

SDMX Basics – An Introduction to the Use of ISO/TS 17369 - SDMX Technical Standards Version 1.0

Arofan Gregory, June 1, 2005

I. Introduction	2
II. SDMX and Data Structure Definition.....	2
III. Tasks	3
A. Create a New Structure Definition.....	3
B. Exchange/Obtain an Existing Structure Definition.....	4
C. View an Existing Structure Definition.....	5
D. Create XML Schemas for Specific Data Structure Definitions and Specific Use Cases.....	6
E. Create SDMX-ML Data - Simple Case.....	7
F. View SDMX-ML Data	8
G. Report, Collect, and/or Exchange Data between Systems	8
(1) SDMX-EDI.....	9
(2) SDMX-ML Generic	9
(3) SDMX-ML Compact	10
(4) SDMX-ML Utility	10
(5) SDMX-ML Cross-Sectional.....	10
H. Publish/Disseminate Data.....	11
I. Use Disseminated/Published Data	13
J. Knowing What an SDMX-Compliant Application is Capable of (Conformance Statements).....	13
IV. Other Topics.....	14
A. Notes on Application Development	14
B. Languages	14
C. Looking Ahead to Version 2.0.....	15
(1) Reference Metadata Structures and Reporting	15
(2) Hierarchical Code-lists and Cubes	15
(3) SDMX Registry Services	15
(4) Data Quality and Validation	15

I. Introduction

This document is aimed at the practical user of ISO/TS 17369 - the version 1.0 SDMX Technical Standards. It assumes some familiarity with the standard, and a willingness to reference the specifications. The intention is to provide a guide to how the SDMX version 1.0 standards can be used to achieve specific tasks.

The focus of this document is on the use of SDMX-ML formats, although it touches on the use of SDMX-EDI as well. Because the use of SDMX-ML and XML generally is less familiar to the potential users of this standard, this focus is seen as appropriate.

II. SDMX and Data Structure Definition

ISO/TS 17369 - the SDMX version 1.0 SDMX standards - place a strong emphasis on the use of an XML instance which describes the structure of data sets. This is the primary metadata content of SDMX messages: what is termed "structural metadata". Within SDMX this type of metadata is often referred to as "key families" - it consists of several types of structural metadata: the dimensions and attributes in the data, the code-lists which represent them, the concepts thus expressed, and the agencies which maintain these structural features are all contained in SDXM Structure Messages.

When we use the term "key family" in this document, we are referring to the complete data model for a single data set, expecting the reader to understand that this is shorthand for a complete structural definition for a data set.

It is important to understand that, because SDMX focuses on the exchange of statistical data between counterparties, all counterparties within an exchange community must use a single structural definition for an exchanged data set. One or more of the counterparties must define the structure of the data, and all must agree to use it. Within each community of data exchange, this question needs to be answered, but it may not always be answered the same way for all communities. It is assumed that whatever agency or group of agencies defines a data structure will also maintain it over time, but this is not necessarily the case either.

III. Tasks

In this section are described a set of typical tasks which users of ISO/TS 17369 - the SDMX version 1.0 standards - might wish to perform. This description includes the use of whatever free tools are available, and the possible use of other common tools. The freeware tools are documented separately, within the tools distribution package.

A. Create a New Structure Definition

Depending on agreements within the community of counterparties for a particular data set, any given counterparty may find themselves acting as the creator and maintainer of a data structure definition ("key family"). This is an activity which involves a job of analysis similar to data analysis activities which may be more familiar, such as the design of schemas for data warehouses, designing OLAP cube structures, etc.

In order to create a key family, it is necessary to have a clear idea of what the end product - the model of the data structure - consists of. To facilitate this understanding, there is a tutorial contained in the ISO/TS 17369 SDMX version 1.0 standards package, as an appendix to the Implementor's Guide. This document also contains a more complete definition of the SDMX Information Model, which might serve as a more advanced reference on this topic.

The definition of a data structure in SDMX - a key family - involves identifying what the dimensions of a multi-dimensional cube structure are, and then listing out for each the possible values it may contain. Documentation fields must also be decided on and defined, which are termed "attributes" in the data model. Further, the concepts associated with the dimensions and attributes must be identified and defined. This document will not go into detail about these activities, but it should be noted that some helpful suggestions on this activity are provided in the SDMX Implementor's Guide.

Very often, key-family design starts with an analysis of and agreement on the tables of data which are being exchanged. Once the tables are identified, they can serve as the basis for the work of analysis which follows. Other good inputs will include existing database structures of this information, especially if these contain cube-like, dimensional structures (OLAP, FAME databases, etc.).

It should be noted that, for cube structures with hierarchical value schemes, a reduction into flat code-list structures is necessary in ISO/TS 17369 SDMX version 1.0 (hierarchies will be supported in version 2.0 of the standards). The creation of key family structures is one of the most important - and difficult - tasks in the use of the SDMX standards. Thus, it is worth taking some care with this

activity, as all other aspects of the standard stem from the use of the data structure definition.

Structure definitions are expressed in an XML format - the SDMX Structure Message - or an EDI format. Within the ISO/TS 17369 SDMX version 1.0 package, these formats are detailed in the documents on SDMX-ML and SDMX-EDI. For SDMX-ML, all messages are contained in an XML instance with a common header section. The body of the message will be the element "Structure", followed by the common header, and then the declaration of all code-lists, concepts, dimensions, and attributes. A sample of this is included in the SDMX-ML package, "StructureSample.xml".

This file can be created in any XML-capable tool, but it is perhaps easiest to use a tool created for the specific purpose of designing key families. One such tool, implemented as a Microsoft Access database, is available from the SDMX Initiative. This tool provides a simple forms-based interface for designing key families, and it will create the XML data structure definition file automatically.

B. Exchange/Obtain an Existing Structure Definition

It is very often the case that a counterparty within the community which exchanges any given data set will not be the creator/maintainer of the structure definition, but instead will be expected to use an existing one. In this case, a copy of the data structure definition (the "key family") will need to be obtained, presumably from the counterparty which created it.

The first thing to realize is that it is important to have a copy of the formal description of the key family - the SDMX-ML Structure Message (or SDMX-EDI equivalent). It may also be useful to have copies of whatever documentation has resulted from the design process for the data structure definition, but it is absolutely critical to have the XML document.

When exchanging a key family with counterparties, the SDMX-ML Structure Message (or corresponding SDMX-EDI message) is a critical part of the exchange. When SDMX-ML data is published or exchanged, it is wise to make the SDMX-ML structure message available as part of the exchange, or on the same website - the data files reference the structure definitions, and can even provide a URL which points to them. Because of the importance of the data structure definitions in SDMX, it is assumed that users of SDMX-ML or SDMX-EDI data will have access to the structure definitions of that data in an XML or EDI form.

It should be noted that, if a structure definition is in the wrong format (that is, if you need it in SDMX-ML but have got it in SDMX-EDI, or vice versa) that there

are free tools available from the SDMX Initiative for transforming it into the corresponding format. For going from SDMX-ML to SDMX-EDI, this tool is an XSLT transformation, which can be run from the command line or from the SDMX transformation interface (a GUI for those who prefer point-and-click interfaces to command-line ones). For going from SDMX-EDI to SDMX-ML, there is a java tool which can be run from the command line.

It should be noted that it is possible to put the necessary definitions for a complete data structure into separate files: the code lists and concepts can be separated from the dimensions and attributes which use them. Thus, the full data structure definition may span several files, although this is not the usual packaging. To determine if this is the case, look at the key family declarations - the place where the dimensions and attributes are declared. If the code lists referenced by the dimensions and attributes are not present in the same file, then other files which describe them are needed. The same is true for the concepts referenced by the dimensions and attributes in the key family.

C. View an Existing Structure Definition

There are several ways to view an SDMX-ML structure definition file, although some are easier than others for the non-technical user. The most basic approach is to open up the structure definition file in a text editor or XML-capable web browser such as Microsoft's Internet Explorer. The view of the XML is very basic, however - it is just a display of the XML with no additional presentation. This makes it difficult to understand what you are looking at without consulting the SDMX-ML documentation, and having a thorough understanding of SDMX structure definitions in general.

Easier to use is the HTML view of data structure definitions provided by the free tool available from the SDMX Initiative. This tool is an XSLT stylesheet transformation which can either be run from the command line, or from the SDMX transformation interface described above. This tool produces a set of HTML files which are similar in kind to JavaDoc or other automatically-derived documentation - you can navigate through the dimensions and attributes to see which concepts they use, and what the legal codes are to represent them. This is very convenient for developers working with other people's key families, and for those doing reviews of key families created with the free key-family-creation tool.

Note that if you have a key family which is contained in more than one XML file, you will need to combine these files into a single structure message file before the tool for creating the HTML documentation will work. This is an activity which is best conducted by someone familiar with the editing of XML files, who can read the SDMX schemas to determine how it must be done.

D. Create XML Schemas for Specific Data Structure Definitions and Specific Use Cases

Once you have a data structure definition, it may or may not be necessary or useful to generate XML Schemas from the data structure definition. This decision is made according to which SDMX-ML data format you wish to use. If you are using the "Generic" data format, then no schema generation is necessary, as the schema will already have been provided to you in the ISO/TS 17369 SDMX version 1.0 standards package. Otherwise - for Compact, Utility, or Cross-sectional formats - you will need to generate an XML Schema which is specific to your key family. (If you are using SDMX-EDI, there is only one EDI format provided, so this is not an issue - no generation is necessary).

If you do not understand the difference between the various SDMX-ML formats, then you might want to read the SDMX-ML document in the ISO/TS 17369 SDMX version 1.0 standards package, which explains how each of these formats is designed to support different exchange scenarios and use cases. To summarize this briefly:

- Generic data format: this is a data format which can contain data which is structured according to any key-family - it is the same schema for all data sets, which allows for generic tools to be created which are not designed to handle a specific kind of data. It is a fairly verbose format, and does not do any innate validation on the data, to make sure that it agrees with the key family. Generic format supports partial data sets and incremental updating.
- Compact data format: this format, as the name implies, is designed to produce a less-verbose XML message - that is, smaller file size. It does perform some validation of the data against the key family, but very often those using this data format will want to have access to the data structure definition file at run time when processing the data files. Compact format supports incremental updates and partial data sets.
- Utility data format: this format is designed to perform maximum validation of SDMX data against the key family. Thus, when the data file is created, all required documentation must be present. Incremental updates are not supported. Access to the data structure definition (the "key family") is not needed at run time, as this information is present in the generated XML schema itself. This is a verbose format.
- Cross-sectional data format: this format allows for non-time series data to be described in SDMX-ML. It does some validation - similar to the Compact format - and it is also quite compact.

Note that within the parameters described above, all of these data formats are equal, and transformations between them are possible. This is discussed in more detail below.

There are two ways to generate schemas from data structure definitions: manually, by reading and following the rules in the specification; or, alternately, by using the free tools from the SDMX Initiative. The former method requires a close reading and full understanding of a very technical part of the specification, and is not recommended. The latter technique is simple and relatively easy, and is therefore recommended.

E. Create SDMX-ML Data - Simple Case

While most SDMX-ML data will be produced by statistical systems supported by databases, there are some scenarios (here termed the "simple case") where a developer will wish to create samples of data for test purposes, or a reporter without any sort of SDMX-ML-capable system will wish to create small amounts of data to meet a reporting requirement.

In these scenarios, there are a number of possible alternatives:

(1) Use a generic XML authoring package such as XMetal, XML Spy, or similar. You will need to know which type of SDMX-ML is wanted, and also to have a copy of the structure description (key family) to be used. Most generic tools will work best with a Utility SDMX-ML format, so the freeware tool for creating this schema from the key family could be employed. The same approach could be used for the Compact or Cross-Sectional SDMX-ML schemas, as these have similar tools. If the Generic format is desired, no schema need be created, since the only one needed is available from SDMX.

Note that the data can be authored in any of these formats, and that tools exist for translation into any of the other SDMX-ML formats. Thus, there are many possible ways of producing the needed format.

(2) Use the SDMX free-ware Microsoft Excel tool for creating the data. This is a simple add-on to Microsoft Excel which allows you to import the structure definition (key family) in its SDMX-ML format, and then author your SDMX-ML data directly as a spreadsheet. The add-on allows you to export the SDMX-ML data in any time-series-oriented SDMX-ML format (that is, as anything but Cross-sectional). Note that the spreadsheet presentation of data in this tool is more oriented toward XML creation than it is a traditional spreadsheet layout for statistical data.

(3) A third option is to write VBA scripts to export the needed SDMX-ML directly from an existing spreadsheet layout. This requires a better knowledge of SDMX-

ML than other approaches, but may be the most useful to those actually authoring the data, as it may be familiar to them.

F. View SDMX-ML Data

For those wishing to view SDMX-ML data obtained from elsewhere in its XML form, there are also a number of alternatives. Note that this type of viewing would be relevant for those who are working with SDMX-ML as developers, or to inspect the contents of a data set slated for publication. The actual publication of SDMX-ML data in a more groomed presentation is discussed below.

(1) Use a Text Editor or Browser: If the audience is good with XML in its raw form, the use of any text editor or popular browser is possible. No particular presentation is made of the SDMX-ML, but for many developers this is not an issue.

(2) Use the SDMX Freeware Tool: This tool will take SDMX-ML Generic data, and create a presentation of it in HTML. This requires that the SDMX-ML key family be provided to the tool, and it may require that the SDMX-ML data be transformed into the generic format (using the appropriate SDMX freeware tool) before viewing. The presentation, while not of a quality suitable for publication on the web or in print, should be sufficient for those not comfortable with viewing XML in its raw form. the tool - like most of the SDMX-ML transformation tools, is an XSLT stylesheet.

G. Report, Collect, and/or Exchange Data between Systems

The topics of data reporting, data collection, and data exchange are really aspects of the same basic task: any exchange requires at least two counterparties, one or more of which is providing data to another. This may be termed "reporting," it may be termed "collection", or it may be termed "data exchange". We will address it as a single task for the purposes of this document.

There are many possible ways to exchange data in SDMX-ML and/or SDMX-EDI, that being the focus of the ISO/TS 17369 SDMX Technical Standards in version 1.0. We will try to characterize this activity in simple terms.

One primary distinction is whether the data is being sent by one counterparty to another (termed a "push" scenario) or whether the data is posted in an accessible location, and then obtained when needed (termed a "pull" scenario). The mechanisms behind these two can be very different when using SDMX-ML.

Typically, data reporters/collectors exchange data on a regular basis, and work together to develop a successful automated data exchange. We assume that the

data exchange discussed here is not one where a previously unfamiliar source of SDMX-ML has been discovered (we consider that data publication, not reporting/collection - it is discussed under another task, below).

One major consideration in data exchange scenarios is the size of the data files, and whether they are a factor. Many exchange systems do not employ data compression technologies, and the result is the exchange of very large files. XML tends to be verbose, and there are several ways to address this requirement, even without using compression technologies. SDMX-ML is just like any other XML in this regard, of course - because it is a text-based format, it tends to compress very well if such technologies are used.

The first decision that must be made in an exchange scenario is to agree on the key family to be used. Other than what is provided in the tasks above, there is not much we can say here to help with key family selection - we assume that a key family exists or has been created, and that it is available to all counterparties in an SDMX-ML format (or, alternately, as SDMX-EDI). The first decision, given this assumption, is which format for data should be used. These are characterized below:

(1) SDMX-EDI

EDIFACT syntax is a very terse syntax, and SDMX-EDI may provide some of the smallest file sizes for exchange. It is possible to send incremental updates in SDMX-EDI, and it is possible to send the data and documentation (that is, the attributes) as separate messages. Further, SDMX freeware tools are being created for translating between SDMX-EDI and SDMX-ML. The down-side of using an EDIFACT syntax is that there are not many commercially available tools designed to work with this format, and the technology expertise for working in EDIFACT generally is less common than for working with XML.

(2) SDMX-ML Generic

The SDMX-ML Generic format is extremely verbose, and should not be used for the exchange of large data sets. It is designed for exchange of data across domain boundaries, because systems which deal with many different key families need only understand the one, Generic schema which is used with all data, regardless of the key family. The Generic format can transmit whole or partial data sets, and can be used for incremental updates. Data and documentation can be exchanged separately.

(3) SDMX-ML Compact

This format is the XML equivalent of SDMX-EDI. It is the least verbose time-series-oriented SDMX-ML format, but is still considerably more verbose than SDMX-EDI. It works with XML tools, of course, which may be enough of an advantage that the larger file sizes are acceptable. Compared to other SDMX-ML formats, however, it is less verbose.

(4) SDMX-ML Utility

This is also a very verbose format, and is intended for use in processing and dissemination scenarios rather than exchange scenarios. It provides very good validation capabilities, leveraging a generic XML parser to enforce many of the business rules expressed in the key family. It is not particularly suited for data exchange, however, except with very small data sets.

(5) SDMX-ML Cross-Sectional

The Cross-sectional format is the only SDMX format which is designed for use with non-time-series data. It is equivalent to the SDMX-ML Compact format in its functionality, but uses a cross-sectional presentation.

For "push" exchange scenarios, the SDMX-EDI or SDMX-ML structural definition (key family) and the corresponding data format are the only ones the user needs to consider. Counterparties need to agree on the exchange only within certain, loose parameters: you need to know what format the data sent to you will be in, and according to what key family, but the SDMX freeware tools offer you a simple way to transform the data from one format into another (albeit not a very fast one, given that they are mostly XSLT stylesheets). It should be pointed out that the SDMX-ML transformations use the SDMX-ML Generic format as a "hub" format - everything can be transformed into and out of SDMX-ML Generic format. The only case where this is not true is the transform which takes SDMX-EDI and transforms it into SDMX-ML.

For "pull" exchange scenarios, there is an additional message which has to be considered - the SDMX-ML Query. Pull scenarios involve a notification or trigger of some type - this can be the data-providing counterparty notifying the data collector that data is now available, or it can be a request sent by the data collector to trigger the creation/publication by the data provider. Web-services-based systems are always of the latter type.

Let's look first at a data notification scenario. This is a "pull" scenario where a notification message - often expressed in RSS XML - tells a data collector what

data is available, and where it can be found (that is, it provides a URL). The only limitation of this type of mechanism is that it requires some understanding of what the notification message refers to. Is it a complete data set? Is it an incremental update? What range of data is included? Typically, these questions can be answered at design-time by counterparties, so the notification mechanism is still very useful in data exchange scenarios.

For trigger scenarios, no such limitation exists. This is where the SDMX-ML Query message becomes useful. A data collector will send an SDMX-ML Query to a known URL, which then returns the data which has been requested. The Query is couched in terms of the key family, so specific data keys can be requested, as well as data over a specific time range, etc. There is still some implicit agreement in this scenario: the format of the returned data should be agreed by the counterparties, and whether incremental updates or partial data sets will be returned. It is possible with the Query message to specify a maximum suggested file size, and to have the returned file indicate that it has been truncated. The application behavior associated with these fields should also be agreed by counterparties at design-time.

The "trigger" scenario could be implemented using web services technology, and there are guidelines provided in the ISO/TS 17369 SDMX Technical Standards Version 1.0 package. There is no requirement to use web-services technology to implement this type of system, however - it could be done using HTTP POST or other familiar Internet protocols.

Ultimately, agreements should be made explicit between counterparties in any data reporting/collection scenario - ISO/TS 17369 SDMX Technical Standards version 1.0 provide many useful tools for supporting these scenarios, but do not dictate to users how all of the tools are used. This must rely on coordinated implementation and testing between counterparties.

H. Publish/Disseminate Data

Dissemination or publication of data is characterized here as distribution of data to those with whom there are not regular counterparty agreements in place. Thus, researchers using data off the web would be a typical audience for disseminated/published data.

When data is being published and disseminated, there are many formats which may be useful as output to consumers. One of these, of course, is a well-designed HTML presentation of the data. Other formats for dissemination include CSV- for use in Excel and Databases - and XML formats. Print may also be a publication "format"; but we will not address that here other than to suggest that XSL-FO is an excellent standard tool for doing finely-tuned presentations for

print, and that there are good open-source tools for generating PDF from XML and performing similar production tasks (Cocoon is the tool of choice for many users).

When disseminating data in an HTML format, XSLT transformations should be considered as a possible tool. Most browsers today will actually take an XML file with an embedded pointer to an XSLT stylesheet, and perform the transformation in the client browser. More reliable is server-side transformation, which can be done at publication time or at the time of request using a wide variety of XSLT tools. Many useful tools are open-source or free.

When looking at these types of transformations, the correct SDMX-ML format should be selected. Because much of the structural definition is captured in the SDMX Utility format, and because it is the most "typical" XML format of all those in SDMX-ML, it may be a good selection for small amounts of data. Other formats may be useful as well - see the discussion under Reporting, Collection, and Exchange of Data, above.

While there is an SDMX freeware tool for creating views of Generic SDMX-ML, that tool is not designed to produce views which are of suitable quality for a website.

There are also SDMX freeware tools for creating CSV (with the time dimension oriented vertically or horizontally), and these should be considered as one way of producing this output for dissemination. The SDMX-ML itself, in whatever format is most suitable, is another output which data consumers find increasingly useful.

Note that whenever SDMX-ML data is published, it is customary - and strongly encouraged - to include in the data links to the SDMX-ML file which contains a complete key family description. While it is possible to distribute this across several files - and there are sometimes good reasons for doing this - in a dissemination scenario, most users will find having all of the concepts, dimensions, attributes, and code-lists declared in a single file to be the easiest way to access the structural information.

Further - because of the variety of formats provided by ISO/TS 17369 SDMX Technical Standards version 1.0 - it is also customary to provide the schema which validates the XML on the site where the data is available. (Many XML tools like to have the schema available, so this is customary with many XML formats, not only SDMX-ML.)

Also, in a dissemination scenario it is not generally a good idea to publish incremental updates or partial data sets. You may wish to divide large data sets into several files, each of which addresses some known set of keys, or covers a known time range.

I. Use Disseminated/Published Data

When SDMX-ML data is provided on a website or elsewhere by a disseminator/publisher, the user needs to know several things about it: what is the key family used to structure the data? How are the concepts defined? What is the SDMX-ML format used? Is the data an incremental update? Does it include attributes? What is the XML Schema? Also, SDMX Applications should explain much of their functioning in a Conformance Statement - these are described below.

Remember that SDMX Generic format always uses the same XML schema, regardless of the key family/structural definition. It may be a good idea to create applications which use the SDMX-ML Generic format, so that there is no dependency on specific key families when processing data. Especially for applications whose primary job is the display of data, a generalized application can be created to work on all SDMX-ML Generic data. If your data source provides some other SDMX-ML data format, then the SDMX freeware tool can be used to put it into SDMX-ML Generic format before processing.

J. Knowing What an SDMX-Compliant Application is Capable of (Conformance Statements)

ISO/TS 17369 SDMX Technical Specification version 1.0 requires that any application which supports SDMX-ML or SDMX-EDI must post a plain-language description of how it supports the SDMX Technical Specifications. In the Framework document of the standards package, the contents of this statement are explained. Basically, each of the various options regarding the use of SDMX must be spelled out, so that users of that application will know whether to expect interoperability between that application and other SDMX-conformant applications they might be using.

The conformance statement will explain things like which SDMX-ML or SDMX-EDI format is supported, whether the capability is read, write, or read-write, whether there are specific requirements that the key family/structural definitions are in a single file, etc.

Users of SDMX data should both look for conformance statements on the applications that they interact with, and think about creating conformance statements for any applications they develop. Note that the term "application" here is used in its broadest sense - any website which disseminates SDMX-ML should have a conformance statement, as well as the kinds of software packages which we typically think of as "applications".

IV. Other Topics

A. Notes on Application Development

Those implementing SDMX-ML systems should be aware that most of the SDMX freeware tools work both through a Windows interface and also can be invoked from the command line. This design is intentional - while some users will only occasionally perform a transformation, and are content to do this by hand, other users will wish to integrate transformations with other processing on a server, or within another application. Most development packages provide a facility for invoking XSLT transformations on XML data, and for this reason, almost all of the SDMX freeware transformations are created as XSLT stylesheets.

There are some tradeoffs here - XSLT, even when compiled - is very memory intensive, and this tends to get worse with large file sizes. Developers will wish to think about whether they are processing XML as a DOM tree or in a streaming fashion, as the latter technique is much more performant.

B. Languages

Those working with SDMX-ML should be aware that all XML applications are required to support the UTF-8 form of Unicode. While other character encodings may be useful, UTF-8 is assumed, as it provides support for many different character sets.

SDMX-ML is designed to support multiple languages. Within the structural definition (key family), most possible values are provided as codes. Each code can be given multiple labels, each in a different language. Because the data carries the codes, applications can use the SDMX-ML key family file to perform automatic translations on data displays.

Any fields within SDMX-ML which are language-specific allow for parallel language versions to be supplied. Thus, if an attribute contains plain text, it is possible to provide that text in several languages. While there is nothing in the SDMX-ML schemas which requires the provision of multiple language versions, this is a possibility, and can be agreed with counterparties who are supplying data.

Languages within SDMX-ML are identified using the normal XML mechanism - that is, with the xml:lang attribute carrying language and locale codes.

C. Looking Ahead to Version 2.0

Version 1.0 of ISO/TS 17369 SDMX Technical Specifications focused entirely on data formats and related issues. Version 2.0 will provide a revised (although backward-compatible) version of the format standards, but will also provide standards which support far more than data formats.

The following list gives a general outline of what will be contained in version 2.0:

(1) Reference Metadata Structures and Reporting

This capability allows for "pure" metadata reporting - that is, metadata can be reported independent of a specific set of data. There is capability to integrate both data and metadata formats, but the standard will support data reporting, metadata reporting, or both.

(2) Hierarchical Code-lists and Cubes

It will be possible in the version 2.0 standards to have hierarchies of values represented by code-lists, rather than just flat lists. Further, more of the cube relationships will have standard formats for exchange between those using OLAP and similar systems.

(3) SDMX Registry Services

Specifications will be provided for the creation of and interaction with SDMX registries. These registries will help support subscription/notification, querying and navigation of data and metadata, and other registry functions. While based on a web-services registry model, the SDMX registry is specifically tuned to the SDMX Information Model.

(4) Data Quality and Validation

A number of features addressing data quality and validation issues will be addressed in SDMX Version 2.0. These are handled with the reference metadata reporting functions in some cases, and in others are more closely related to

issues around data integrity checking and functional dependencies within cube structures.