadata, and as more SDMX communities are created then the greater is the need for control over the maintenance and sharing of the structural metadata.

Community “Agencies” should consider taking on the role of hosting a shared SDMX Registry for the community. Clearly, there will be many of these shared registries and it may be necessary that these registries can be linked in a “federation” so that common structural metadata is accessible. This is especially true of the Agency Schemes that will exist: it will only be possible to validate an agencyID if the Agency is known and this may involve knowledge of the full hierarchy of Agencies which will be need to be built from the Agencies maintained in separate Agency Schemes. Furthermore, it is probable that many Code Lists and Concept Schemes will not be disseminated with DSDs and MSDs,

but rather be referenced from the structure file disseminated. This type of federated architecture is best supported by an SDMX Registry, as this is the one of the key roles of an SDMX Registry.

10.6 Data Provider Maintenance

An organization that collects data or reference metadata from other organizations (these are known as Data Providers in SDMX) must maintain a `DataProvider` Scheme. This is self evident as the collecting organization must know from which reporting organization data or reference metadata is received. Clearly, an organization can collect data or reference metadata in an SDMX format without having an SDMX `DataProvider` Scheme. Nevertheless, data collecting organizations which adopt SDMX are encouraged to maintain an SDMX `DataProvider` Scheme, or to be able to disseminate such a scheme from their own internal scheme.

If the collecting organization wishes to adopt the “pull” method of data/metadata reporting, or wishes to host an SDMX Registry where its community can publish the existence of data and reference metadata sources (by means of a data/metadata Registration), then it is mandatory to have a `DataProvider` Scheme. The relationship between the fundamental structures supporting the “pull” mechanism and data/metadata discovery is shown in the diagram below.

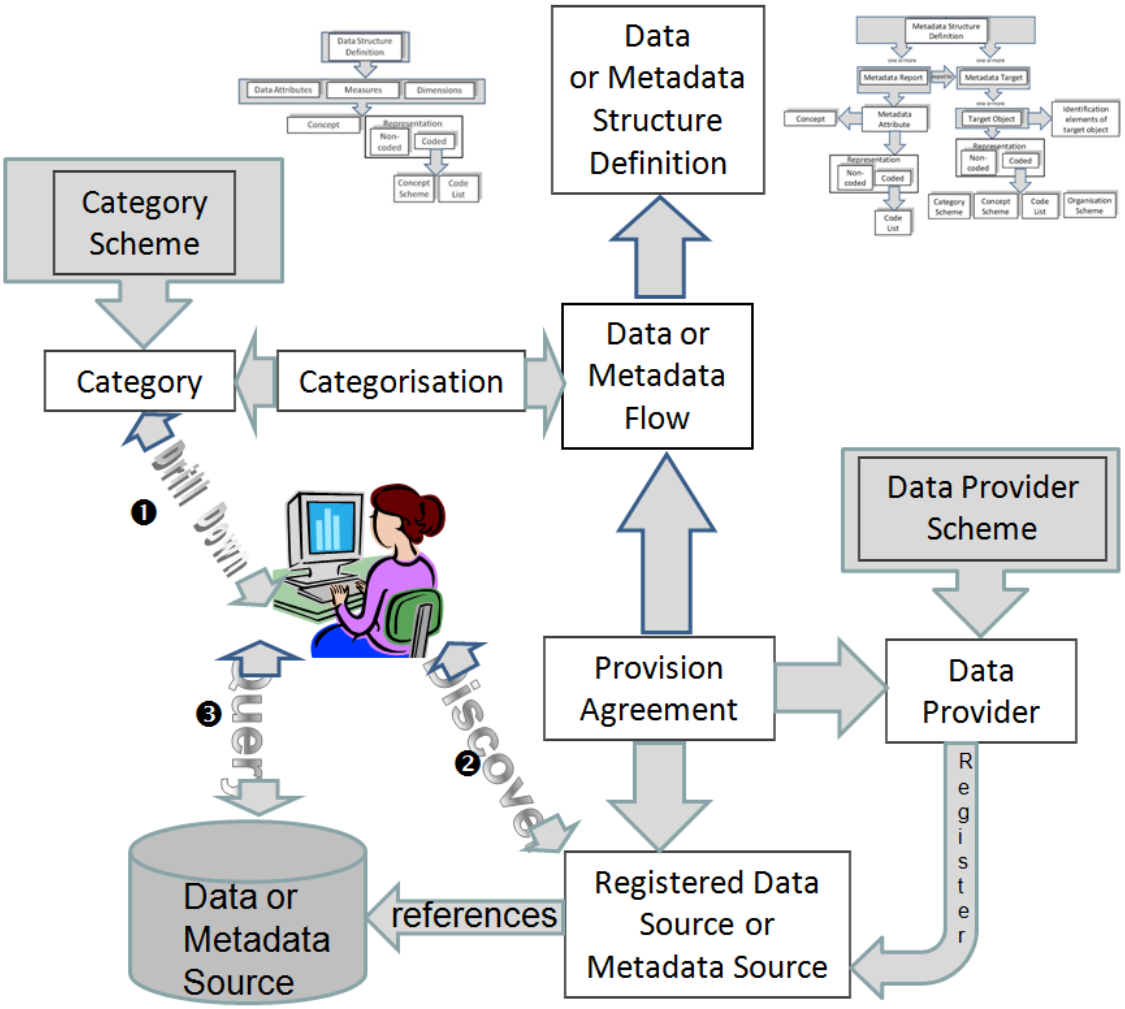


Figure 33: SDMX structures required for the "pull" reporting mechanism or for data/metadata discovery

11 Annex 1 –Content Oriented Guidelines (COG)

11.1 Scope of the COG

The SDMX Content-Oriented Guidelines are a set of work products focused on the use of SDMX within statistical domains, to support harmonization of data collection and dissemination across those domains. These documents are frequently updated, and can be found on the SDMX website at: http://sdmx.org/?page_id=11.

SDMX assumes that organizations responsible for coordinating activities within their domains will be the producers of domain-specific SDMX artefacts such as DSDs and MSDs. The intent of the COG is to produce artefacts that generally apply across a wide number of statistical domains – there may be exceptions within specific domains. Thus, organizations are encouraged to use the terminology, concepts, codelists, etc. described in these documents where applicable.

There is a single covering document, *Content-Oriented Guidelines*, and five annexes:

Annex 1 – Cross-Domain Concepts
Annex 2 – Cross-Domain Code Lists
Annex 3 – Statistical Subject-Matter Domains
Annex 4 – Metadata Common Vocabulary
Annex 5 – SDMX-ML for Content-Oriented Guidelines

The last of these is simply a compressed .ZIP file containing the SDMX-ML rendering of each of the other annexes, so this does not need further description.

Each of these documents will be described in turn, to assist the reader in knowing how to use them in their own implementations.

11.2 Content-Oriented Guidelines

This document is the covering document for the entire set of guidelines. It provides an overview, including some background on the COG and their scope and purpose, and it introduces the three main threads of the work: (1) cross-domain concepts and codelists; (2) the classifications of statistical subject-matter domains; and (3) the relevant standard terminology/vocabulary. A fourth major section describes the governance model for the COG, and related topics.

This document is one which should be read before reading the specific annexes.

11.3 COG Annex 1 – Cross-Domain Concepts

11.3.1 Scope

This document provides a set of concepts which apply broadly across statistical domains, used both for describing statistical data sets (as dimensions, attributes, or measures in DSDs) and for describing statistical metadata not related directly to data sets (as metadata attributes in DSDs). There is a listing of 66 top-level concepts in the current draft, although this list will be growing. Some of these are broken out into subordinate uses of the concepts for different applications. As one would expect, these cover many obvious areas such as geography and time, but also include many of the standard concepts used in reporting data quality. If the definition of a cross-domain concept is not exactly what it needs to be for a specific use, then a similar concept should be defined, but its agency should reflect its origins, and the “sdmx” agency should not be used.

The Cross-Domain Concepts are the product of many years of discussion, and inter-agency agreement in terms of using SDMX and the earlier GESMES/TS for describing data, and in many senses are already well-implemented in existing SDMX systems.

11.3.2 Usage of Cross-Domain Concepts

Definitions comprise, for example, frequency, reference area, possibly currency etc. They also contain statistical concepts which are in general used for the measurement or declaration of data quality. Examples are "accuracy", "comparability" or "timeliness". These quality-related concepts are normally implemented by statistical organisations within their respective Metadata Structure Definitions. This enables the applying statistical organisation to measure and analyse data quality in a structured and harmonised manner across the whole statistical production of this organisation.

The Guidelines provide for each SDMX Cross-domain concept an ID and a detailed description, which may be complemented by additional comments. The table below presents the two Cross-Domain concepts “Accuracy” and “Currency”.

Concept ID:	ACCURACY
Description:	Closeness of computations or estimates to the exact or true values that the statistics were intended to measure.
Context:	The accuracy of statistical information is the degree to which the information correctly describes the phenomena. It is usually characterized in terms of error in statistical estimates and is often decomposed into bias (systematic error) and variance (random error) components. Accuracy can contain either measures of accuracy (numerical results of the methods for assessing the accuracy of data) or qualitative assessment indicators. It may also be described in terms of the major sources of error that potentially cause inaccuracy (e.g., coverage, sampling, non response, response error). Accuracy is associated with the "reliability" of the data, which is defined as the closeness of the initial estimated value to the subsequent estimated value. This concept can be broken down into: Accuracy - overall (summary

	assessment); Accuracy - non-sampling error; Accuracy - sampling error.
Presentation:	Free text
Concept ID:	**CURRENCY**
Description:	Monetary denomination of the object being measured.
Presentation:	CL_CURRENCY

Error! Reference source not found. The figure below provides a simplified view of how the cross-domain concepts are used for defining data/metadata structure definitions in the SDMX framework. In the data/metadata structures they are usually combined with domain specific concepts.

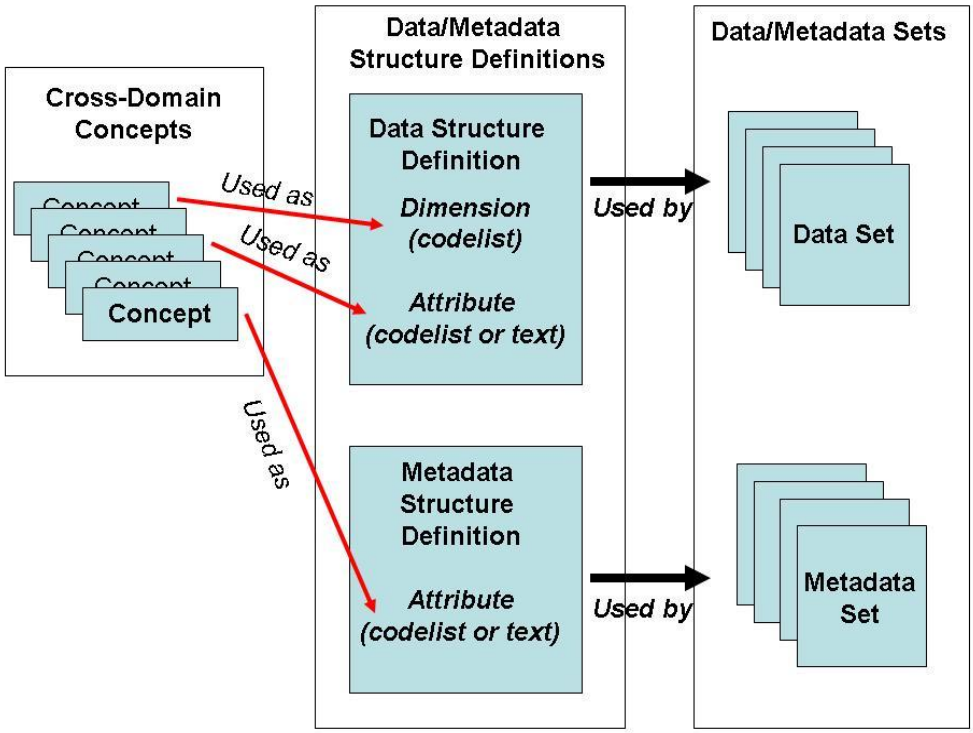


Figure 34: Use of Concepts Defining Data/Metadata Structure Definitions in the SDMX Framework

The illustration shows that cross-domain concepts can have two different roles:

- As structural metadata: as dimensions in a data structure definition, to identify each statistical observation. For example, a dimension named “Reference Area” would explain which country or geopolitical aggregate a specific statistical observation refers to.

- As Reference metadata: Attributes in a data structure definition or in a metadata structure definition. Attributes provide information about the data, thus qualifying the data further (example, “unit of measure”) or can be used to report metadata, for example with concepts such as timeliness, reference period, classification system and data compilation. The values of these concepts may be coded, but are more often free text.

In the case where a concept can be represented as coded, there must be a link to the code list containing valid values that may be reported. If the concept is used as an attribute, the attachment level must be indicated. This means an indication of the data object or structure, e.g. “time series” or “observation”, to which the concept is linked.

11.4 COG Annex 2 – Cross-Domain Code Lists

The companion to the Cross-Domain Concepts is their representation using code lists. This annex identifies some recommended code lists for use with several of the cross-domain concepts, or even independent of the standard concepts. At the current time, there are 9 recommended code lists, with some issues and supplementary values identified for them.

It should be noted that often, agreement on code lists is harder to achieve than agreement on concepts, as code lists vary more from organization to organization and domain to domain than do concepts.

Unless a recommended code list has all needed values, it should not be used. A code list differing in any way from the ones provided should be given its own name and agency – the “sdmx” agency should never be used for an altered code list. It is useful to note that having a few unused codes in an SDMX structure is typically not an issue – SDMX provides constraints to handle this situation, so it is only when a superset is needed that a different code list should be used.

11.5 COG Annex 3 – Statistical Subject-Matter Domains

This set of guidelines provides a break-down of all the domains of official statistics, at a high level. It is based on work done at the UN/ECE, which created a database tracking organizations and their statistical activities. Thus, it reflects the actual state of statistical data collection at a high level within the world of official statistics.

The Statistical Subject-Matter Domains breaks down statistics into high-level categories – no greater than three levels deep, and often only two. It is intended to be used as the basis for organizing collections of statistical data within repositories and registries, and as such is provided in the form of an SDMX category scheme when expressed as SDMX-ML. It will often be the case that specific domains or organizations may wish to drive this classification to a more detailed level for actual use. In such a case, the provided classification will need to be extended.

The subject matter domain scheme is shown below.

Domain Scheme[3]

Domain 1: Demographic and social statistics	Domain 2: Economic statistics	Domain 3: Environment and multi-domain statistics
1.1 Population and migration 1.2 Labour 1.3 Education 1.4 Health 1.5 Income and consumption 1.6 Social protection 1.7 Human settlements and housing 1.8 Justice and crime 1.9 Culture 1.10 Political and other community activities 1.11 Time use	2.1 Macroeconomic statistics 2.2 Economic accounts 2.3 Business statistics 2.4 Sectoral statistics 2.4.1 Agriculture, forestry, fisheries 2.4.2 Energy 2.4.3 Mining, manufacturing, construction 2.4.4 Transport 2.4.5 Tourism 2.4.6 Banking, insurance, financial statistics 2.5 Government finance, fiscal and public sector statistics 2.6 International trade and balance of payments 2.7 Prices 2.8 Labour cost 2.9 Science, technology and innovation	3.1 Environment 3.2 Regional and small area statistics 3.3 Multi-domain statistics and indicators 3.3.1 Living conditions, poverty and cross-cutting social issues 3.3.2 Gender and special population groups 3.3.3 Information society 3.3.4 Globalisation 3.3.5 Indicators related to the Millennium Development Goals 3.3.6 Sustainable development 3.3.7 Entrepreneurship 3.4 Yearbooks and similar compendia

[3] Reference source: United Nations Economic Commission for Europe. This classification of statistical subject-matter domains is based on that used in the Database of International Statistical Activities (DISA - http://unece.unog.ch/disa/). The DISA classification includes two additional domains covering statistical methodology and strategic managerial issues, which do not relate directly to data or metadata, so are not considered relevant for SDMX purposes.

11.6 COG Annex 4 - Metadata Common Vocabulary (MCV)

This document provides definitions and other information regarding many of the terms and concepts which are relevant when using SDMX. Very often, it takes terms from other sources (such as the *OECD Glossary of Terms* or Eurostat's *CODED* database). It is not intended to be a comprehensive definition of all terms related to official statistics – these other sources are broader in scope. It is intended to reflect the standard meanings found within an SDMX context.

In SDMX-ML, it is represented as a reference metadata report. This simply provides a useful XML format for those wishing to load this into a database or otherwise process it.

12 Annex 2 – SDMX Business Process Model

12.1 Introduction

The Generic Statistical Business Process Model (GSBPM) is a reference model of the statistical production life-cycle in national statistical agencies, developed by the METIS group in UN/ECE. The work was based on many earlier models, and represents a view of statistical production which is now being accepted as the standard view.

For this reason, we are using the GSBPM as the basis of an example, demonstrating how SDMX fits into the work of a national-level statistical agency.

This example is not a technical one – rather, it is meant to describe the use of SDMX from a business perspective: how, where, and why is SDMX used? These questions will be answered by using the example of effective exchange rates..

It is important to note that in some scenarios, where collected data is already in the form of aggregates, SDMX might be used earlier in the business process. However, for NSOs the most common scenario is probably where micro-data are collected and aggregated at the national level.

There are many benefits to the use of SDMX, and while many of these are related to the use of technology, in the end the real benefits are simple: it becomes easier for users to locate and utilize data, and the data themselves are more comparable. Further, the data become easier to visualize and format, into whatever form is needed, either for the creation of dissemination outputs, or for re-formatting by data users or collectors.

12.2 High Level Schematic of the GSBPM

It is important to have at least a high-level understanding of the GSBPM, as shown in the diagram below.

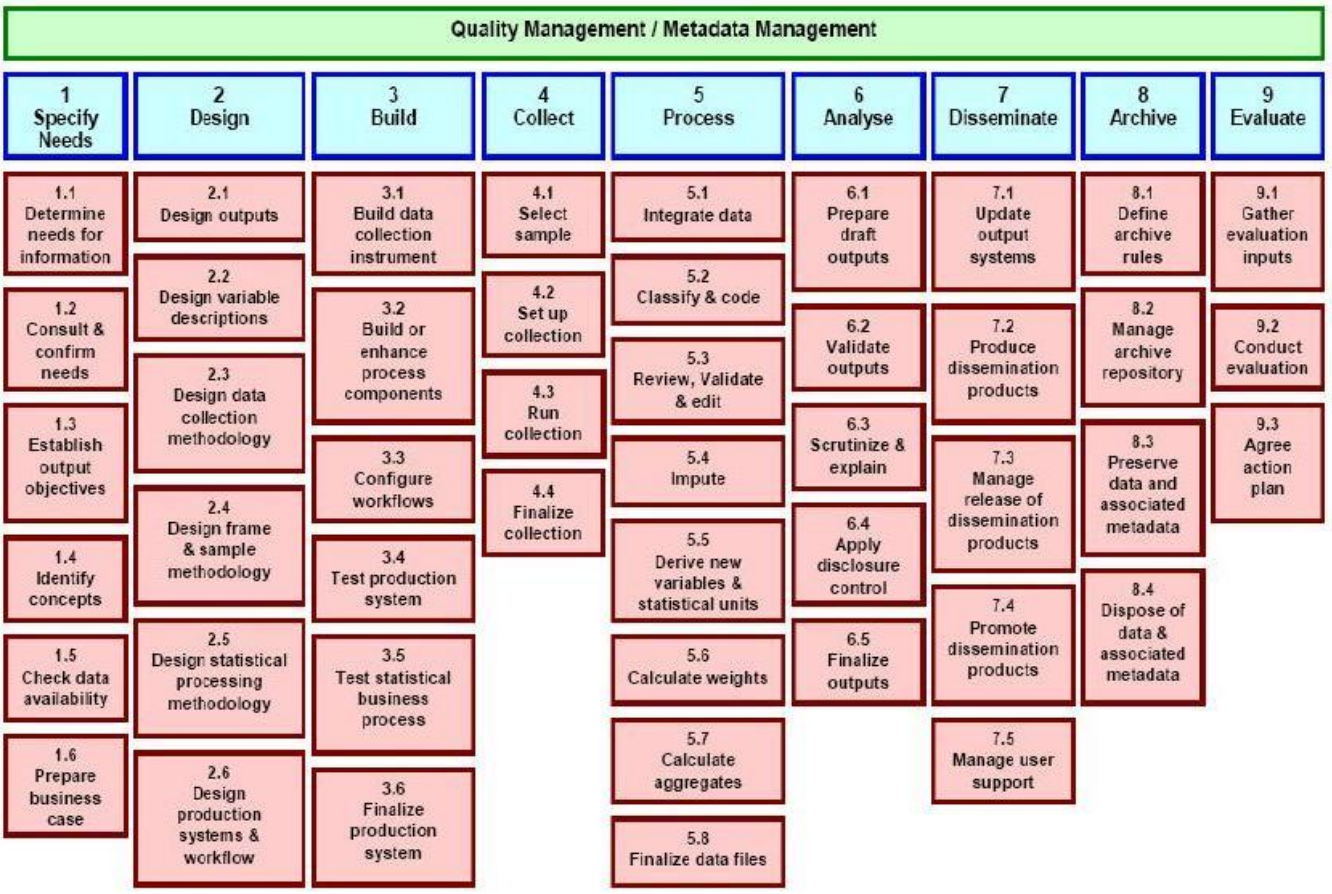


Figure 35: Hush-level schematic of the GSBPM

Across the top of the diagram, we see the high-level process steps, from 1 to 9. The process begins with the evaluation of data collection needs, and proceeds through the design and creation of data-collection instruments, and then moves on to the actual collection of data. Once collected, data are processed, coded, edited, imputation is performed, weights are calculated, and the data are aggregated.

Up to this point (i.e. 5.6), the GSBPM has been concerned with the collection and processing of micro-data (at least from the perspective of an NSO – from the perspective of a supra-national organization, the collected data may themselves often be aggregates.)

For this example, we show how SDMX can be used from the point of aggregation forward, as we move through the GSBPM. For our purposes, then, we will focus on steps 5.7 and later, as shown below:

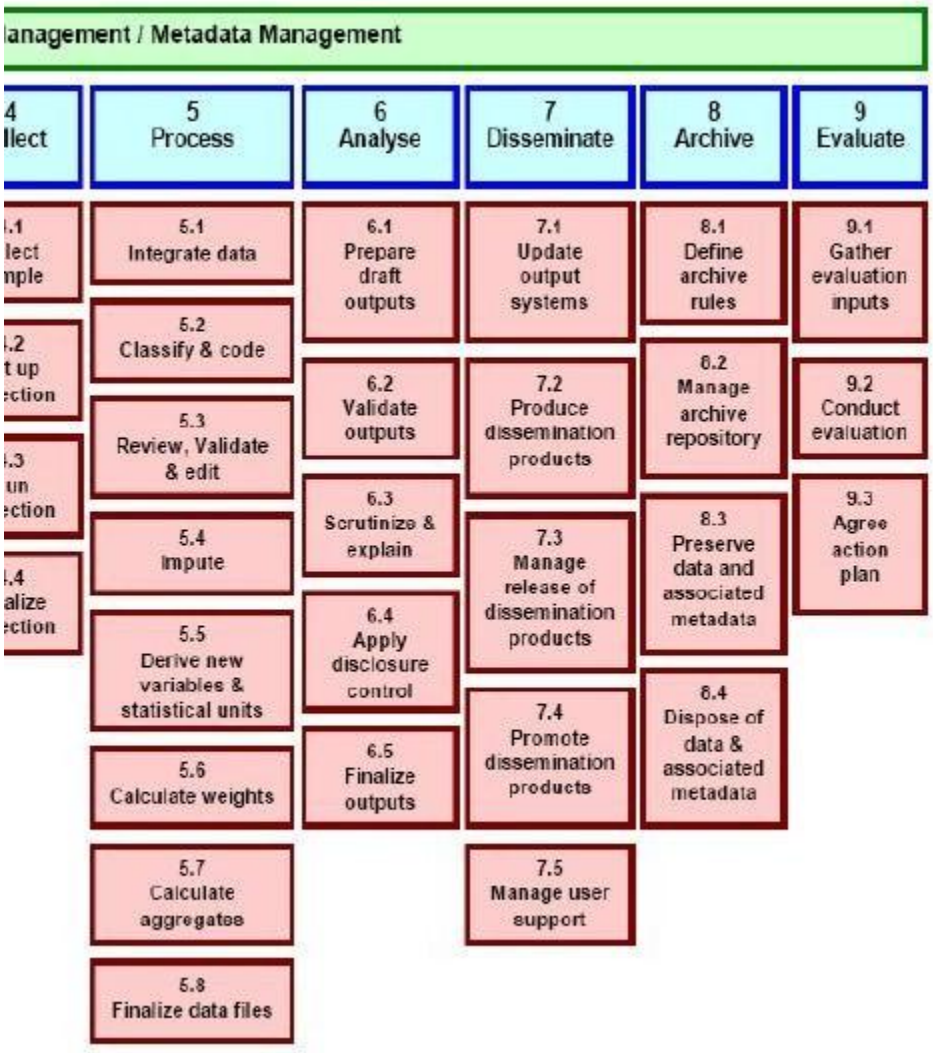


Figure 36: The part of the GSBPM supported by SDMX

To understand how SDMX can be used throughout this process, we need to look not only at the internal business process of the Central Bank or NSO, but also at the collection framework of the organization to which the aggregate data are reported.

Because SDMX focuses on the exchange of statistics, it will be necessary to consider the organization in our example which will be performing the collection. This will involve some constructs external to, but accessible by, the compiling organization – notably an SDMX Registry, with the various constructs that it contains (data flows, data providers, etc.).

SDMX is not only for reporting of aggregates, however – it also performs useful functions in the dissemination of data directly to users, and the archiving of data within the organization, so these functions of the organization will also be included in our example.

12.3 GSBPM and SDMX

12.3.1 Aggregation (Step 5.7), and Data Analysis (Step 6)

12.3.1.1 Calculation of Aggregates, and Understanding the SDMX Data Structure

Once data have been aggregated from the micro-data (Step 5.7 of the GSBPM) they will be stored in some format such as a relational database or data warehouse (Oracle, etc.) or in some processing format (SAS, SPSS), or in Excel spreadsheets or similar format. This will depend on the internal system and tools used within the organization, and is different in different organizations.

In order to capture the aggregates in a standard SDMX format, we must first look at the required data structures as dictated by the collecting agency. In our example, we will use the Effective Exchange Rates data structure developed by the ECB[4]. Below is an example of the type of data which is contained in a data set structured according to this DSD.

[4] ECB: European Central Bank

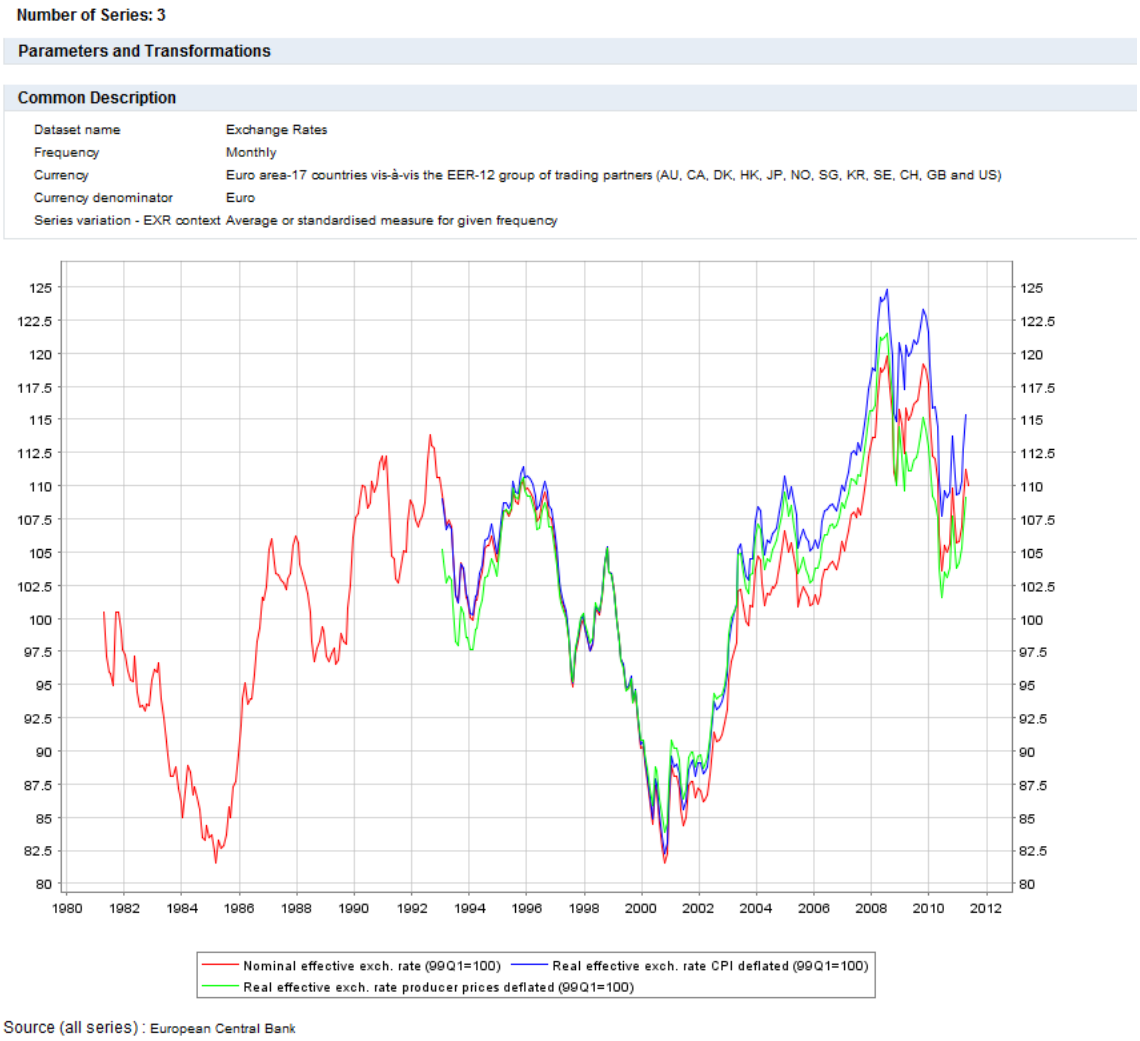


Figure 37: Example ECB Data of Effective Exchange Rates

It is not important from the NSO or Central Bank's perspective to understand every aspect of the analysis which went into the creation of the data structure, as the SDMX Data Structure Definition to be used for reporting will be provided by the agency collecting the data. All data reporters are expected to use the same data structure definition (DSD).

It is important to understand the data structure definition, because this is the resource which describes how the reported aggregates themselves must be structured. Below is view of the Effective Exchange Rates DSD.

The DSD is an XML file, created according to the SDMX-ML standard (it can also be expressed in SDMX-EDI, but the XML version is more common). The fact that is XML is important to the IT staff who must process it, and is used by many SDMX tools, but what is most important is that the statistical concepts and codelists (or "classifications") it uses

are also used in the reported aggregate data file. Thus, we do not look at the XML for this example – it is enough to know that the XML version of the DSD exists, and may be needed by tools or developers at some point.

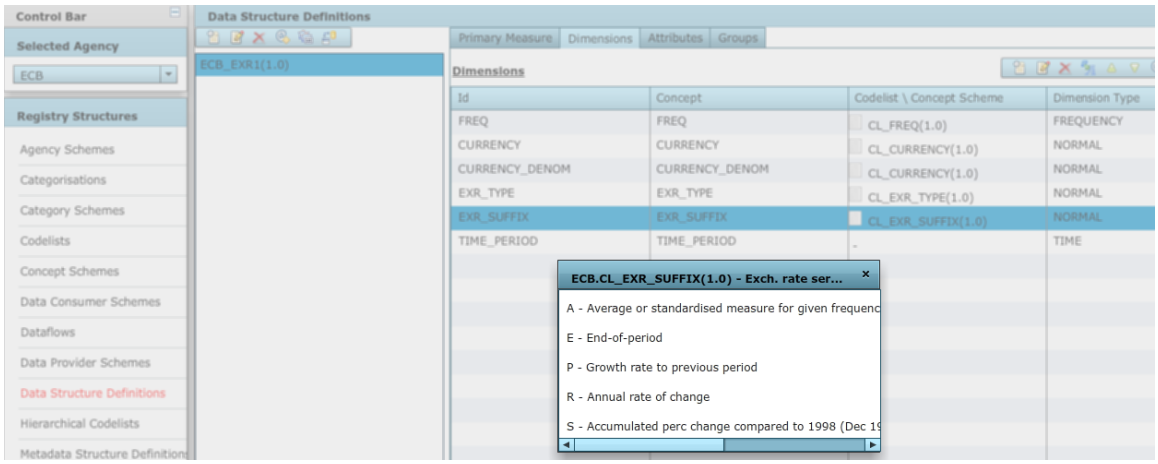


Figure 38: Example Data Structure Definition (Dimensions) for Effective Exchange Rates with Code List

Attributes

Id	Concept	Codelist	Attribute Relationship
COMPILATION	COMPILATION	-	Group (GROUP)
COVERAGE	COVERAGE	-	Group (GROUP)
DECIMALS	DECIMALS	-	Group (GROUP)
NAT_TITLE	NAT_TITLE	-	Group (GROUP)
SOURCE_AGENCY	SOURCE_AGENCY	CL_ORGANISATION(1.0)	Group (GROUP)
SOURCE_PUB	SOURCE_PUB	-	Group (GROUP)
TITLE	TITLE	-	Group (GROUP)
TITLE_COMPL	TITLE_COMPL	-	Group (GROUP)
UNIT	UNIT	CL_UNIT(1.0)	Group (GROUP)
UNIT_MULT	UNIT_MULT	CL_UNIT_MULT(1.0)	Group (GROUP)
BREAKS	BREAKS	-	Dimension Group
COLLECTION	COLLECTION	CL_COLLECTION(1.0)	Dimension Group
DOM_SER_IDS	DOM_SER_IDS	-	Dimension Group
PUBL_ECB	PUBL_ECB	-	Dimension Group
PUBL_MU	PUBL_MU	-	Dimension Group
PUBL_PUBLIC	PUBL_PUBLIC	-	Dimension Group
TIME_FORMAT	TIME_FORMAT	-	Dimension Group
UNIT_INDEX_BASE	UNIT_INDEX_BASE	-	Dimension Group
OBS_COM			

Figure 39: Example Data Structure Definition (Attributes) for Effective Exchange Rates

The view here shows a number of very important things:

The ID, Agency, and version of the DSD are displayed at the top of the screen. Below this, there is a listing of statistical concepts which are used as dimensions. (Frequency, Currency, Exchange Rate Type, etc.) Each of these concepts will be taken from some authoritative source, and descriptions and definitions can be obtained from the organization which publishes and maintains the DSD. In many cases, these concepts will be the standard concepts defined in the SDMX Cross-Domain Statistical Concepts document (which can be obtained at www.sdmx.org). In other cases, they may be formally defined and documented by the maintaining agency which publishes the DSD. It is important to understand the definition of these concepts, as they may be slightly different from those used by the NSO or Central Bank, but in most cases they will likely be very similar or the same as the concepts already used at the national level for this data.

The concepts used as Dimensions each take a value which has a standard representation. In most cases, this representation will be a codelist – a standard classification which must be used to identify and describe the observations. In the right-hand part of the screen, we can see which codelist is used to represent each concept used as a dimension. For example, the Frequency dimension uses a codelist called "CL_FREQ" version 1.0:

Frequency is perhaps the simplest example, as the reporting agency will generally know what the frequency of the data is, and have a record of this in their systems (quarterly, monthly, etc.)

Notice that the Time dimension is not coded, but instead has a time value, indicating the time of the observation.

The values for the codes may or may not match the classification used at the national level, and if they do not match then it will be necessary to map the codes used internally against the classification used by the SDMX DSD.

For each statistical concept used as a dimension, it must be possible to provide a value as specified by the SDMX DSD. This might seem like a lot of work, but it is done for obvious and important reasons – if the collected data are to be comparable at the supra-national level, then there must be a standard expression of the data, using the same statistical concepts and classifications to identify and describe the observations. This is no different than when reporting aggregate data today – each data collector will want to have a specific expression of the data collected. What is different, with SDMX, however, is that the data collectors are harmonizing the DSDs used in each domain, and there is an effort, through the SDMX Content-Oriented Guidelines, to use identical statistical concepts and representations where this is possible.

Harmonization of data is a difficult process, but it is one which will result, in time, in more useful data (because it is more comparable), and also hopefully in a more uniform collection of data at the national level, because all reporting countries must conform to a

standard DSD as they calculate aggregates, which in turn impacts the data collection process as shown in the GSBPM.

If we look again at the high-level picture of the DSD (above), we will also see a section which shows statistical concepts used as “Attributes”. These are descriptive concepts, sometimes represented with codes, or sometimes with strings. They are different from Dimensional concepts, because they are not always required – in the table © indicates “Conditional” and Ⓜ indicates “Mandatory”.

An Attribute also has a relationship “Attribute Relationship” to a construct such as a group of Dimensions, which can be one or more Dimensions, Observation etc. as discussed in Chapter 4.

In other ways, the Attributes are very similar to the Dimensions of the DSD – the coding (if they are coded) must be standard, as dictated by the DSD, and for the same reasons.

12.3.1.2 Formatting Aggregates with SDMX

Once it has been determined that the aggregate data can be expressed as SDMX, according to the rules of the DSD, then we need to think about what is involved in actually creating the SDMX-ML format for the data. This is important because if we can express the aggregates as SDMX-ML, there are a number of tools which will become useful in performing later activities in the GSBPM.

There are several techniques for creating SDMX-ML data sets, and several choices will need to be made. First, there are several “flavours” of SDMX: SDMX-EDI (also known as GESMES/TS) and four types of SDMX-ML (the XML version). This is a technical consideration, and it is typically the case that the data collector will dictate exactly which format is wanted. The XML formats include “Generic”, and “Structure Specific”. Different organizations use different flavours of SDMX-ML, but it is important to note that there are free tools available which will allow for transformations between these different flavours.

These are technical considerations which should be left to the IT staff, so we will not go into them in depth here – the reasons for using one or another are most often purely IT-technical ones.

We do need to look at the practical options for creating the SDMX-ML files, however. The options are discussed in Chapter 4- Data and Metadata Creation and Reporting and the technical mechanism for achieving different outputs from a database is discussed in

Annex 4 – Data Reader and Data Writer Functions. There are several types of tools which will allow the formatting of aggregates into the correct form: tools based on Excel, tools which take the data from a relational database such as Oracle, and tools which work within processing tools such as SAS or PCAxis. Again, we will not look at the technical details of these tools, as this is an IT issue, but it is important to be aware that there are several different tools available. Eurostat provides a free tool for "data mapping" which is broadly useful, and PCAxis will have native SDMX support built into it in future versions. It should be noted that when working with processing applications such as SAS and SPSS, it is often the case that dedicated scripts will need to be written within those environments, as different national formats within those applications will require specific formatting scripts to produce SDMX-ML outputs.

12.3.1.3 SDMX and Analysis of Aggregates (Step 6)

It may not seem obvious that SDMX is relevant to the process of analysis of aggregates, but it can sometimes be very useful. This will depend on which tools are used by an NSO to perform these various steps. Because most systems work well with XML generally – and because SDMX-ML is one flavour of XML – SDMX can provide some useful functions as the aggregates are analyzed and further processed.

In the preparation of draft outputs (Step 6.1), it may be helpful to use any of the various visualization tools based on SDMX when looking at the data. Tools exist for doing graphical visualizations of the SDMX data, using modern technology packages such as the Flex code developed by the European Central Bank (http://code.google.com/p/flex-cb/). Other packages also exist, provided by various commercial vendors. Other free tools exist for producing Excel spreadsheets and HTML displays of the data.

Especially if files are passed between several individuals while the draft outputs are prepared, it may be useful to exchange the SDMX-ML file, so that different individuals can use different visualizations of the same data while performing this work.

The validation of outputs (Step 6.2) requires more than just data visualization, and it is here that SDMX-ML can provide some solid benefit. Some of the validation rules exist within the DSD, and these can be automatically checked using free SDMX data and metadata set tools, others exist within an SDMX Registry where cross references, versioning, and request for deletions are validated to ensure the integrity of the structural metadata.

What SDMX cannot validate is that the numbers reported are correct in terms of other values in the data set – that is, are they plausible values given the numbers reported in preceding periods, or in relation to other reported data. These are statistical issues that cannot be solved by SDMX-based technology, but which will require dedicated checks created by a statistician who understand the statistical issues.

Scrutinizing and explaining the aggregates (Step 6.3) is something which typically involves visualization of the data (as for Step 6.1) but may also include the creation of specific tabular views for inclusion in reports. The same tools which provide the ability to

visualize SDMX data may also allow for the creation of tabular views for use in reports (Excel tables, etc.) but this will vary based on the systems within each NSO or Central Bank.

There is nothing in SDMX which directly addresses **disclosure control** (Step 6.4) or the finalization of outputs (6.5), other than the use of visualization tools as described for earlier parts of Step 6. However, it should be noted that any corrections or edits to the data will need to be reflected in the SDMX-ML data to be reported. Depending on how the SDMX-ML is generated, this may involve going back to the tools and systems used to format the SDMX-ML in the first place, and making sure that the correct data are available in those tools for re-formatting as SDMX-ML.

12.3.2 Reporting/Dissemination (Step 7)

Step 7 of the GSBPM covers the process of dissemination in its broadest sense – that is, all users of the data are the target of this process step, including organizations which collect the aggregate data from NSOs and Central Banks. Thus, the GSBPM addresses reporting and dissemination as a single set of activities.

There are several types of data dissemination, and when we consider dissemination and reporting using the Internet this category is very broad. As we look at each sub-process in this step, we will need to consider this broad range of possibilities.

In addition to the sub-processes described by the GSBPM, we also need to consider one aspect of SDMX that potentially concerns all forms of reporting and dissemination, the SDMX Registry Services (see SDMX Registry/Repository).

The first sub-process in Step 7 is the **updating of output systems**. This involves taking the aggregates as prepared in Step 6, and loading them into whatever systems are used to drive dissemination. Typically, this will involve database systems (e.g. Oracle) and - if the same database is not used to drive Web dissemination – also loading data into whatever system drives the views of data on the Website.

SDMX can be used as a format for the **exchange of data between systems**, whether these systems are internal to an organization, or external, and thus it makes a good format for loading databases used in all types of dissemination. Further, because it is an XML format, SDMX-ML can be used as inputs to systems for creating HTML, PDF, Excel, and other output formats. An SDMX Registry can make the reporting of such data more automated by using the data registration mechanism supported by a registry. The benefit of such a system is that – once new data have been “registered” (see below), the data collector can come and simply query the service for the new data. This helps to ease the burden of data reporting.

This application of SDMX tends to be very technical – because XML is well-supported by many types of systems, it is useful also in loading the databases used to drive dissemination. The details of this application are not something we will explore in any detail here.

The next sub-processes in the GSBPM is the **preparation of outputs**, and the management of their release. This covers a wide variety of potential products based on the data: reports (typically printed and disseminated as PDF, combining tabular views of the aggregate data with explanatory text and analysis), HTML pages displayed on a Web-site, data downloads in various formats (Excel, CSV, etc.), and Web-based interfaces for querying the data, and for doing graphic visualizations, which may even be interactive.

SDMX can be used as the single XML format for the creation of all other dissemination products, at least for providing the tabular views of the data. (Obviously, websites have more than just data on them.) Again, this can be a very IT-technical topic, but it is important to understand that there are many good technologies for "styling" XML to produce other outputs, including all of the ones typically found on statistical websites.

SDMX is also directly useful in two ways: as a format for reporting to data collectors and as a direct download format. The use of SDMX as a download format has become increasingly popular, and in some cases has proven to be the most popular form of disseminated data available on Web-sites. Many users prefer this format because it is easy to process (being XML) and it is accompanied by rich metadata, including the structural metadata necessary for applications to process or visualize the data. Further, the format is predictable, allowing for easy use of the data coming from outside the organization.

It is worth noting that for many organizations, SDMX is being deployed using a Web service (such as those developed by ECB, IMF, and OECD). Such services allow for direct querying of the data sources, in SDMX-ML format, by any user allowed access to the service. Eurostat is currently developing a "Census Hub" Web service for querying the census data to be collected in 2011. Here, the data for each country remains in the database of the country and role of the "hub" is to broker a user query such that an "SDMX Query" is sent to each relevant database which responds in SDMX. The resultant responses are then combined by the hub.

The "advanced" use of SDMX – where an SDMX-capable database can create many dissemination products which transform the SDMX into other formats, and even in an on-demand fashion for Web dissemination – can greatly simplify the process of preparing dissemination outputs. Instead of having to produce several parallel forms of the data, having a single SDMX source means that, once loaded, print and PDF reports must be prepared, and static Web-pages created, but all other types of data dissemination are basically handled by systems which generate needed outputs (Excel, CSV, graphical visualizations, SDMX-ML) in an on-demand fashion.

The figure below illustrates the basic principle behind this type of SDMX use.

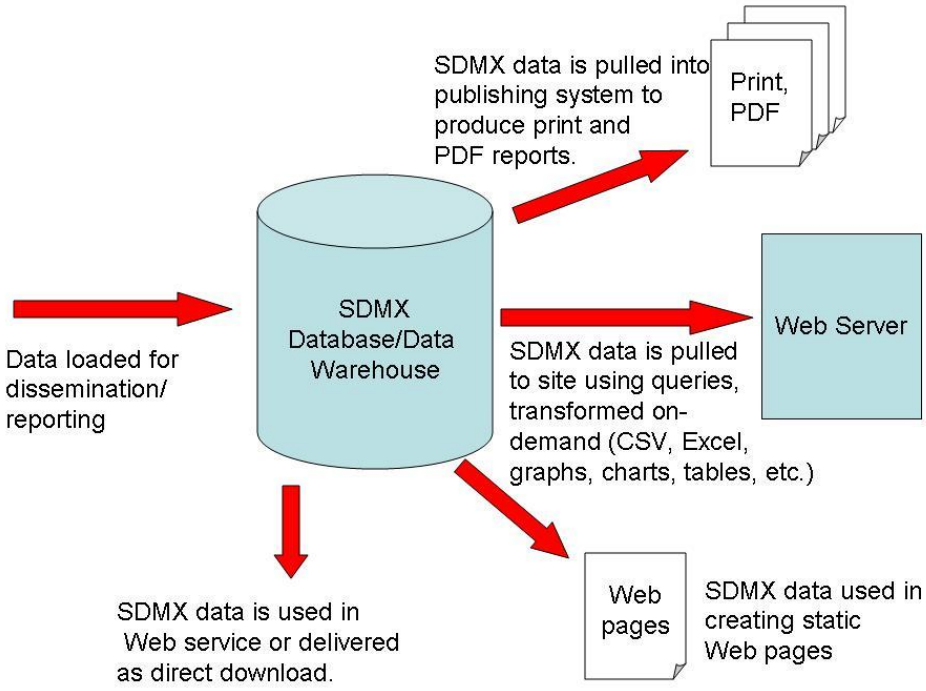


Figure 40: SDMX as the pivotal format in a dissemination system

It should be noted that when SDMX-ML represents a dissemination format in its own right, the SDMX-ML structure file containing the DSD and all its components should be provided along with the SDMX-ML data set it structures, as users will want both types of files for use in their own systems. In most cases, the DSD files will be available from their agency, but in this case a link to that source should be readily accessible to users (this may be through an SDMX Registry – see below).

Typically, all data products (including on-demand delivery via a Web service or query interface) are loaded into a "staging" environment, so that they can be subjected to quality assurance before being actually disseminated. SDMX does not change this aspect of the dissemination and reporting process, but does place an emphasis on the proper testing of Web-delivery for data.

The next sub-process in the GSBPM is the **promotion of dissemination** products. SDMX is extremely useful in this regard, although not perhaps in a fashion which is obvious. This process in the GSBPM is typically seen as the "advertising" of the statistical products, and SDMX is not much use here except that the use of leading-edge standards may offer some opportunities for promotion (presentations at conferences, etc.).

Far more interesting in increasing the visibility and use of data is the existence of the SDMX Registry Services, which provide a platform for the automatic discovery of data products. Users have become used to the idea that resources can be "Googled", and while the SDMX Registry services are not part of Google itself, they do provide a

focused way of searching for all of the data produced within a domain, regardless of which site the data is published on.

In essence, the SDMX Registry Services provide an online catalogue, listing all of the data available within a community. That community can be open or closed, depending on who is allowed access to the catalogue. Thus, there are registries today which only provide access to data collectors, such as the SDMX Registry used by the Joint External Debt Hub (it is only visible to the organizations which exchange data: the BIS, the IMF, OECD, and the World Bank). SDMX Registries can be public, however, which means that any Website or Internet-aware application could search for all of the data listed in that catalog, and then go to the site where that data is found.

This is a very powerful thing: increasingly, this approach to locating data is being used, because it leverages the latest generation of Web-based technology. Exposing the existence of your data to these types of sites and applications, and making it queryable or otherwise accessible in SDMX-ML format, is a very efficient way to make it visible and available to re-publishers and users of all types.

12.3.3 Archiving (Step 8)

SDMX is not specifically designed to support archiving, which is Step 8 of the GSBPM, but it is worth noting a few significant aspects of the standard which can be useful in this process. First, because SDMX-ML is XML, it provides a format which is not specific to any particular software package. Because of this, it can be used as a good archival format. Second, because SDMX has an XML expression of the structures in the DSD, it is possible to always understand how an SDMX-ML data set is structured, such that it can always be easily processed. Third, SDMX has strict rules about versioning. For archival use, this is good, because changes in the data sets and their structures over time can be recorded and stored.

Thus, while SDMX is not explicitly designed as an archival format, there are aspects to it which are very useful in this process.

12.3.4 The GSPBM and Other Relevant Aspects of SDMX

One feature of SDMX that should be mentioned is the ability to document standard statistical processes. This is done by describing a Process, which is made up of Process Steps, which may themselves have sub-steps. Each step or sub-step can have inputs and outputs, and can be named and described. A Process Step can link to another Process Step either as a hierarchy or by reference via a Transition. The Computation involved in a Process Step can be documented, including the actual software used in the Process Step. Note that there is no support yet in SDMX for specifying actual computations in a way that can be invoked by software.

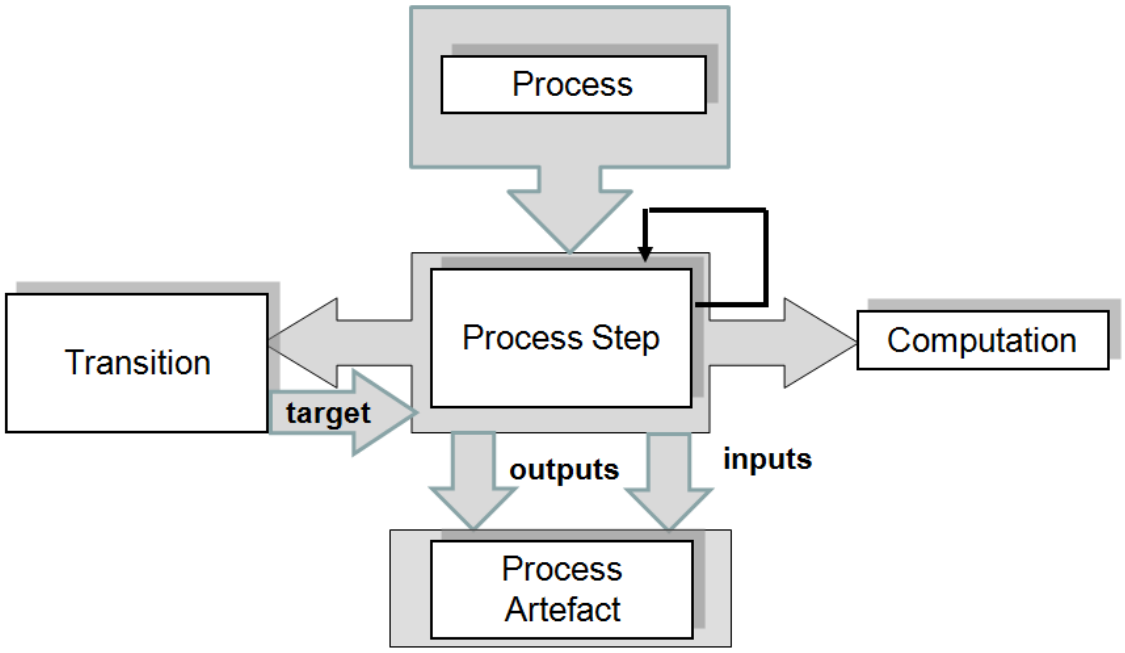


Figure 41: Schematic of Model for Process in SDMX

The process can be expressed in SDMX-ML, so that documentation can be produced in many useful formats, using the same types of transforms described earlier for disseminating statistical data sets. Thus, a PDF or HTML version of a process description could be generated from the XML.

It is easy to see that a particular organization could use the GSBPM as the basis of such a process, describe each input and output, and then send this to another organization, so that the exact processing of the data was clear.

There is currently no particular requirement from Eurostat or other data collectors for this functionality of SDMX, but it is being implemented by some NSOs internationally, for internal process descriptions being exchanged between departments within the organization. In future, this feature may be used between organizations as well.

12.4 Summary

Our example involves the micro-data coming from external sources being recoded and aggregated, with consequent reporting and dissemination of the tabulated data. To provide a view of how SDMX can be used in this scenario, the relevant parts of the GSBPM are highlighted below, and a summary table provided.

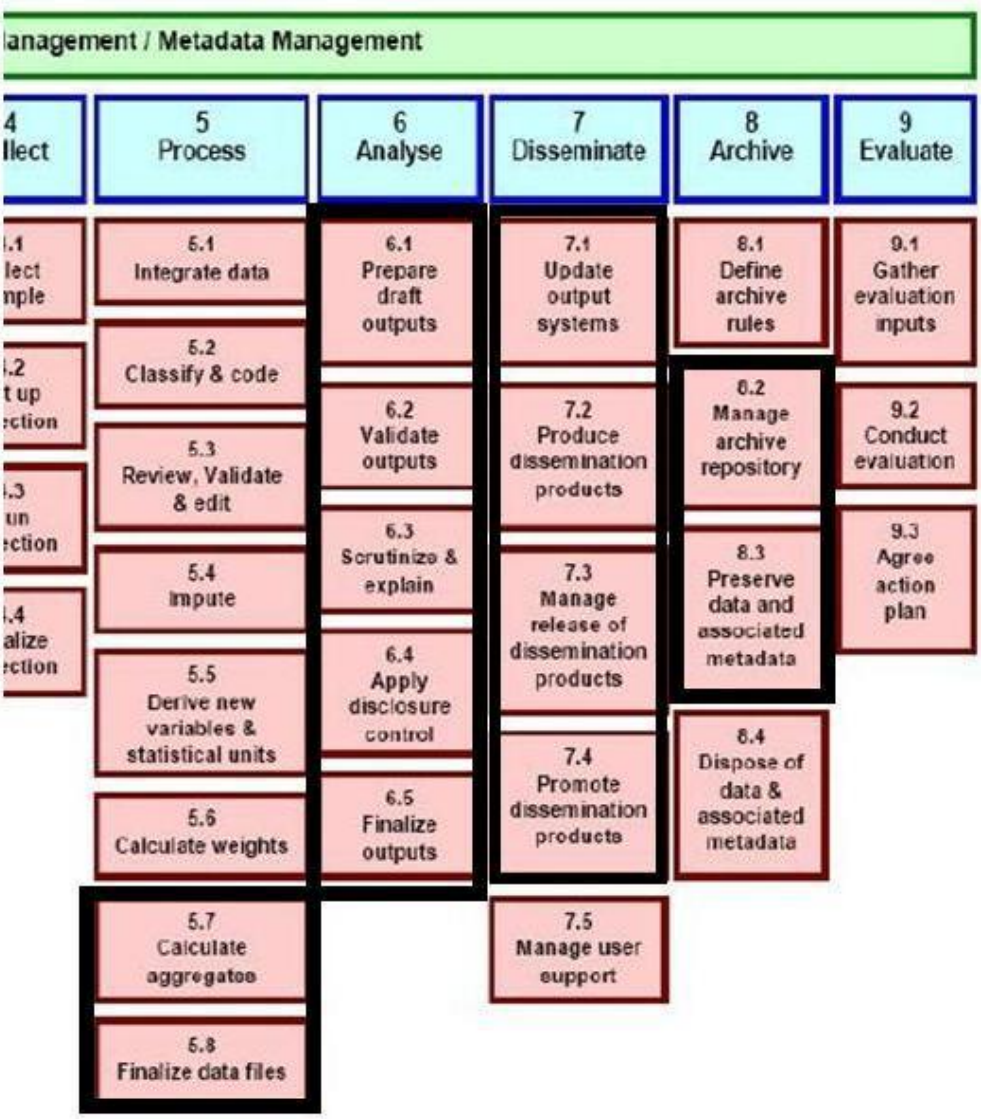


Figure 42: Summary showing the processes supported by SDMX

The table below summarizes each step in the GSBPM where SDMX is used in our scenario.

GSBPM Step	Use of SDMX	Notes
5.7 Calculate Aggregates	No direct use – may influence earlier steps in collection process	Derived variables and recodes must match the requirements of the standard DSD
5.8 Finalize Data Files	Use of SDMX-ML DSD and data formats to format aggregates	Used to pass data and structure to subsequent process steps
6.1 Prepare Draft Outputs	SDMX can help to visualize and process data, and is used as a source format for outputs	Relies on technologies which easily transform XML into other output formats

GSBPM Step	Use of SDMX	Notes
6.2 Validate Outputs	SDMX-ML provides validation of all rules in the DSD (correct codes, complete and valid descriptions and keys, etc.)	Some validation can be validated by XML schema (e.g. use of valid codes and dimension Ids), some validation can be undertaken with other SDMX constructs such as constraints, whilst some cannot be performed using SDMX structures e.g. comparison of numbers to determine plausibility
6.3 Scrutinize and Explain	SDMX visualizations may help to easily view data and generate views for output products	
6.4 Apply Disclosure Control	SDMX visualizations help to verify disclosure processing	Not a primary application of SDMX, which does not dictate anything about disclosure
6.5 Finalize Outputs	SDMX visualizations may provide views of data for final outputs; outputs may be generated on-demand for dissemination on Website, etc.	SDMX data must be updated if data are corrected
7.1 Update Output Systems	SDMX provides useful format for loading into output systems	Most technology tools and databases provide good support for XML formats such as SDMX-ML
7.2 Produce Dissemination Products	SDMX visualizations may provide views of data for final outputs; outputs may be generated on-demand for dissemination on Website, etc.	
7.3 Manage Release of Dissemination Products	SDMX serves as a format for reporting and dissemination to some users/data collectors; serves as basis for generation of other outputs, whether static or on-demand	
7.4 Promote Dissemination products	Use of SDMX Registry Services provides a high level of visibility for data	Depends on the availability of a domain registry for this purpose – requires that new data be registered automatically or manually
8.2 Manage Archive Repository	SDMX provides an easy format for generation of formats needed, based on the user demands on the archive; strict version control allows for explicit management of dependencies between data and metadata	
8.3 Preserve Data and Associated Metadata	Rich metadata and application/platform independence make SDMX a good archival format	

The benefits of using SDMX here are several:

- Standard data structures, statistical concepts and classifications, and formats make it easy to process and compare similar types of data from different national sources, both for data collectors and other users

- Richer dissemination format, complete with metadata, supports not only good visualization of data, but also allows easy downloading and use of data in internal systems

- SDMX-ML provides an excellent format for having a single source of data which can be easily transformed into different output formats for use

- Data becomes easier to find and use, through SDMX Registry Services, promoting the use of the data

- Data is archived in a long-lived format, independent of applications/platforms, and is accompanied by rich metadata, managed according to strict versioning rules

In some other scenarios, SDMX might also be useful as a data collection format, but in the case where micro-data are aggregated, the use of SDMX will be similar to that described here.

13 Annex 3 – Data and Metadata Samples

Accompanying this document is a set of sample files. These samples consists of structural metadata, data, reference metadata, and schema files for structured data and reference metadata. There are 4 basic sets of files. The details of each of these sets is described in the sections to follow.

13.1 Common Structures

Directory Name: common
Description: common structural metadata based on the content oriented guidelines
Contents:

Name	Description
common.xml	structural metadata, based on the content oriented guidelines; this is reused by many of the structural metadata files in the sample set
sdmx_common.xsd	XML schema file based on the structural metadata in common.xml; this is reused by many structure specific schemas in the sample set

13.2 ECB Exchange Rates

Directory Name: exr
Description: a set of 3 data structure variations and related data files demonstrating the various ways in which data can be organised
Contents: The contents are divided into 4 directories, each described in the sections to follow:

13.2.1 ECB Exchange Rates: Common Metadata

Directory Name: common
Description: common structural metadata utilized by each data structure definition
Contents:

Name	Description
common.xml	structural metadata; this is reused by the rest of the structural metadata files in the exchange rate sample set
sdmx_common.xsd	XML schema file based on the structural metadata in common.xml; this is reused by structure specific schemas in the exchange rate sample set

13.2.2 ECB Exchange Rates: No Group

Directory Name: ecb_exr_ng

Description: exchange rate structural metadata data and data with no groups defined; all attribute relationships are to specific dimensions
Contents:

Name	Description
ecb_exr_ng.xml	structural metadata, defining the data structure
ecb_exr_ng_full.xml	structural metadata, represents the same structure as ecb_exr_ng.xml, but does not use external referencing to make the file easier to load into tools which cannot resolve local file references
generic/ecb_exr_ng_flat.xml	data in the generic format, structured with all dimensions at the observation level
generic/ecb_exr_ng_ts.xml	data in the time series-specific generic format, structured with the time dimension at the observation level
generic/ecb_exr_ng_ts_gf.xml	data in the generic format, structured with the time dimension at the observation level
generic/ecb_exr_ng_xs.xml	data in the generic format, structured with the currency dimension at the observation level
structured/ecb_exr_ng_flat.xsd	structure specific data schema for data structured with all dimensions at the observation level
structured/ecb_exr_ng_ts.xsd	structure specific data schema for data structured with the time dimension at the observation level
structured/ecb_exr_ng_xs.xsd	structure specific data schema for data structured with the currency dimension at the observation level
structured/ecb_exr_ng_flat.xml	data in the structure specific format, structured with all dimensions at the observation level
structured/ecb_exr_ng_ts.xml	data in the structure specific format, structured with the time dimension at the observation level
structured/ecb_exr_ng_ts_gf.xml	data in the structure specific format, structured with the time dimension at the observation level
structured/ecb_exr_ng_xs.xml	data in the structure specific format, structured with the currency dimension at the observation level

13.2.3 ECB Exchange Rates: Sibling Group

Directory Name: ecb_exr_sg
Description: exchange rate structural metadata data and data with a sibling group defined; only one attribute references the sibling group

Contents:

Name	Description
ecb_exr_sg.xml	structural metadata, defining the data structure
ecb_exr_sg_full.xml	structural metadata, represents the same structure as ecb_exr_ng.xml, but does not use external referencing to make the file easier to load into tools which cannot resolve local file references
generic/ecb_exr_sg_flat.xml	data in the generic format, structured with all dimensions at the observation level
generic/ecb_exr_sg_ts.xml	data in the time series-specific generic format, structured with the time dimension at the observation level
generic/ecb_exr_sg_ts_gf.xml	data in the generic format, structured with the time dimension at the observation level
generic/ecb_exr_sg_xs.xml	data in the generic format, structured with the currency dimension at the observation level
structured/ecb_exr_sg_flat.xsd	structure specific data schema for data structured with all dimensions at the observation level
structured/ecb_exr_sg_ts.xsd	structure specific data schema for data structured with the time dimension at the observation level
structured/ecb_exr_sg_xs.xsd	structure specific data schema for data structured with the currency dimension at the observation level
structured/ecb_exr_sg_flat.xml	data in the structure specific format, structured with all dimensions at the observation level
structured/ecb_exr_sg_ts.xml	data in the structure specific format, structured with the time dimension at the observation level
structured/ecb_exr_sg_ts_gf.xml	data in the structure specific format, structured with the time dimension at the observation level
structured/ecb_exr_sg_xs.xml	data in the structure specific format, structured with the currency dimension at the observation level

13.2.4 ECB Exchange Rates: Rate Group

Directory Name: ecb_exr_rg
Description: exchange rate structural metadata data and data with a sibling group and rate group defined; only one attribute specifies a relationship with the rate group, but the dimensions from the previous sample now declare the sibling group as an attachment group

Contents:

Name	Description
ecb_exr_rg.xml	structural metadata, defining the data structure
ecb_exr_rg_full.xml	structural metadata, represents the same structure as ecb_exr_ng.xml, but does not use external referencing to make the file easier to load into tools which cannot resolve local file references
generic/ecb_exr_rg_flat.xml	data in the generic format, structured with all dimensions at the observation level
generic/ecb_exr_rg_ts.xml	data in the time series-specific generic format, structured with the time dimension at the observation level
generic/ecb_exr_rg_ts_gf.xml	data in the generic format, structured with the time dimension at the observation level
generic/ecb_exr_rg_xs.xml	data in the generic format, structured with the currency dimension at the observation level
structured/ecb_exr_rg_flat.xsd	structure specific data schema for data structured with all dimensions at the observation level
structured/ecb_exr_rg_ts.xsd	structure specific data schema for data structured with the time dimension at the observation level
structured/ecb_exr_rg_xs.xsd	structure specific data schema for data structured with the currency dimension at the observation level
structured/ecb_exr_rg_flat.xml	data in the structure specific format, structured with all dimensions at the observation level
structured/ecb_exr_rg_ts.xml	data in the structure specific format, structured with the time dimension at the observation level
structured/ecb_exr_rg_ts_gf.xml	data in the structure specific format, structured with the time dimension at the observation level
structured/ecb_exr_rg_xs.xml	data in the structure specific format, structured with the currency dimension at the observation level

13.3 Eurostat Demography

Directory Name: demography
Description: data and reference metadata, with accompanying structural metadata
Contents:

Name	Description
demography.xml	structural metadata, defining the data structure for demographic data
demography_xs_ex.xsd	structure specific data schema for demography data structured with the demographic measure dimension at the observation level
demography_xs.xml	data in the structure specific format, structured with the demographic measure dimension at the observation level
esms.xml	structural metadata, defining a much simplified variation of the Eurostat SDMX metadata structure; all of the new features of the metadata structure including the various target objects, the ability to reference multiple possible targets from a report structure and the ability to have more complex structures with repeating attributes, attributes with both sub content and values, and attributes with structured text as it content are included
esms.xsd	structure specific metadata schema for the Eurostat SDMX metadata structure
esms_generic.xml	sample reference metadata in the generic format, attached to the demography data, based on the Eurostat SDMX metadata structure
esms_structured.xml	sample reference metadata in the structure specific format, attached to the demography data, based on the Eurostat SDMX metadata structure
xhtml	XML schema files for XHTML - included for reference from the structure specific metadata schema

14 Annex 4 – Data Reader and Data Writer Functions

14.1 Schematic

The schematic below is that shown in Section 5.4.

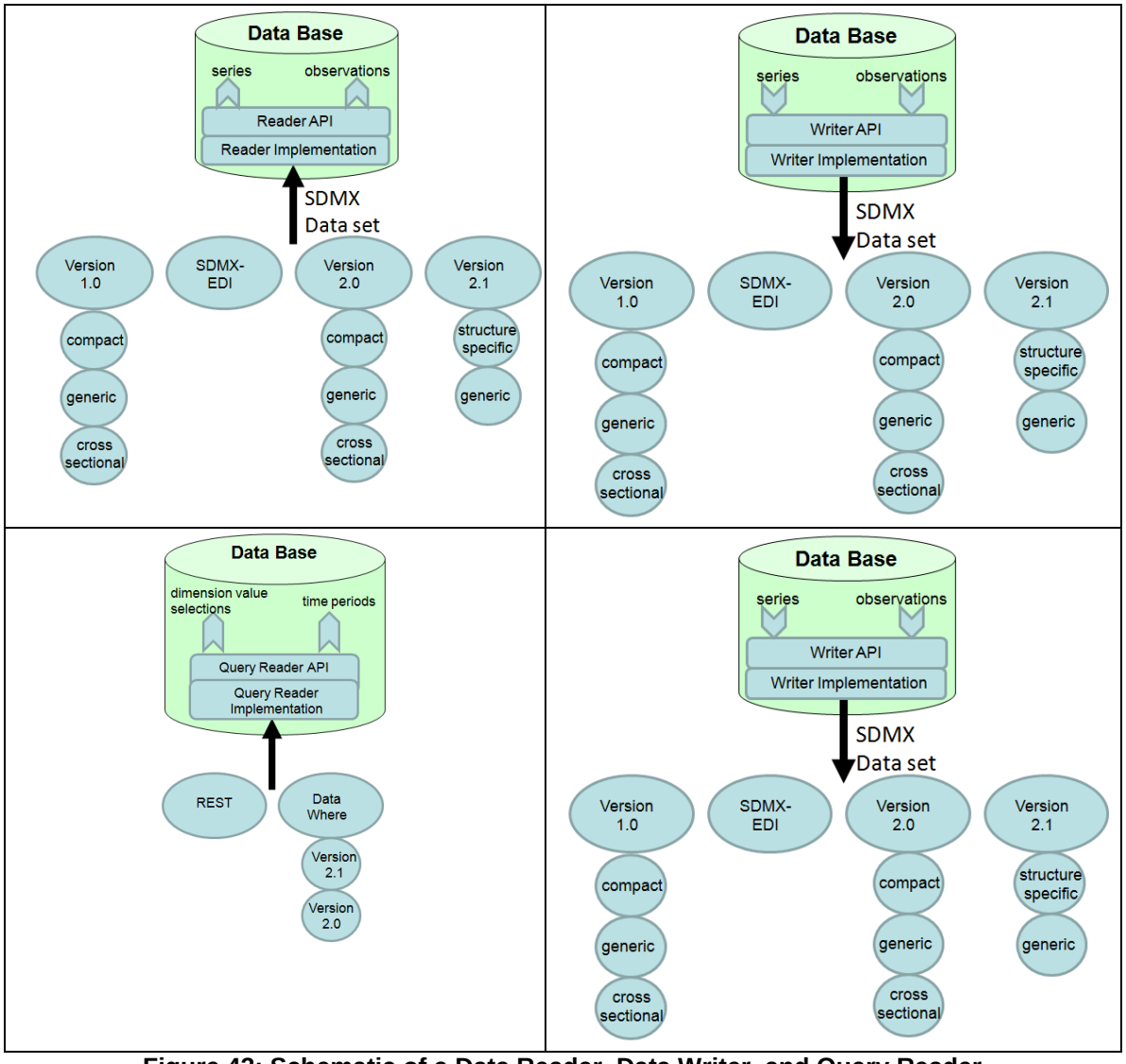


Figure 43: Schematic of a Data Reader, Data Writer, and Query Reader

14.2 Example Interfaces

Data Reader

Method Summary
copyToOutputStream(java.io.OutputStream outputStream) Copies the data that the reader is reading, to the output stream
getCurrentKey() Returns the current Keyable entry in the dataset, if there has been no initial call to moveNextKeyable, then null will be returned.
getCurrentObservation() Returns the current observation value for the current Keyable, if there are no more observations for the current Keyable then null will be returned.
getObsPosition() Returns the observation index within the key, starting at -1 (no observations read)
isTimeSeries() Returns true if the DataReaderEngine is reading time series data
moveNextKeyable() Returns true if there are any more keys in the dataset
moveNextObservation() If this reader is in a series, this will return true if the series has any more observation values.

Figure 44: Example of some Methods of the Data Reader Interface

Data Writer

Method Summary
startGroup(java.lang.String groupId) Starts a group with the given id, the subsequent calls to writeGroupKeyValue will write the id/values to this group.
startSeries() Starts a new series, closes off any existing series keys or attribute/observations.
writeAttributeValue(java.lang.String id, java.lang.String value) Writes an attribute value for the given id.
writeGroupKeyValue(java.lang.String id, java.lang.String value) Writes a group key value, for example 'Country' is 'France'.
writeObservation(java.lang.String obsIdValue, java.lang.String obsValue) Writes an Observation value against the current series.
writeSeriesKeyValue(java.lang.String id, java.lang.String value) Writes a series key value.

Figure 45: Example of some Methods of the Data Writer Interface

As you will see the only “methods” that the database application needs to invoke are based on the constructs in the SDMX Information Model (Group key, Series Key, Observation, Attribute). Each different “implementation” of this interface will enable a database (or, indeed, any other) application to read or write an SDMX data set in a specific type of data set message (e.g. SDMX-ML generic, SDMX-ML DSD-specific, SDMX-EDI). The important, and only, thing to understand from the Interface specification is that the database application never changes, no matter what is the input or output format. The format can even be CSV or mathematical format such as R or even a presentation format, depending on the implementation passed to the database application. The interface is the important asset here, and this is structured using the constructs of the SDMX Information Model: the database application need not be concerned with the actual format of the data.

Data Query Reader

Method Summary
getDataflow() Returns the Dataflow that the query is returning data for
getDataProvider() Returns the data provider(s) that the query is for, an empty list represents ALL
getDataQueryDetail() Returns the detail of the query
getDateFrom() Returns the start date to return data for Returns null if unspecified
getDateTo() Returns the end date to return data for Returns null if unspecified
getKeyFamily() Returns the Key Family (Data Structure Definition) that this query is returning data for
getSelectionsForConcept(java.lang.String dimensionId) Returns the selection(s) for the given dimension id, returns null if no selection exist for the dimension id

Figure 46: Example of some Methods of the Data Query Reader Interface

15 Annex 5 - Query Samples

15.1 Scope

This annex includes a set of example queries which walk through the process of querying a data source for the purposes of data discovery. It also includes an additional section which details some of the advanced features of the SDMX query message.

15.2 Discovering Categories

The first step a user may take in discovering data is to see the categories which data might be classified against. In this example scenario, a user is trying to find exchange rate data from the European Central Bank. The first step is to discover if there is a category for such data.

15.2.1 REST

In the REST syntax, there is no means to search based on text, so the simplest means for the user to find the exchange rate category is to retrieve all categories and filter through the results. Such a query would be structured as follows:

http://ws-entry-point/categoryscheme

A sample result for this can be seen in the sample file, ecb_query/ecb_exr_category_rest.xml.

Note that the entire category scheme is returned. If the data source had more category schemes, these would have been returned as well.

15.2.2 SOAP

Using the SOAP query, one can query for a category by name, and request that only the matched items be returned. The sample file, ecb_query/ecb_exr_category_soap.xml demonstrates a query for category schemes where a category contains the text "Exchange rate" in its English name. Note that it also request that only matched items and their children be returned. The result file for this query is ecb_query/ecb_exr_category_soap.xml. It can be seen from this result file that only the relevant categories are returned.

15.3 Discovering Data Structures

After the user queried for categories, he can now examine these categories in order to determine how to proceed with the data discovery process. Suppose that upon examination the user decide that he actually wanted data for the effective exchange rate category (2018773.2018810.2018779.2018795). The user will want to find data flows which are categorised against this category.

15.3.1 REST

In the REST syntax, the user will have to query for the category scheme in which the exchange rate is defined and request that categorisations and their references be returned. This query would be as follows:

http://ws-entry-point/categoryscheme/ECB/SDW_ECONOMIC_CONCEPTS/1.0?references="parentandsiblings"

Note the references parameter. It specifies parent (i.e. objects referencing the queried object) and siblings, objects referenced directly from the parent object. This will return categorisations which reference the categories of the queried category scheme and the objects which these categorisations categorise. The sample file, ecb_query/ecb_exr_dataflow_rest.xml shows the result of this query. Note that this assumes that there is only one data flow categorised against the one category. In reality, all categorisations and their objects for the entire category scheme would be returned.

15.3.2 SOAP

The SOAP syntax is able to take a much more direct approach. It can query directly for categorisations which use the specific effective exchange rate category, and only categorisation which categorise data flows. Further, it can omit the actual categorisation from the results, leaving only the dataflow to be returned. The sample file, ecb_query\ecb_exr_dataflow_query.xml shows this sample query and ecb_query\ecb_exr_dataflow_soap.xml shows the result.

15.4Discovering Data

Now that the dataflow has been found for the effective exchange rate data, the actual data reported against this flow can be retrieved.

15.4.1 REST

In the REST syntax, the data query is very straight forward for querying for an entire data set for a give data flow, in this case the effective exchange rates flow (2034482):

http://ws-entry-point/data/2034482

The result for this query is shown in the sample file ecb_query\ecb_exr_data.xml. Note that this result assumes that the REST request specified that it wished to receive structured specific data.

15.4.2 SOAP

The query in the SOAP syntax is just as simple, as shown in ecb_query/ecb_exr_data_query.xml. The request is for structure specific data, but does not specify which dimension to orient the data on. The implication is that the data source will return the data oriented with the dimension it chooses at the observation level. In this case, it is the time dimension.

The result for this query is the same as the REST and is shown in the sample file ecb_query\ecb_exr_data.xml.

Technically, the query for the dataflow was not actually necessary to discover the data using the SOAP query. It is possible to query for data directly by a category - with the result being any data in which the data set, the data structure, the data flow, or the provision agreement which is categorised against the referenced category.

15.5 Better Data Queries

The above example assumes a very simple scenario, where the data itself was not actually queried. In actuality, such a query may not be wise as the volume of data returned could be quite large. In a more realistic scenario, one might choose to first examine the data structure for the data flow in order to determine what data might be available. After the full data structure details are returned, one can examine the dimensionality of the data and even look at the available codes to do things such as search for exchange rates for only select currencies.

15.5.1 REST

To retrieve the full details of the data structure in REST based on the dataflow, one simply queries for the data flow and all descendant references:

http://ws-entry-point/dataflow/ECB/2034482/1.0?references="descendants"

The sample file ecb_query/ecb_exr_structure.xml shows the result of this query.

15.5.2 SOAP

The SOAP query is similarly simple, as show in the sample file ecb_query/ecb_exr_data_structure_query.xml. The sample file ecb_query/ecb_exr_structure.xml also shows the result of this query.

15.6 Other Structural Metadata Query Features

There is a sample set included with this document that serve to highlight some of the more advanced mechanisms in the structural metadata queries. These sample files are as follows.

Directory Name: query
Description: query messages and corresponding result files
Contents:

Name	Description
query_cl_all.xml	a query for a subset of the very large area code list; return details specify that the matched item should be cascaded down the hierarchy, meaning all of its child codes should be returned; results in all regions and sub-regions for Greece

repsonse_cl_all.xml	response for query_cl_all.xml
query_cl_regions.xml	query for only the regions that are direct children of Greece; uses the parent property of the code to find the administrative regions within the country; results are not cascaded, so only one level is returned
response_cl_regions.xml	response for query_cl_regions.xml
query_demo_stub.xml	query which demonstrates how one can check for the existence of an object by simply requesting that no references be resolved and only the stub be returned; intention of this query is to simply find out what is the version of the currently active demography data structure
response_demo_stub.xml	response for query_demo_stub.xml
query_esms_children.xml	query which demonstrates reference resolution and mixing return details; only objects directly referenced from the queries metadata structure are returned; the full details of the metadata structure are returned and only the stubs of the queried objects are returned
response_esms_children.xml	response for query_esms_children.xml
query_esms_descendants.xml	where used query; intention is to query for any objects which are used directly or indirectly by the ESMS metadata structure; metadata structure is requested to not be returned; only the stubs of all objects referenced form the metadata structure and the object which they reference (and so on) are returned
response_esms_descendants.xml	response for query_esms_descendants.xml

16 Annex 6: Worked Use Case

16.1 Scope

The scope of this Chapter is to show the SDMX-ML interactions with the Registry, database, and metadata repository for the use case scenario outlined in Chapter 3. The scenario is repeated below.

The SDMX-ML examples used are version 2.1.

16.2 Scenario

This scenario is the dissemination of data and related reference metadata using SDMX web services. It shows how the data, reference metadata, and structural metadata are used to build a web dissemination system.

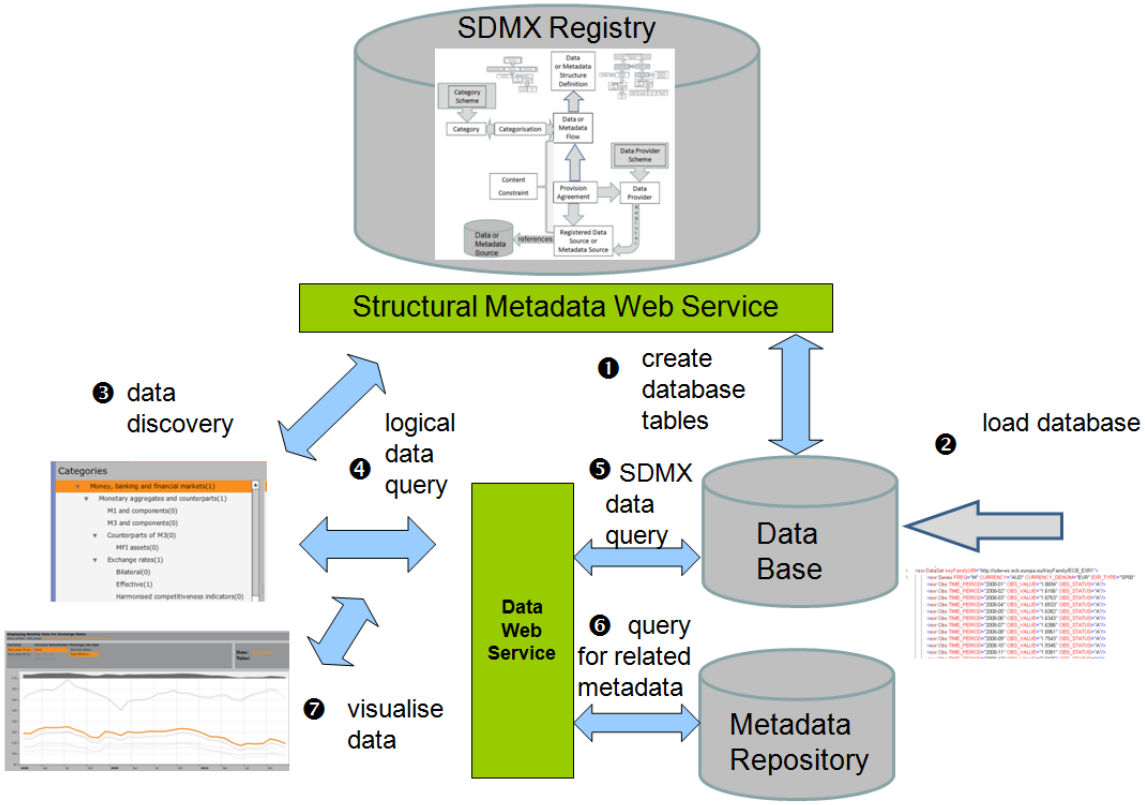


Figure 47: Process flow of an SDMX Web Data Dissemination System

Process	Description
❶	Retrieve the DSD from a structural metadata source (e.g. an SDMX Registry), and create database tables.
❷	Read an SDMX data set file and load the data into the database

Process	Description
❸	Data discovery system continually synchronises its metadata with the structural metadata source. A user makes a data selection from choices built from the information held in an SDMX Registry (structural metadata such as category scheme, dataflow, DSD, data provider, provision agreements and data registration)
❹	These choices are logical choices, built from the dimension selections.
❺	The logical choice is formatted as an SDMX data query. This is passed to the Data Base which responds with an SDMX data set.
❻	Reference metadata relevant to the data returned is retrieved from a metadata repository.
❼	The data and metadata are passed to a visualization tool to display the data in tables, charts, graphs, maps etc. Often a download is offered in various formats. The download options often include also the DSD or MSD.

16.3 Structural Metadata

16.3.1 Schematic

The following structural metadata and provisioning metadata is used in the scenario.

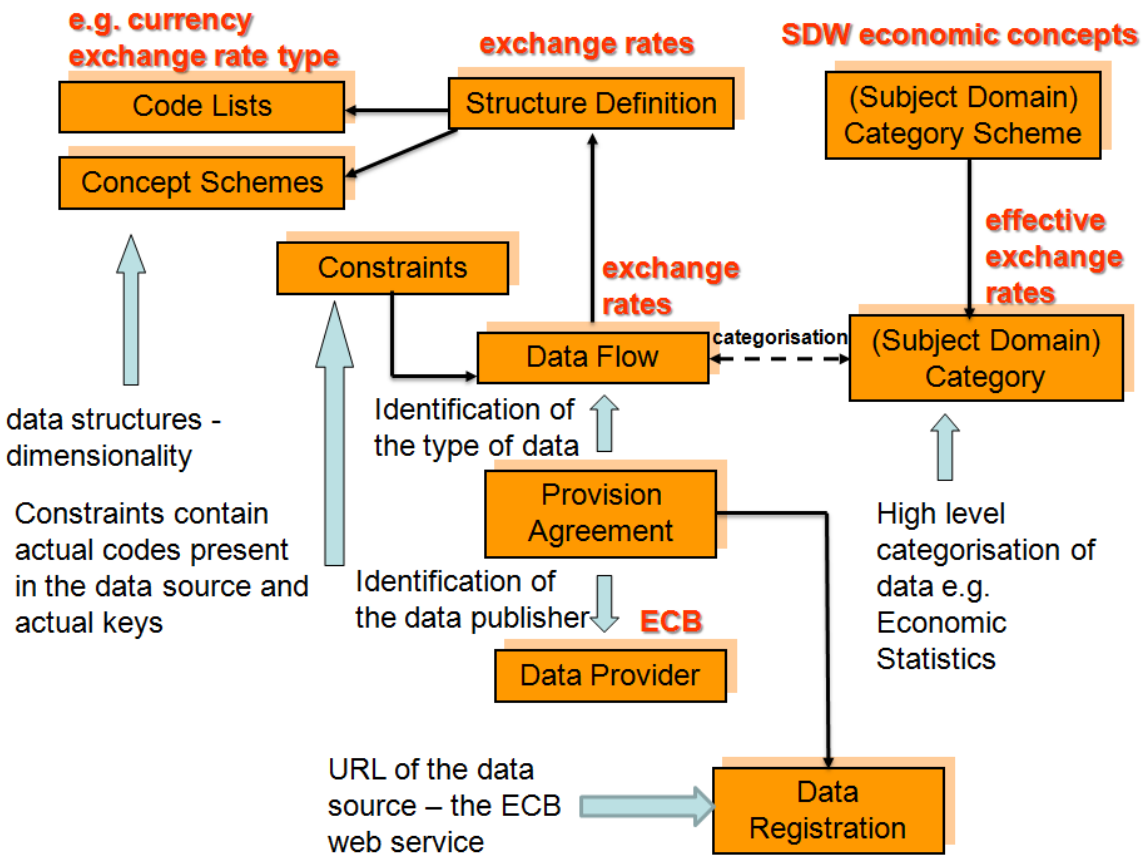


Figure 48: Structural and Provisioning Metadata Used in the Scenario

16.3.2 Data Structure Definition

For the reasons of clarity a sub set of the contents of the Code Lists and Concept Schemes are used in this example, and only a few of the DSD Attributes are used.

Dimensions

Dimensions

Id	Concept	Codelist \ Concept Scheme	Dimension Typ
FREQ	Frequency	Frequency code list(1.0)	FREQUENCY
CURRENCY	Currency	Currency code list(1.0)	NORMAL
CURRENCY_DENOM	Currency denominator	Currency code list(1.0)	NORMAL
EXR_TYPE	Exchange rate type	Exchange rate type code list	NORMAL
EXR_SUFFIX	Series variation - EXR context	Exch. rate series variation code	NORMAL
TIME_PERIOD	Time period or range	-	TIME

Attributes

Attributes

Id	Concept	Codelist	Attribute Relationsh
M DECIMALS	Decimals	Decimals code list(1.0)	Group (GROUP)
C TITLE	Title	-	Group (GROUP)
M UNIT	Unit	Unit code list(1.0)	Group (GROUP)
M UNIT_MULT	Unit multiplier	Unit multiplier code list(1.0)	Group (GROUP)
M COLLECTION	Collection indicator	Collection indicator code list(1.0)	Dimension Group
M TIME_FORMAT	Time format code	-	Dimension Group
C OBS_CONF	Observation confidentiality	Observation confidentiality code list(1.0)	Observation
M OBS_STATUS	Observation status	Observation status code list(1.0)	Observation

Group

Groups

Group Ids
Group

Dimension Reference
CURRENCY
CURRENCY_DENOM
EXR_TYPE
EXR_SUFFIX

Primary Measure

Primary Measure

Id	Concept Agency	Concept Scheme	Concept
OBS_VALUE	ECB	Default Scheme	Observation value

Text Type	-	Start Value	-	Decimals	-
Sequence	-	End Value	-	Pattern	-
Min Length	-	Interval	-		
Max Length	-	Time Interval	-		

16.4Scenario Steps

16.4.1 Create Data Base

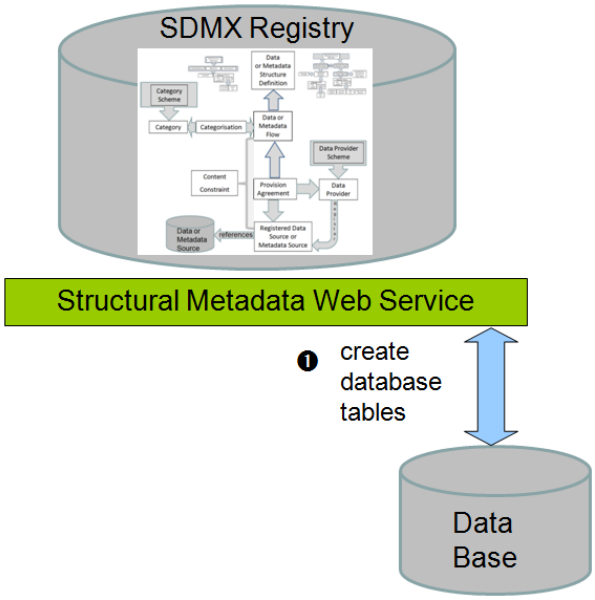


Figure 49: User Case - Create Database Tables from DSD

The following SDMX-ML is extracted from the Registry using the REST query

http:///[ws-entry-point]/DataStructure_ECB/ECB_EXR1

Dimensions

```
:str:DataStructure id="ECB_EXR1" agencyID="ECB" version="1.0" isFinal="false">
  <com:Name xml:lang="en">Exchange Rates</com:Name>
  <str:DataStructureComponents>
    <str:DimensionList id="DimensionDescriptor">
      <str:Dimension id="FREQ" position="1">
        <str:ConceptIdentity>
          <Ref id="FREQ" maintainableParentID="STANDALONE_CONCEPT_SCHEME" maintainableParentVersion="1.0" agencyID="ECB" package="conceptscheme" class="Concept"/>
        </str:ConceptIdentity>
        <str:LocalRepresentation>
          <str:Enumeration>
            <Ref id="CL_FREQ" version="1.0" agencyID="ECB" package="codelist" class="Codelist"/>
          </str:Enumeration>
        </str:LocalRepresentation>
      </str:Dimension>
      <str:Dimension id="CURRENCY" position="2">
      <str:Dimension id="CURRENCY_DENOM" position="3">
      <str:Dimension id="EXR_TYPE" position="4">
      <str:Dimension id="EXR_SUFFIX"  position="5">
      <str:TimeDimension id="TIME_PERIOD" urn="urn:sdmx:org.sdmx.infomodel.datastructure.TimeDimension=ECB:ECB_EXR1(1.0).DimensionDescriptor.TIME_PERIOD" position="6">
        <str:ConceptIdentity>
          <Ref id="TIME_PERIOD" maintainableParentID="STANDALONE_CONCEPT_SCHEME" maintainableParentVersion="1.0" agencyID="ECB" package="conceptscheme" class="Concept"/>
        </str:ConceptIdentity>
        <str:LocalRepresentation>
          <str:TextFormat/>
        </str:LocalRepresentation>
      </str:TimeDimension>
    </str:DimensionList>
```

Note that for brevity only the XML of the FREQ and TIME_PERIOD are shown in full.

Notes

1. A Dimension can state its position in the key, but the actual sequence is defined by the sequence in the `<DimensionList>` i.e. the inferred sequence in the `<DimensionList>` takes precedence over the value of the `position` attribute.
2. This DSD is converted from version 2.0. The version 2.0 Concepts were standalone in the form
   `<Concept agencyID="ECB" version="1.0"/>`
3. The standalone concept is not supported in 2.1 and any standalone concept is placed in a Concept Scheme with the Id "STANDALONE_CONCEPT_SCHEME" of the version corresponding to the version of the original standalone concept.

Group

```
<str:Group id="Group">
    <str:GroupDimension>
        <str:DimensionReference>
            <Ref id="CURRENCY"/>
        </str:DimensionReference>
    </str:GroupDimension>
    <str:GroupDimension>
        <str:DimensionReference>
            <Ref id="CURRENCY_DENOM"/>
        </str:DimensionReference>
    </str:GroupDimension>
    <str:GroupDimension>
        <str:DimensionReference>
            <Ref id="EXR_TYPE"/>
        </str:DimensionReference>
    </str:GroupDimension>
    <str:GroupDimension>
        <str:DimensionReference>
            <Ref id="EXR_SUFFIX"/>
        </str:DimensionReference>
    </str:GroupDimension>
</str:Group>
```

Notes:

1. A Group in version 2.1 plays the same role as it does in version 2.0 except that it does not group Series in the data set. The Group in version 2.1 is used solely as a mechanism to attach Attributes in the data set.
2. The Group in version 2.1 is retained for compatibility with version 2.0 and for avoiding repetition in the new Attribute Relationship construct which is introduced in version 2.1.

Attributes

```
<str:AttributeList id="AttributeDescriptor">
    <str:Attribute id="DECIMALS" assignmentStatus="Mandatory">
    <str:ConceptIdentity>
      <Ref id="DECIMALS" maintainableParentID="STANDALONE_CONCEPT_SCHEME"
      maintainableParentVersion="1.0" agencyID="ECB" package="conceptscheme" class="Concept"/>
    </str:ConceptIdentity>
    <str:LocalRepresentation>
      <str:Enumeration>
        <Ref id="CL_DECIMALS" version="1.0" agencyID="ECB" package="codelist" class="Codelist"/>
      </str:Enumeration>
    </str:LocalRepresentation>
    <str:AttributeRelationship>
      <str:Group>
        <Ref id="Group"/>
      </str:Group>
    </str:AttributeRelationship>
    </str:Attribute>
    <str:Attribute id="TITLE" assignmentStatus="Conditional">
  <str:Attribute id="UNIT" assignmentStatus="Mandatory">
  <str:Attribute id="UNIT_MULT" assignmentStatus="Mandatory">
  <str:Attribute id="COLLECTION"  assignmentStatus="Mandatory">
    <str:ConceptIdentity>
      <Ref id="COLLECTION" maintainableParentID="STANDALONE_CONCEPT_SCHEME"
      maintainableParentVersion="1.0" agencyID="ECB" package="conceptscheme" class="Concept"/>
    </str:ConceptIdentity>
    <str:LocalRepresentation>
      <str:Enumeration>
        <Ref id="CL_COLLECTION" version="1.0" agencyID="ECB" package="codelist" class="Codelist"/>
      </str:Enumeration>
    </str:LocalRepresentation>
    <str:AttributeRelationship>
      <str:Dimension>
        <Ref id="FREQ"/>
      </str:Dimension>
      <str:Dimension>
        <Ref id="CURRENCY"/>
      </str:Dimension>
      <str:Dimension>
        <Ref id="CURRENCY_DENOM"/>
      </str:Dimension>
      <str:Dimension>
        <Ref id="EXR_TYPE"/>
      </str:Dimension>
      <str:Dimension>
        <Ref id="EXR_SUFFIX"/>
      </str:Dimension>
    </str:AttributeRelationship>
  </str:Attribute>
```

```
<str:Attribute id="TIME_FORMAT"  assignmentStatus="Mandatory">
<str:Attribute id="OBS_CONF"  assignmentStatus="Conditional">
  <str:ConceptIdentity>
    <Ref id="OBS_CONF" maintainableParentID="STANDALONE_CONCEPT_SCHEME"
    maintainableParentVersion="1.0" agencyID="ECB" package="conceptscheme" class="Concept"/>
  </str:ConceptIdentity>
  <str:LocalRepresentation>
    <str:Enumeration>
      <Ref id="CL_OBS_CONF" version="1.0" agencyID="ECB" package="codelist" class="Codelist"/>
    </str:Enumeration>
  </str:LocalRepresentation>
  <str:AttributeRelationship>
    <str:PrimaryMeasure>
      <Ref id="OBS_VALUE"/>
    </str:PrimaryMeasure>
  </str:AttributeRelationship>
</str:Attribute>
<str:Attribute id="OBS_STATUS" assignmentStatus="Mandatory">
</str:AttributeList>
```

Note that for brevity only the XML of the DECIMALS (relationship to a group) COLLECTION (relationship to a set of Dimensions) and OBS_CONF (relationship to the Primary Measure) are shown in full.

Notes

1. The Attribute has a relationship to either a data set of one or more Dimensions as described in Chapter 4.
2. DECIMALS has a relationship to a Group with the Id of Group.
3. COLLECTION has a relationship to a set of Dimensions – in this case this is, in reality, the series key as this has been converted from version 2.0. It is possible for an attribute to have a relationship with just one, a few, or all of the Dimensions.
4. OBS_CONF has a relationship with the Primary Measure.

Measure

```
<str:MeasureList id="MeasureDescriptor">
  <str:PrimaryMeasure id="OBS_VALUE">
    <str:ConceptIdentity>
      <Ref id="OBS_VALUE" maintainableParentID="STANDALONE_CONCEPT_SCHEME"
      maintainableParentVersion="1.0" agencyID="ECB" package="conceptscheme" class="Concept"/>
    </str:ConceptIdentity>
  </str:PrimaryMeasure>
</str:MeasureList>
```

A typical database schema that can be set up using this DSD is shown below.

Series Key Table

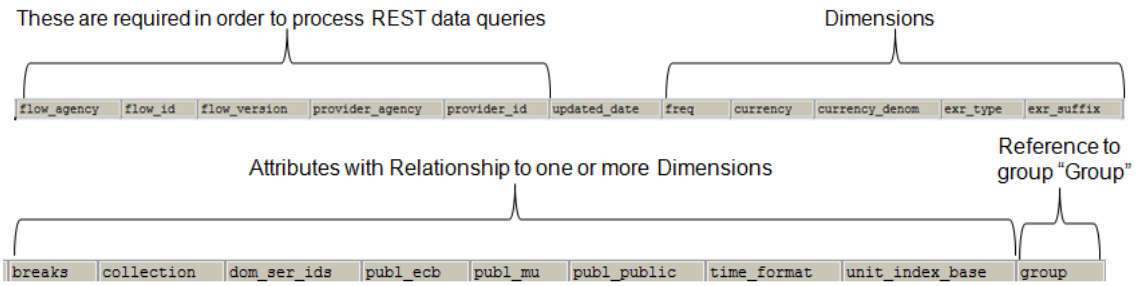


Figure 50: Relational Table Structure created from a DSD Dimensions and Related Attributes

Group "Group" Table

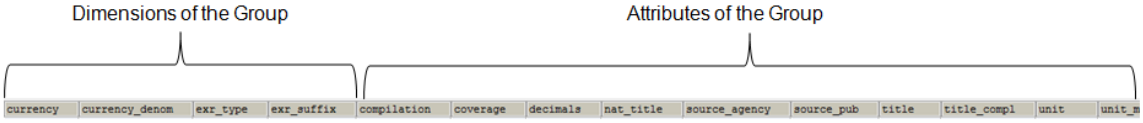


Figure 51: Relational Table Structure created from a DSD Group and Related Attributes

Observation

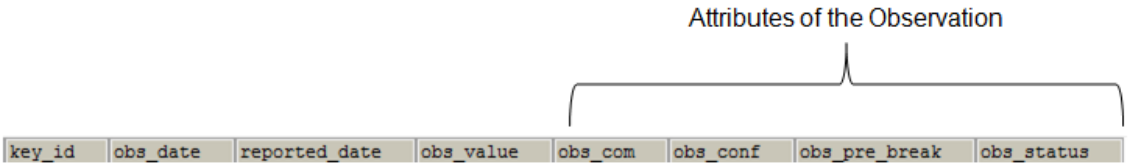


Figure 52: Relational Table Structure created from a DSD Primary Measure and Related Attributes

16.4.2 Load Database

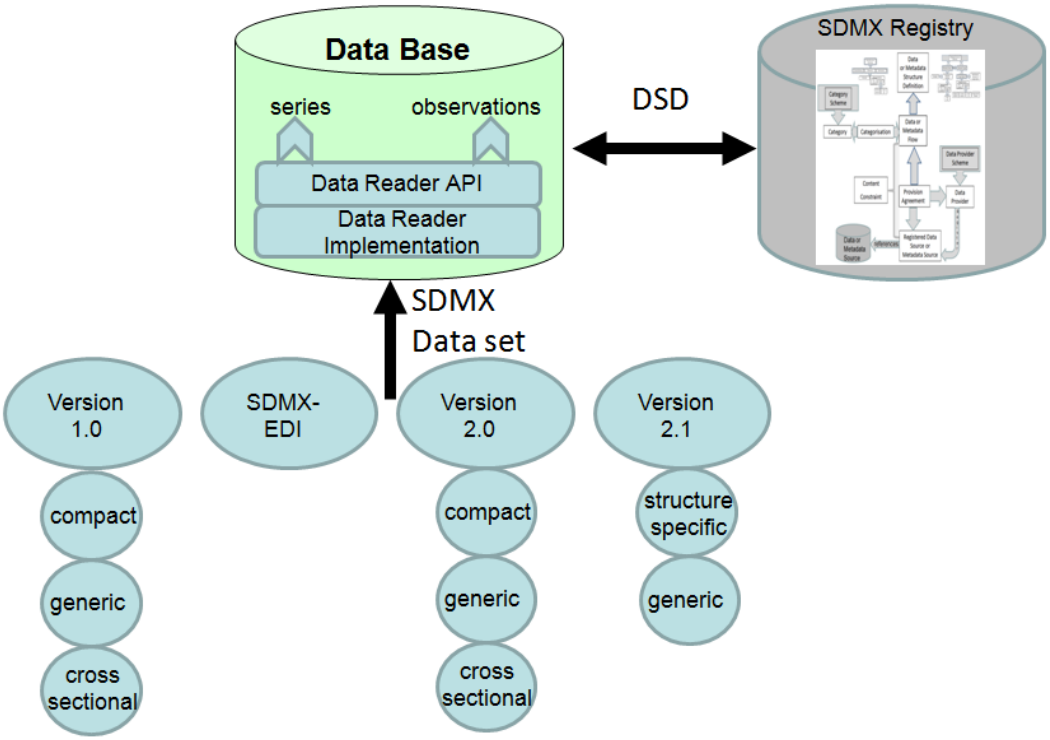


Figure 53: Loading an SDMX Dataset into a Database

The following is an extract from the SDMX dataset.

```
<message:StructureSpecificData xmlns:ns="urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ECB:ECB_EXR1(1.0)ObsLevelDim:TIME_PERIOD" xmlns:structuresp
   <message:Header>
      <message:ID>MT_1306412900750</message:ID>
      <message:Test>false</message:Test>
      <message:Prepared>2011-05-26T13:28:20</message:Prepared>
      <message:Sender id="MetadataTechnology"/>
      <message:Structure structureID="ECB_EXR1" namespace="urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ECB:ECB_EXR1(1.0)ObsLevelDim:TIME_PERIO
      dimensionAtObservation="TIME_PERIOD">
         <common:Structure>
            <Ref agencyID="ECB" id="ECB_EXR1" version="1.0"/>
         </common:Structure>
      </message:Structure>
   </message:Header>
   <message:DataSet dataScope="DataStructure" xsi:type="ns:DataSetType" structureRef="ECB_EXR1">
      <Group xsi:type="ns:Group" CURRENCY="Z08" CURRENCY_DENOM="EUR" EXR_TYPE="EN00" EXR_SUFFIX="A" UNIT_MULT="0" UNIT="POINTS" DECIMALS="4"
      <Group xsi:type="ns:Group" CURRENCY="Z08" CURRENCY_DENOM="EUR" EXR_TYPE="ERC0" EXR_SUFFIX="A" UNIT_MULT="0" UNIT="POINTS" DECIMALS="4"
      <Group xsi:type="ns:Group" CURRENCY="Z08" CURRENCY_DENOM="EUR" EXR_TYPE="ERD0" EXR_SUFFIX="A" UNIT_MULT="0" UNIT="POINTS" DECIMALS="4"
      <Group xsi:type="ns:Group" CURRENCY="Z08" CURRENCY_DENOM="EUR" EXR_TYPE="ERP0" EXR_SUFFIX="A" UNIT_MULT="0" UNIT="POINTS" DECIMALS="4"
<Series FREQ="A" CURRENCY="Z08" CURRENCY_DENOM="EUR" EXR_TYPE="EN00" EXR_SUFFIX="A" UNIT_INDEX_BASE="99Q1=100" COLLECTION="A" TIME_FORMAT="P1Y">
   <Obs TIME_PERIOD="1980" OBS_VALUE="NaN" OBS_STATUS="L" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1981" OBS_VALUE="97.001" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1982" OBS_VALUE="94.8242" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1983" OBS_VALUE="91.0767" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1984" OBS_VALUE="85.6634" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985" OBS_VALUE="84.4832" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1986" OBS_VALUE="96.7901" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1987" OBS_VALUE="104.1988" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1988" OBS_VALUE="100.7366" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1989" OBS_VALUE="99.0564" OBS_STATUS="A" OBS_CONF="F"/>
```

```
</Series>
<Series FREQ="D" CURRENCY="Z08" CURRENCY_DENOM="EUR" EXR_TYPE="EN00" EXR_SUFFIX="A" UNIT_INDEX_BASE="99Q1=100" COLLECTION="A" TIME_FORMAT="P1D">
   <Obs TIME_PERIOD="1985-01-02" OBS_VALUE="82.8815" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-03" OBS_VALUE="82.9576" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-04" OBS_VALUE="83.0586" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-07" OBS_VALUE="83.1025" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-08" OBS_VALUE="83.497" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-09" OBS_VALUE="83.3658" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-10" OBS_VALUE="83.63" OBS_STATUS="A" OBS_CONF="F"/>
   <Obs TIME_PERIOD="1985-01-11" OBS_VALUE="83.93" OBS_STATUS="A" OBS_CONF="F"/>
    </Series>
  </message:DataSet>
</message:StructureSpecificData>
```

Note that this is a sub set of the actual data message.

Notes

1. The reference to the DSD is given in the message header and given a local structureId
2. The local structureId (DSD) is referenced from the DataSet
3. A Group does not contain Series in version 2.1: it is used solely to declare attributes
4. The first <Obs> in the first <Series> has an OBS_VALUE of “NaN”. This is an XML expression that declared a value of “not a number”, thus allowing a “missing value” to be declared.
5. A dataset can contain observations from different frequencies.

The diagrams below show the database content based on the schema created.

Series Key Table

flow_agency	flow_id	flow_version	provider_agency	provider_id	updated_date	freq	currency	currency_denom	exr_type	exr_suffix	breaks	collection
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z08	EUR	EN00	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z08	EUR	ERC0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z08	EUR	ERD0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z08	EUR	ERP0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z08	EUR	ERU0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z08	EUR	ERU1	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z64	EUR	EN00	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z64	EUR	ERC0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z64	EUR	ERD0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z64	EUR	ERP0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z64	EUR	ERU0	A	(NULL)	A
ECB	2034482	1.0	ECB	ECB	2011-06-10 15	A	Z64	EUR	ERU1	A	(NULL)	A

Group Table

currency	currency_denom	exr_type	exr_suffix	compilation	coverage	decimals	nat_title	source_agency	source_pub	title	title_compl
Z08	EUR	EN00	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Nominal effective exch.
Z08	EUR	ERC0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z08	EUR	ERD0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z08	EUR	ERP0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z08	EUR	ERU0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z08	EUR	ERU1	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	EN00	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Nominal effective exch.
Z64	EUR	ERC0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERD0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERD0	P	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERD0	R	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERD0	S	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERP0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERU0	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat
Z64	EUR	ERU1	A	(NULL)	(NULL)	4	(NULL)	(NULL)	(NULL)	(NULL)	ECB Real effective exch. rat

Observation Table

key_id	obs_date	reported_date	obs_value	obs_com	obs_conf	obs_pre_break	obs_status
A:Z08:EUR:EN00:A	1980-12-31	1980	NaN	(NULL)	F	(NULL)	L
A:Z08:EUR:EN00:A	1981-12-31	1981	97.001	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1982-12-31	1982	94.8242	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1983-12-31	1983	91.0767	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1984-12-31	1984	85.6634	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1985-12-31	1985	84.4832	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1986-12-31	1986	96.7901	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1987-12-31	1987	104.1988	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1988-12-31	1988	100.7366	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1989-12-31	1989	99.0564	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1990-12-31	1990	109.9969	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1991-12-31	1991	106.6058	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1992-12-31	1992	110.3799	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1993-12-31	1993	104.9531	(NULL)	F	(NULL)	A
A:Z08:EUR:EN00:A	1994-12-31	1994	103.6886	(NULL)	F	(NULL)	A

The database is now available for query.

16.4.3 Register Data Source

The following registration is submitted to the Registry.

```
<mes:SubmitRegistrationsRequest>
  <registry:RegistrationRequest action="Replace">
    <registry:Registration>
      <registry:ProvisionAgreement>
        <Ref agencyID="ECB" id="ECB_EFFECTIVE_EX_RATES"/>
      </registry:ProvisionAgreement>
      <registry:Datasource>
      <registry:QueryableDataSource isWebServiceDatasource="false" isRESTDatasource="true">
        <common:DataURL>http://[ws-entry-point]</common:DataURL>
  <!-- note that this is not a valid URL. The atcual URL will contain the actual web service entry point -->
      </registry:QueryableDataSource>
      </registry:Datasource>
    </registry:Registration>
  </registry:RegistrationRequest>
</mes:SubmitRegistrationsRequest>
```

The SDMX web service of the database can now be searched for in the Registry.

16.4.4 Retrieve and Visualise Category Scheme and Dataflows

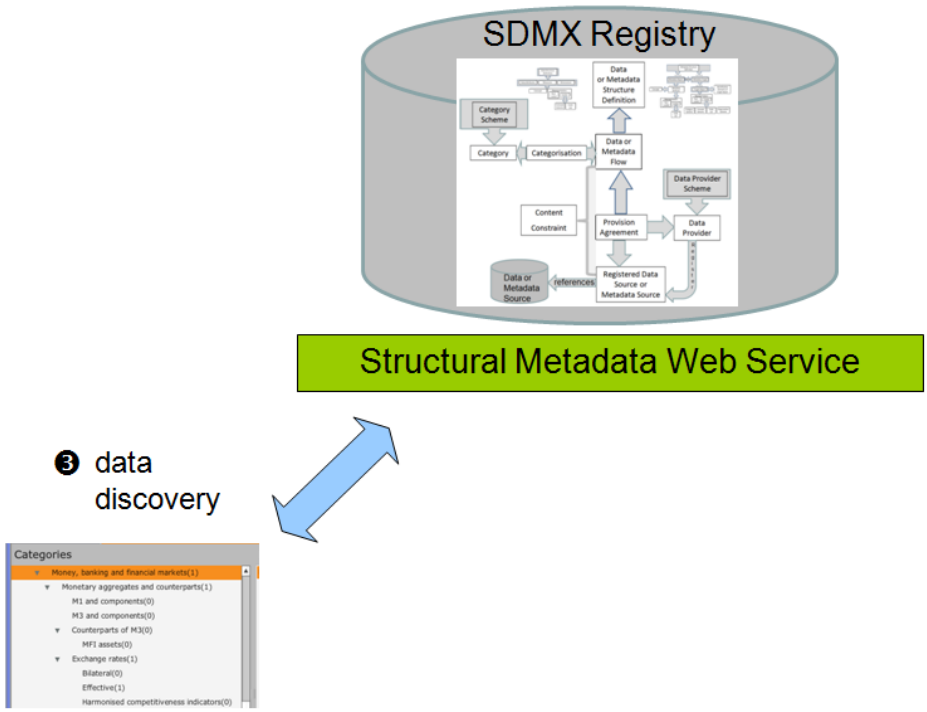


Figure 54: Building a Search Screen from Structural Metadata

The following REST query will return the Category Scheme and Categorisations that reference any of the Categories in the scheme.

http:// [ws-entry-point]/categoryscheme/ECB/SDW_ECONOMIC_CONCEPTS/1.0?references=parents

```
<str:CategoryScheme id="SDW_ECONOMIC_CONCEPTS" agencyID="ECB" version="1.0" isFinal="false">
  <com:Name xml:lang="en">SDW economic concepts</com:Name>
  <str:Category id="2018800">
    <com:Name xml:lang="en">Monetary operations</com:Name>
    <str:Category id="2018801">
      <com:Name xml:lang="en">Key interest rates</com:Name>
    </str:Category>
    <str:Category id="2018802">
      <com:Name xml:lang="en">Minimum reserves and liquidity</com:Name>
    </str:Category>
  </str:Category>
  <str:Category id="2018803">
    <com:Name xml:lang="en">Prices, output, demand and labour market</com:Name>
    <str:Category id="2120783">
      <com:Name xml:lang="en">Prices</com:Name>
      <str:Category id="2120778">
        <com:Name xml:lang="en">Consumer price indices</com:Name>
      </str:Category>
      <str:Category id="2120779">
        <com:Name xml:lang="en">Industrial producer prices</com:Name>
      </str:Category>
    </str:Category>
    <str:Category id="2120784">
      <com:Name xml:lang="en">Costs</com:Name>
      <str:Category id="2120785">
        <com:Name xml:lang="en">Compensation per employee</com:Name>
      </str:Category>
      <str:Category id="2120786">
        <com:Name xml:lang="en">Unit labour costs</com:Name>
      </str:Category>
    </str:Category>
    </str:Category>
    <str:Category id="2018773">
    <com:Name xml:lang="en">Money, banking and financial markets</com:Name>
    <str:Category id="2018810">
      <com:Name xml:lang="en">Monetary aggregates and counterparts</com:Name>
      <str:Category id="2018779">
        <com:Name xml:lang="en">Exchange rates</com:Name>
        <str:Category id="2018794">
          <com:Name xml:lang="en">Bilateral</com:Name>
        </str:Category>
        <str:Category id="2018795">
          <com:Name xml:lang="en">Effective</com:Name>
        </str:Category>
        <str:Category id="6374972">
          <com:Name xml:lang="en">Harmonised competitiveness indicators</com:Name>
        </str:Category>
      </str:Category>
```

```
<str:Categorisations>
<str:Categorisation id="-1624994409_-1114960412" agencyID="ECB" version="1.0">
  <com:Name xml:lang="en">Exchange Rates  - Effective exchange rates</com:Name>
  <str:Source>
    <Ref id="2034482" version="1.0" agencyID="ECB" package="datastructure" class="Dataflow"/>
  </str:Source>
  <str:Target>
    <Ref id="2018773.2018810.2018779.2018795" maintainableParentID="SDW_ECONOMIC_CONCEPTS"
    maintainableParentVersion="1.0" agencyID="ECB" package="categoryscheme" class="Category"/>
  </str:Target>
</str:Categorisation>
</str:Categorisations>
```

The application will now retrieve the Dataflows in the list of Categorisations – in this case there is only one – effective exchange rates.

The following REST query will return the dataflow.

http:// [ws-entry-point]/dataflow/ECB/2034482/1.0

```
<str:Dataflows>
  <str:Dataflow id="2034482" agencyID="ECB" version="1.0" isFinal="false">
    <com:Name xml:lang="en">Exchange Rates  - Effective exchange rates</com:Name>
    <str:Structure>
      <Ref id="ECB_EXR1" version="1.0" agencyID="ECB"/>
      <URN>urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ECB:ECB_EXR1(1.0)</URN>
    </str:Structure>
  </str:Dataflow>
</str:Dataflows>
```

The actual choreography of the query application is dependent upon the implementation but the following is assumed for this example.

1. Query application queries the Registry for the Provision Agreement.
2. Query application queries the Registry for data Registrations.
3. If there is a data source registered then the dataflow is displayed and can be selected.

The following REST query will return any provision agreements referenced from the dataflow

http://[ws-entry-point]/dataflow/ECB/2034482/1.0?references=provisionagreement

```
<mes:Structures>
  <str:Dataflows>
    <str:Dataflow id="2034482"  agencyID="ECB" version="1.0" isFinal="false">
      <com:Name xml:lang="en">Exchange Rates  - Effective exchange rates</com:Name>
      <str:Structure>
        <Ref id="ECB_EXR1" version="1.0" agencyID="ECB"/>
        <URN>urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ECB:ECB_EXR1(1.0)</URN>
      </str:Structure>
    </str:Dataflow>
  </str:Dataflows>
  <str:ProvisionAgreements>
    <str:ProvisionAgreement id="ECB_EFFECTIVE_EX_RATES" agencyID="ECB" version="1.0">
      <com:Name xml:lang="en">ECB Effective Exchange Rates</com:Name>
      <str:StructureUsage>
        <Ref id="2034482" version="1.0" agencyID="ECB" package="datastructure" class="Dataflow"/>
      </str:StructureUsage>
      <str:DataProvider>
        <Ref id="ECB" maintainableParentID="DATA_PROVIDERS" maintainableParentVersion="1.0"
        agencyID="ECB" package="base" class="DataProvider"/>
      </str:DataProvider>
    </str:ProvisionAgreement>
  </str:ProvisionAgreements>
</mes:Structures>
```

Notes

1. The <URN> contains the same identification information as the individual attributes in the <Ref> tag in the context of the <Dataflow> and <Structure> tags (the <Dataflow> identifies that the <Structure> is a DataStructure).

The following REST query will return the Data Provider Scheme that contains the Data Provider connected to the Provision Agreement. Note that this is not necessary in order to query for the data unless the Data Provider contains information required by the query application (such as the name of the Data Provider).

http://[ws-entry-point]/dataprovider/ECB

Notes

1. As an Agency can have only one Data Provider Scheme and this has a mandatory Id of DATA_PROVIDER_SCHEME and must be version 1.0 there is no need to specify any other parameters to the REST query.

This returns the following as part of the Data Provider Scheme.

```
<str:DataProvider id="ECB">
  <com:Name xml:lang="en">ECB</com:Name>
</str:DataProvider>
```

The following Registry query will return the REST data source referenced in a Registration for the Provision Agreement

```
<mes:QueryRegistrationRequest>
  <registry:QueryType>DataSets</registry:QueryType>
  <registry:ProvisionAgreement>
    <Ref agencyID="ECB" id="ECB_EFFECTIVE_EX_RATES"></Ref>
  </registry:ProvisionAgreement>
</mes:QueryRegistrationRequest>
```

The response from this query is:

```
<mes:QueryRegistrationResponse>
  <registry:StatusMessage status="Success "/>
  <registry:QueryResult timeSeriesMatch="false">
    <registry:DataResult>
      <registry:Registration>
        <registry:ProvisionAgreement>
          <Ref agencyID="ECB" id="ECB_EFFECTIVE_EX_RATES"></Ref>
        </registry:ProvisionAgreement>
        <registry:Datasource>
          <registry:QueryableDataSource isWebServiceDatasource="false" isRESTDatasource="true">
            <common:DataURL>http://[ws-entry-point]</common:DataURL>
  <!-- note that this is not a valid URL. The atcual URL will contain the actual web service entry point -->
          </registry:QueryableDataSource>
        </registry:Datasource>
      </registry:Registration>
    </registry:DataResult>
  </registry:QueryResult>
</mes:QueryRegistrationResponse>
```

The query application now has the details of the web service to call when making a query, and it also knows that the query expected uses the REST interface.

http://[ws-entry-point] /[RESTParameters]

Where [ws-entry-point] is the URL of the SDMX web service and [RESTParameters] are the query parameters.

The application can now visualize the information it has retrieved from the Registry in a way that is meaningful to the user. An example visualization is shown below.

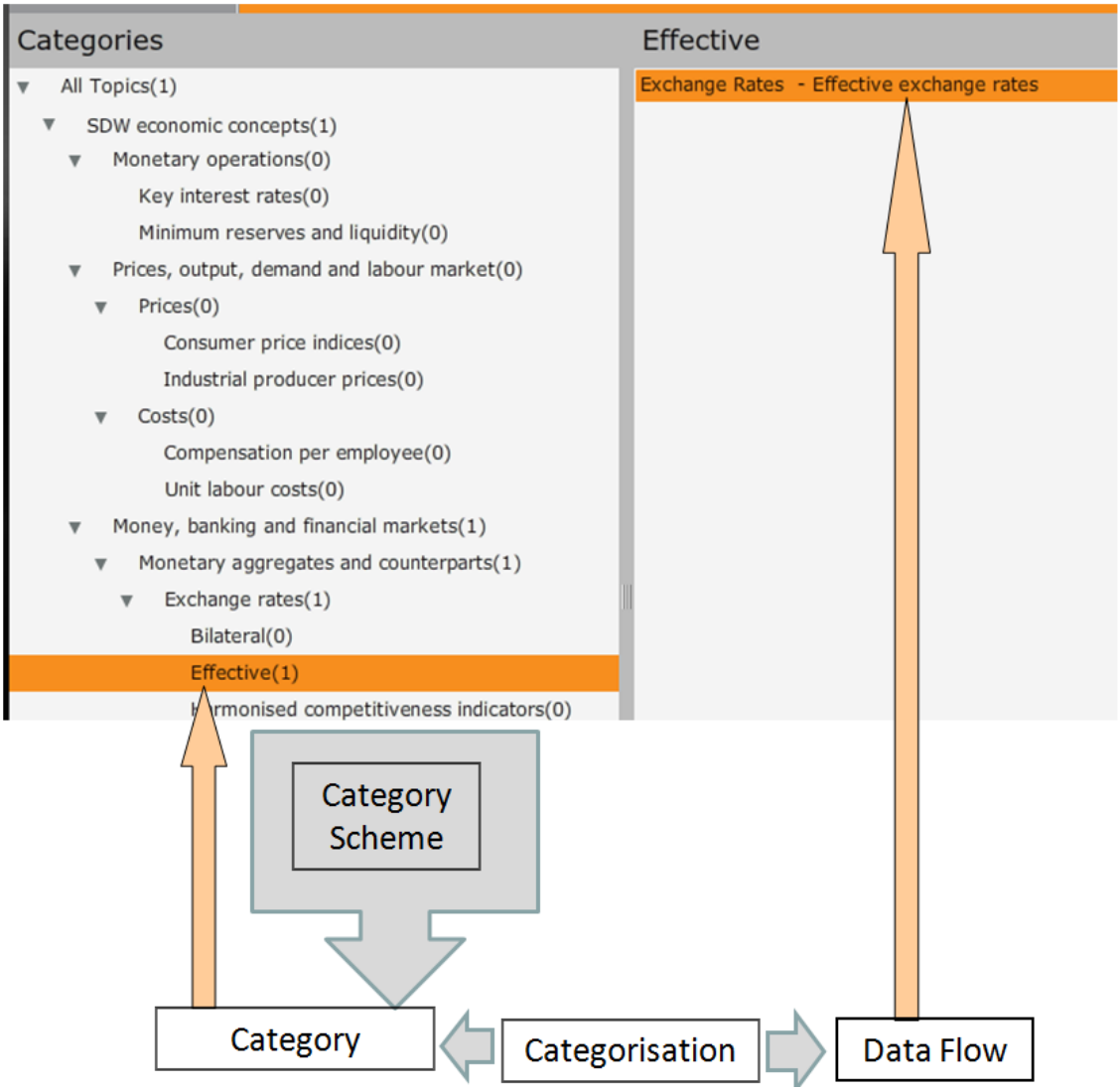
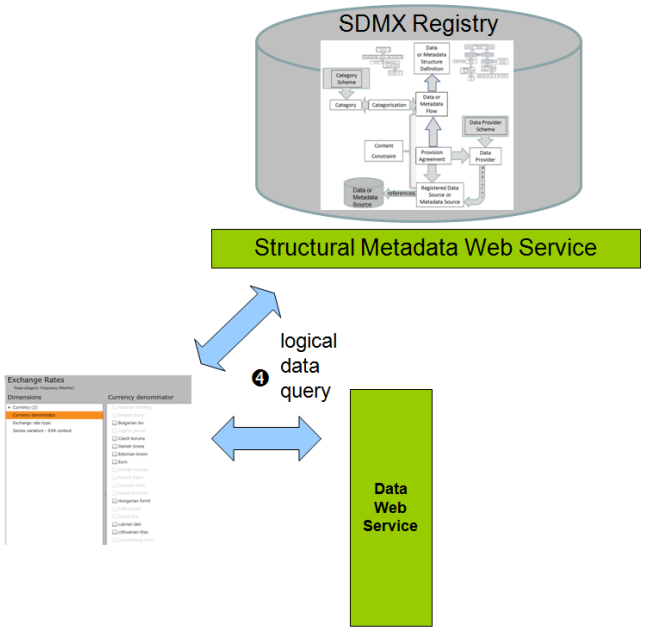


Figure 55: Example Query Screen Built from SDMX Category Scheme and Dataflows

The user will select the Category and this then shows the Dataflows that are available and which have data available for query (i.e. there is a Registration). There can be many such Dataflows but in this example there is only one.

16.4.5 Build Dimension Selection and Logical Query



Clicking on the Dataflow will cause the query application to query the Registry for the DSD associated to the Dataflow. The Dimensions and the relevant codes associated with the Dimensions are then shown for detailed data selections.

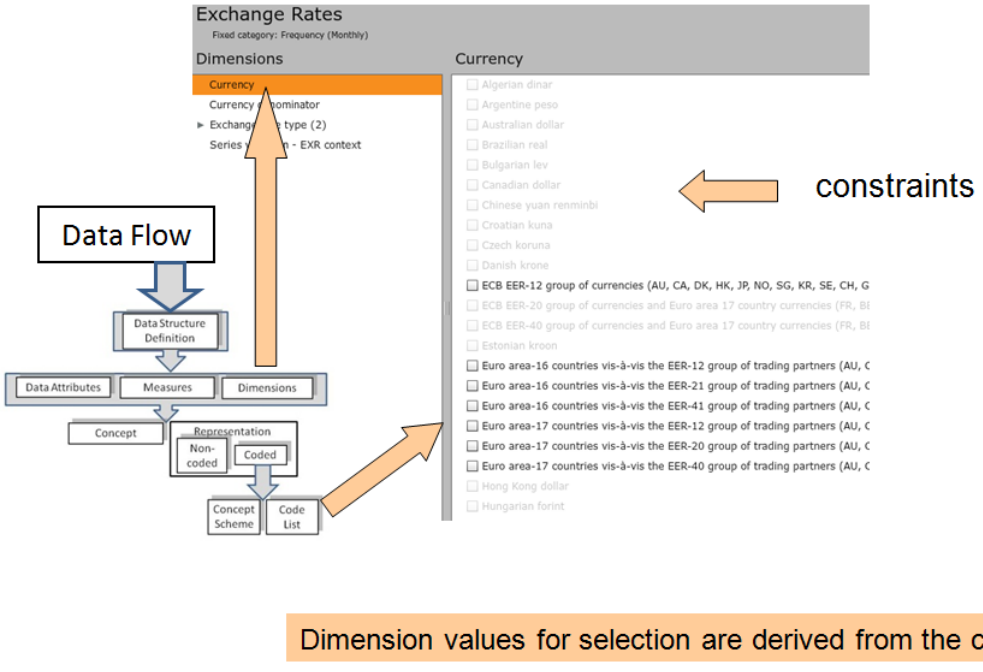
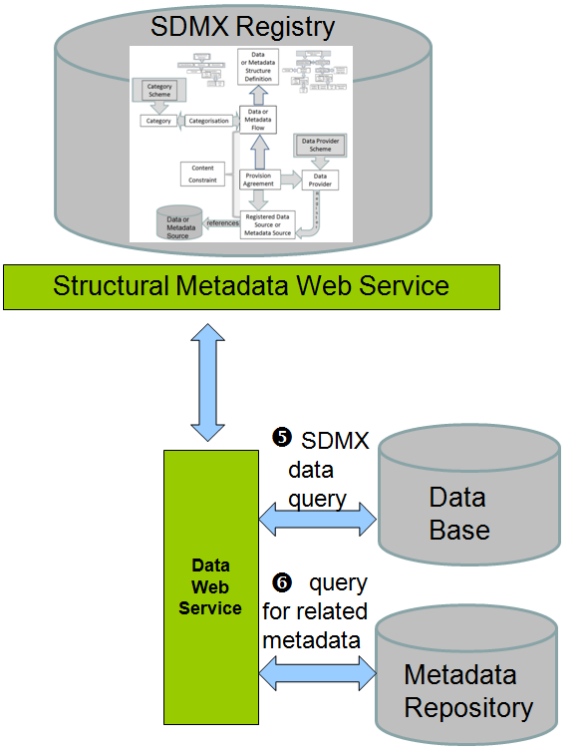


Figure 56: Selection Screen build from SDMX DSD

The user can click on each of the Dimensions and the relevant code selections are displayed. Note that in the example above the application is assumed to have queries for all the keys so that it can process them in order to grey-out the Dimension selections for which there is no data. These are greyed-out based on the current selections in the other Dimensions. This shows the importance of content constraints in a web dissemination system, as they can be used to guide the user to make only the selections that will return data. The same result can be achieved if these constraints are available for query (e.g. in a Registry), but this is a less effective way of processing if the contenrts of the database are dynamic.

16.4.6 Query Database



The logical query of the user is converted to an SDMX query and sent to the Data Web Service of the data publisher. The Data Web Service may also query the Metadata Repository in order to return both data and related metadata to the user application. The Data Web Service may also require structural metadata (DSD and MSD and related Code Lists and Concepts) from the structural metadata repository, in this example this an SDMX Registry.

Data Query Logical Selections

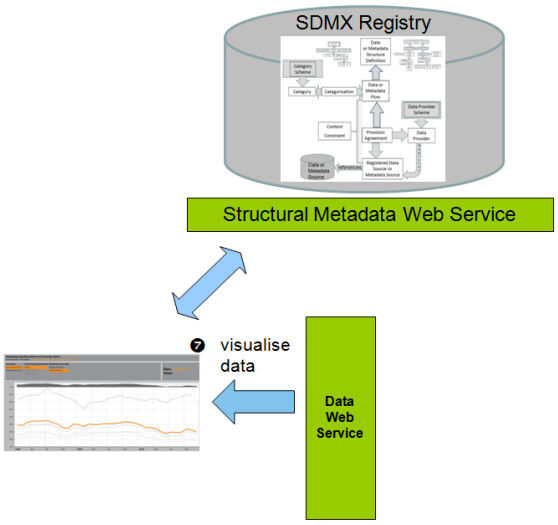
Dimension	Values
FREQ	M
CURRENCY	Z26,Z28
CURRENCY_DENOM	EUR,CZK,DKK

EXR-TYPE	DFCO,EN00
EXR_SUFFIX	A
Time	Jan 2008-Dec 2010

REST Query

http://ws-entry-point/data/2034482/M.Z26+Z28.EUR+CZK+DKK.DFC0+EN00+A/ECB?startPeriod=2008-01&endPeriod=2010-12&detail=dataonly

16.4.7 Visualise the Resultant Data Set



The returned data and metadata can be presented to the user in many ways – tables, graphs, charts, maps etc. SDMX does not support visualization directly but it is possible to define roles for a Dimension and Attribute in the DSD. For instance, it is possible to assign a role of “Geography” or “Title” or “Measure Unit” etc. which will aid a data visualization service to present this information in a meaningful way to the user.

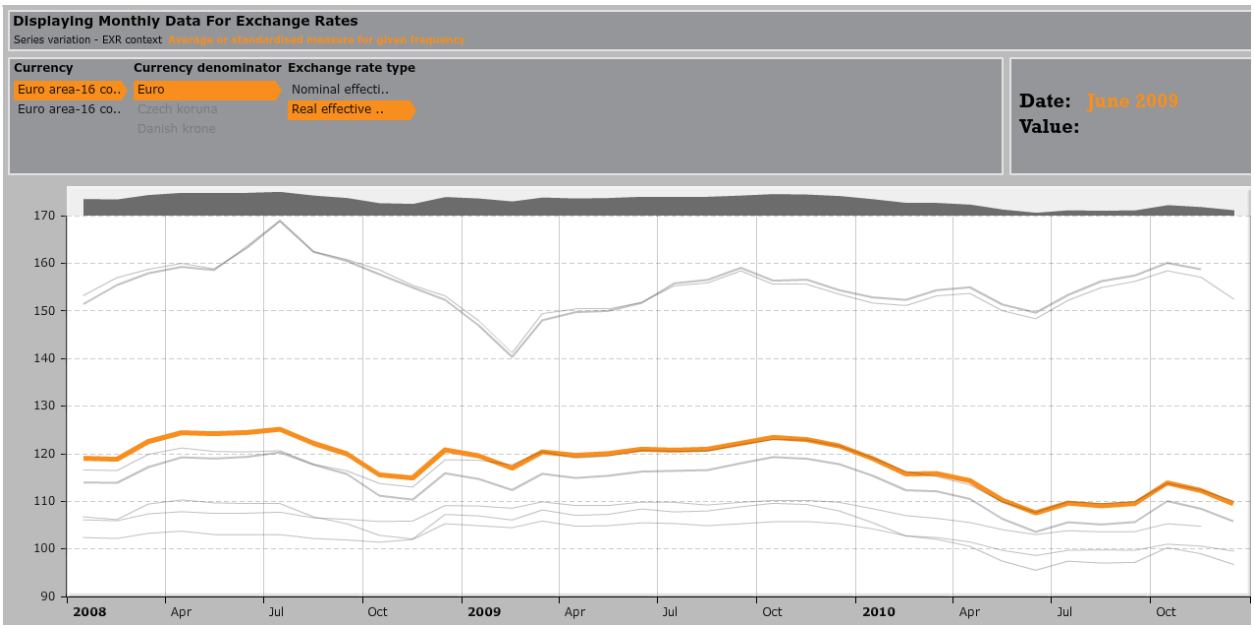


Figure 57: Example Graph Built from Data Returned from a Query