SDMX-ML:
SCHEMA AND DOCUMENTATION

(VERSION 2.0)

November 2005

Contents

88

1 BACKGROUND

1.1 History and Version 2.0 Developments

The SDMX Technical Standards Version 1.0 established an information model which described aggregated statistical data sets and the structural metadata needed to exchange them in a standard fashion. This drew on the earlier example of the GESMES/TS standard. Based on the SDMX information model, several formats were developed: XML formats for exchange of structural metadata, data sets, and queries for these (SDMX-ML), and EDIFACT formats for the structural metadata and data sets (SDMX-EDI). These standards supported a number of exchange patterns, characterized as "bilateral", "gateway", and "data-sharing" models, as described in the Framework document in the Version 1.0 standards package.

Version 2.0 builds on this foundation to provide a higher degree of support for all of these models, with an emphasis on data sharing in the form of a set of standard registry services interfaces. It has also expanded to include support for new types of metadata exchange and reporting, with a focus on "reference metadata" concerned with quality, methodology, and other issues. Further, the ability to provide metadata about the relationships between data sets and structures has been expanded, providing more support for data cubes. Finally, experience has shown that some minor additions to the existing structural metadata and dataset structures will allow SDMX to support more different types of statistical information.

The scope of the Version 2.0 SDMX Technical Standards is thus much broader, and is accompanied by a larger set of message types in the SDMX-ML formats. While the XML formats described here have grown in number and scope, the EDI formats remain relatively unchanged.

1.2 The XML Design

All of these document types will share a common "envelope" at the message level ("`SDMXMessage.xsd`"), as well as a set of common low-level components ("`SDMXCommon.xsd`") so that header information and basic structure will always be the same.

- Schema for describing all types of structural metadata – for data sets (key families), for metadata sets (metadata structure definitions), for related groups of metadata and data structures, and for all types of structural objects involved in registry-based exchanges ("`SDMXStructure.xsd`")
- Generic data schema for data-sharing exchange ("`SDMXGenericData.xsd`")
- Generic query schema for invoking web services ("`SDMXQuery.xsd`")
- Key-family-specific schema for updates and revisions/bilateral exchange ("`SDMXCompactData.xsd`")
- Key-family-specific schema for presentational processing and internal use ("`SDMXUtilityData.xsd`")

- Key-family-specific schema for cross-sectional data ("SDMXCrossSectionalData.xsd")
- Generic schema for registry interfaces ("SDMXRegistry.xsd")
- Generic schema for reference metadata sets ("SDMXRefMetadata.xsd")
- Metadata-structure-definition-specific schema for metadata sets ("SDMXMetadataReport.xsd")

1.3 Fostering the Use of a Standard SDMX-ML

In addition to these different formats, standard mappings and corresponding transformation tools have been developed for the creation of key-family-specific schemas from structure descriptions, to transform XML data instances from one XML data description format to another, and from these formats into the corresponding SDMX-ML messages. This level of free tools support will foster the early use of SDMX and permit the data to be easily used across all processes, which is otherwise a difficult requirement to meet. Ultimately, it is the fact that all formats share a common information model that enables this approach to meet the wide set of SDMX requirements.

2 NORMATIVE REFERENCES

W3C XML Schema Definition Language, version 1.0 (URL: http://www.w3c.org/XML/Schema#dev), World Wide Web Consortium
W3C Extensible Markup Language, version 1.0, Third Edition (URL: http://www.w3c.org/TR/2004/REC-xml-20040204/), World Wide Web Consortium

3 CONFORMANCE

Sections V and VI of this document are normative, providing rules for the creation of conformant SDMX-ML XML instances and W3C XML Schemas.

4 DESIGN OVERVIEW

4.1 Scope and Requirements

To understand the relationships between the several document types, it is important to have some familiarity with the requirements they are designed to fulfil.

- Large amounts of data must be captured in a reasonably compact format, because of the potential size of databases being exchanged.
- It must be possible to send incremental updates, rather than entire, complete databases. The validation of such exchanges demands not that an entire data set be exchanged, but only that enough information be sent to ensure accurate updating and revision processes.

- Structural information as well as data will need to be transmitted.
- There must be a reliable transformation to and from the GESMES/TS EDIFACT syntax.
- It should be possible to present natural-language information in multiple, equivalent languages.
- To support web services and similar technological approaches, there is a requirement to send queries to information sources as well as data and structures.
- Users (and registry services) may not know about a specific key family, and will need to be able to handle data across key families, and even (for, say, a comparison service) to put data structured according to multiple key families in a single XML instance.
- The XML must be as simple as possible (but no simpler) to allow use by web-masters and developers who are not familiar with statistics as a domain.
- The XML should behave as "normally" as possible within standard XML tools such as web development environments, parsers, guided editing tools, etc.
- Validation of data sets should provide validation that the data set is complete – the validation profile for incremental updates is not sufficient. Because the XML schema for the data set must of necessity allow partial data sets when used for the purposes of updating, it cannot provide validation of the complete data set. The need exists, however, for the validation of both complete data sets and partial data sets used for updates.
- Data should be structured not only as time series data, but potentially also as cross-sectional data, to meet the demands of different users. It must be possible to take data structured according to a single key family and transform it into a standard format enabling either of these structural optimizations.
- XML formats should promote re-use of common semantics, concepts, and codelists to the greatest possible extent, while still recognizing the agency which maintains a specific resource (a codelist, a key family, a data set, etc.)
- XML formats must support interactions of applications with standard registry services, based on standard interfaces. These must function both as web services, and as services operating over http and similar protocols.
- XML formats must support the reporting of reference metadata which is not structural in nature, but which constitutes a primary information flow of metadata attached to other parts of the statistical collection, reporting, processing, exchange, and dissemination. Quality initiatives, methodological metadata, administrative metadata, and similar types of metadata reporting must be supported, and must be user-configurable.

- XML formats for describing the relationships between groups of metadata sets and data sets, by mapping concepts and codelists between these structures, and by allowing for common querying of data and metadata described with not only a single structural definition, but with a related set of structural definitions, based on these mappings.

- Allow for time-related concepts which are not related to the time of the observation to be used in data structures.

- Allow for simple, un-coded incremental identifiers in data structure definitions, to be used to dis-ambiguate data series/observations which do not have a simple 1-to-1 relationship with the time period of the observation.

- Allow for un-coded identifiers and descriptors to be associated with data structure definitions which establish an external entity or identifier to disambiguate between otherwise identical series/observations (ie, when a data set describes a group of organisations, or a set of accounts, which might otherwise have identical key values).

- Allow for non-numeric observation values (usually but not always coded)

- Allow "cube"-based systems (such as OLAP) to interoperate with less sophisticated systems, without necessarily losing the richness of metadata found in the more sophisticated systems.

This is a very broad set of requirements, and in examining these it becomes evident that some of the requirements are very much at cross-purposes. It is almost impossible to design a single XML document type for any single function (exchange of data, exchange of reference metadata, querying, etc.) which will satisfy all of these requirements. At the same time, it was very much felt that whatever design was adopted should have a clear relationship with the information model.

4.2 Design Approach

One of the most powerful aspects of the GESMES/TS implementation guide is its data model, which allows the EDIFACT message to be used for many different types of data. The XML design built on this approach by extending the use of the model to span not only types of statistical data – expressed as key families – but also syntaxes. A key family is a metadata construct – it can be expressed in many syntaxes, but relies on none. In looking at the idea of using the SDMX Information Model (a superset of the GESMES/TS data model) to span syntaxes, it became apparent that a similar approach could be used to span use-case-specific XML formats. Because they would all be based on the same model, their equivalence would be guaranteed. With a simple transformation, anyone's data or metadata, expressed in EDIFACT or a process-specific XML, could be transformed into the flavour preferred by the receiver of the data. Further, from a processable description of a key family or metadata structure (the XML description), it would be possible to generate format descriptions, tools, and configurations specific to that key family or metadata structure.

The main argument against this approach is its apparent complexity, which is a negative factor when launching international standards. In looking at requirements,

moreover, it was realized that not only were key-family-specific XML formats needed, but also formats which could accommodate more than one key family or metadata structure without changing – that is, to be non-key-family-specific/non-metadata structure-specific.

The result of this analysis was the idea of a compromise position. It was immediately agreed that there could be only one XML format for describing a key family or metadata structure – more than one is unnecessary. A requirement existed for services which could use data and/or metadata structured according to any key family, and sometimes in combination. This presented the need for a "generic" data format and a "generic" metadata format. The querying requirement insisted that a Query message be created (which had, at one time, been discussed within the GESMES/TS community, although never finalized.) Additionally, it was seen that there were other scenarios which had significantly conflicting requirements in terms of XML design:

- Database exchange, update, and revision
- "Normal" XML use and processing for webmasters, developers, and other users of typical XML tools
- Exchange of cross-sectional data (which could potentially be the same as the Database Exchange scenario)
- Standard interactions with registry services

To support the broad set of requirements, it was felt that a small number of standard document types should be articulated, to meet specific processing requirements. This included the scenarios described above, and the use of the query document type, which would only be needed for those developing web services or similar applications involving run-time creation of SDMX-ML data from databases.

The idea of reuse has not been lost in this design, however – wherever possible, common structures have been reused. This has resulted in a common "message" structure, in which there is a single header shared by all document types, and a single "envelope" (not to be confused with a web-services SOAP envelope, which contains entire SDMX-ML messages of any type). Additionally, the core structure of any key-family-specific XML document type should be common with that of any other, to the greatest extent reasonably possible. A shared set of XML constructs was also developed, to be used throughout all the XML formats, to increase consistency.

The end result is a primary division between "generic" XML formats, which are not specific to particular key families, and a set of formats which are specific to key families or metadata structure definitions, and to particular scenarios for use.

Such design decisions as whether something is to be expressed as an XML element or attribute have been made based on the specific requirements for each XML format. For those formats where compactness of data is paramount, almost everything is expressed as attributes, because this results in a more compact expression of the data. In other cases – in UtilityData messages, for example – other types of structures are used which are more verbose, but which capture more of the

metadata expressed in the key family (eg, ordering of the key). This type of difference in design stems always from the requirements for the specific XML format being designed.

4.3 SDMX-ML Packaging: Namespace Modules

In the proposed XML Schema design, there is a packaging scheme based on the idea that XML namespaces can be used as "modules", so that any given user or application need only be familiar with a subset of the entire library in order to use it. This approach fit very well with the design described above, and is often used in major XML standards for other domains.

The other major benefit of namespaces – especially in light of the requirement that maintenance agencies be tracked across the potential reuse of the structures and data they maintained – is that it allows SDMX to own certain namespace modules, and allows other maintenance agencies to own namespaces specific to the key-families or metadata structure definitions they also maintain.

The result is a set of namespace packages which agree with the design approach described above. Each module is a single instance of the W3C XML Schema Language's schema element, associated with its own XML namespace. Where these modules have dependencies on one another, they use the XML Schema importing mechanism to draw on constructs described in another module.

- An SDMX Namespace Module containing the common message constructs, including the common header information ("`SDMXMessage.xsd`") - used with all other SDMX-ML namespace modules

- An SDMX Namespace Module containing the descriptions of structural metadata such as key families, concepts, and codelists ("`SDMXStructure.xsd`")

- An SDMX Namespace Module containing constructs shared in common across all of the SDMX message types ("`SDMXCommon.xsd`") – needed for all other SDMX-ML namespace modules (also included for convenience is the XML namespace ["`xml.xsd`"] provided by the W3C for including the xml:lang attribute in schemas).

- An SDMX Namespace Module describing the generic (non-key-family-specific) format for formatting data ("`SDMXGenericData.xsd`")

- An SDMX Namespace Module for describing the structure of the generic query message ("`SDMXQuery.xsd`") – for web services developers and users, etc.

- An SDMX Namespace Module providing the common framework to be used for all key-family-specific schemas for Database Exchange, Update, and Revisions ("`SDMXCompactData.xsd`") – for bilateral use

- A set of namespaced modules created and maintained by those who create key-family-specific "Compact" schemas – not maintained by SDMX

- An SDMX Namespace Module providing the common framework to be used for all key-family-specific schemas for webmasters and developers using standard XML tools ("`SDMXUtilityData.xsd`") –for processing and publication production use

- A set of namespaced modules created and maintained by those who create key-family-specific "Utility" schemas – not maintained by SDMX

- An SDMX Namespace Module providing the common framework to be used for all key-family-specific schemas for cross-sectional data ("`SDMXCrossSectionalData.xsd`") – for bilateral use and cross-sectional processing of data

- A set of namespaced modules created and maintained by those who create key-family-specific "Cross-sectional" schemas – not maintained by SDMX

- An SDMX Namespace Module providing a generic format for reporting of reference metadata, regardless of metadata structure definition ("`SDMXRefMetadata.xsd`").

- An SDMX Namespace Module providing the common framework to be used for all metadata-structure-specific schemas for reference metadata reporting ("`SDMXMetadataReport.xsd`").

- A set of namespaced modules created and maintained by those who create metadata-structure-specific "Metadata Report" schemas – not maintained by SDMX.

- An SDMX Namespace Module providing standard interfaces for interactions with a set of registry services ("`SDMXRegistry.xsd`").

The following sections describe in detail the proposed XML formats, which should be examined alongside the documentation provided. These proposed schemas are divided into the generic schemas, for which a complete set of schema definitions can be provided, and key-family-specific schemas, for which a core structure is provided (with schema code), plus a guide to how a specific key-family or metadata structure definition can be mapped onto the core structure.

When namespaces are created by the creators and maintainers of the key-family-specifiic and metadata-structure-definition-specific types described above, the namespaces must be generated according to a specific format which is predictable. This is identical to the construction of registry URNs, as explained in section 5.2 of the SDMX Registry Interfaces specification, with the addition of a single field at the end of the URN:

- For Utility schemas: ":utility"
- For Compact schemas: ":compact"
- For Cross-Sectional schemas: ":cross"
- For Metadata Report schemas: ":metadatareport"

5 GENERIC (NON-STRUCTURE-DEFINITION-SPECIFIC) SCHEMAS

Some SDMX-ML schemas are the same for all key families and metadata structure definitions. These include:

- `SDMXMessage.xsd`, for generically describing the basic message structure common to all SDMX-ML messages
- `SDMXStructure.xsd`, for describing key families, metadata structure definitions, dataflows, metadataflows, codelists, concepts, structure sets, processes, hierarchical codelists, and reporting taxonomies
- `SDMXGenericData.xsd`, for describing data across key-families for generic processing
- `SDMXQuery.xsd`, for marking-up queries against SDMX-conformat databases and web services
- `SDMXCommon.xsd`, describing the common constructs used in other schemas
- `SDMXGenericMetadata.xsd`, for generically reporting reference metadata
- `SDMXRegistry.xsd`, for all interactions with the SDMX Registry Services

Of these, only the `SDMXStructure` message and the `SDMXGenericData` message are required for general exchange of data. For generic exchange of reference metadata, only the `SDMXStructure` message and the `SDMXGenericMetadata` message are required. The documentation for each of these schemas is provided below. (The schemas themselves are appended separately.)

5.1 SDMX Message Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/structure (SDMXStructure.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/generic (SDMXGenericData.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/utility (SDMXUtilityData.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/compact (SDMXCompactData.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross (SDMXCrossSectionalData.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/query (SDMXQuery.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/registry (SDMXRegistry.xsd)

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/genericmetadata (SDMXGenericMetadata.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/metadatareport (SDMXMetadataReport.xsd)

5.1.1 Global Elements

Structure(StructureType): The Structure is a message that contains all the structural metadata about a data set. This can be key families, concepts, or codelists.

GenericData(GenericDataType): The GenericDataType is used to convey data in a cross-key-family form.

UtilityData(UtilityDataType): The UtilityData contains data in an XML form which is specific to each key family, according to standard mappings, and which is optimized to support guided editing tools and other applications which expect a "typical" XML schema. This format can be used to validate data in a key-family-specific fashion as is typically expected of XML schemas, and requires the entire data set. It cannot be used for incremental updates.

CompactData(CompactDataType): CompactData contains data in an XML format which is optimized for incremental updating, and the transfer of large data sets bilaterally. It is specific to each key family, according to standard mappings. It allows for key values to be expressed at a Group level.

CrossSectionalData(CrossSectionalDataType): CrossSectionalData contains data in an XML format which is optimized for describing many observations at a single point in time, and for the transfer of large data sets bilaterally. It is specific to each key family, according to standard mappings. It allows for key values to be expressed from the Group level down to the Observation level, and permits multiple observation values with different "measures".

GenericMetadata(GenericMetadataType): GenericMetadata contains reported metadata in an XML format which supports any metadata structure definition.

MetadataReport(MetadataReportType): MetadataReport contains a metadata report which is specific to a particular metadata structure definition. This format allows for the validation of the constraints described in the metadata structure definition with a generic XML parser.

RegistryInterface(RegistryInterfaceType): The RegistryInterfaceMessage is used to conduct all interactions with the SDMX Registry Services.

QueryMessage(QueryMessageType): The QueryMessageType is used to query databases published on the web, and to invoke web services. It allows for queries to be made regarding both data and structural metadata.

MessageGroup(MessageGroupType): The MessageGroupType is used to allow for more than one data or metadata message of a single type to be included in a single transmission. This element arises from the requirement for some services to be able to exchange data or metadata which may come from more than one source, and be structured according to more than one key family or metadata structure definition.

Header(HeaderType): Header type is declared globally so that it can function as the head of a substitution group for schemas which are used internally. While this is an exception to the overall design of SDMX-ML, many users feel this construct is useful. Note that when SDMX-ML messages are exchanged outside an organization, the standard header should be used - no assumptions about additional fields in substituted types should be made unless explicitly agreed-to by counterparties.

5.1.2 Complex Types

MessageType: The Message is an abstract type which is used by all of the messages, to allow inheritance of common features. It also provides uniqueness constraints for the header fields.

Element Content (Type):

Header (HeaderType)

StructureType: StructureType defines the contents of a structure message.

Extends: MessageType

Element Content (Type):

OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
Dataflows (structure:DataflowsType) - min. 0
Metadataflows (structure:MetadataflowsType) - min. 0
CategorySchemes (structure:CategorySchemesType) - min. 0
CodeLists (structure:CodeListsType) - min. 0
HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
Concepts (structure:ConceptsType) - min. 0
MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) - min. 0
KeyFamilies (structure:KeyFamiliesType) - min. 0
StructureSets (structure:StructureSetsType) - min. 0

ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0
Processes (structure:ProcessesType) - min. 0

GenericDataType: GenericDataType defines the contents of a GenericData message.

Extends: MessageType

Element Content (Type):

DataSet (generic:DataSetType)

UtilityDataType: UtilityDataType defines the contents of a UtilityData message.

Extends: MessageType

Element Content (Type):

[Reference] (utility:DataSet)

CompactDataType: CompactDataType defines the contents of a CompactData message.

Extends: MessageType

Element Content (Type):

[Reference] (compact:DataSet)

CrossSectionalDataType: CrossSectionalDataType defines the contents of a CrossSectionalData message.

Extends: MessageType

Element Content (Type):

[Reference] (cross:DataSet)

GenericMetadataType: GenericMetadataType defines the contents of a Generic Metadata message.

Extends: MessageType

Element Content (Type):

[Reference] (genericmetadata:MetadataSet)

MetadataReportType: MetadataReportType defines the contents of a metadata structure definition-specific Metadata Report message.

Extends: MessageType

Element Content (Type):

[Reference] (metadatareport:MetadataSet)

QueryMessageType: QueryMessageType defines the contents of a QueryMessage.

Extends: MessageType

Element Content (Type):

Query (query:QueryType)

RegistryInterfaceType: This is a type which describes a structure for holding all of the various dedicated registry interface message types.

Extends: MessageType

MessageGroupType: MessageGroupType defines the contents of a MessageGroup message.

Extends: MessageType

Attribute: id(xs:NMTOKEN) - optional

HeaderType: HeaderType defines the header fields used for all messages. ID identifies a data flow definition, which, when combined with time, uniquely identifies the data set. Test indicates whether the message is for test purposes or not. Truncated is used in data messages which are responding to Query messages, and is set to true only if the response has been truncated to meet size limits suggested by the defaultLimit attribute in the Query mesage. Name provides a name for the transmission. Prepared is the date prepared. Sender is information about the sender, and Receiver is information about the receiver. Agency provides the code identifier/abbreviation for the maintenance

agency of a data set. Data set id provides an identifier for a contained data set. Action code provides a code for determining whether the enclosed message is an Update or Delete message (not to be used with the UtilityData message). KeyFamilyRef is used to reference a key family for a contained data set, using its id. (This information is required at the DataSet level for some messages, but is provided here as a convenience for those messages which do not require it.) KeyFamilyAgency specifies the agency of the key family using its coded id. Fields which refer to a contained data set need not be used if the message contains a query or structural information - these messages provide specific fields for holding this information. The ones here are not to be used as defaults. Extracted is a time-stamp from the system rendering the data; ReportingBegin and ReportingEnd provide the time period covered by the message (in the case of data). Source provides human-readable information about the source of the data.

Element Content (Type):

ID (common:IDType)
Test (xs:boolean)
Truncated (xs:boolean) - min. 0
Name (common:TextType) - min. 0 - max. unbounded
Prepared (HeaderTimeType)
Sender (PartyType) - max. unbounded
Receiver (PartyType) - min. 0 - max. unbounded
KeyFamilyRef (xs:NMTOKEN) - min. 0
KeyFamilyAgency (xs:NMTOKEN) - min. 0
DataSetAgency (xs:NMTOKEN) - min. 0
DataSetID (xs:NMTOKEN) - min. 0
DataSetAction (common:ActionType) - min. 0
Extracted (xs:dateTime) - min. 0
ReportingBegin (HeaderTimeType) - min. 0
ReportingEnd (HeaderTimeType) - min. 0
Source (common:TextType) - min. 0 - max. unbounded

PartyType: PartyType defines the information which is sent about various parties such as senders and receivers of messages. The Name is the ID of the party, and Contact provides contact details.

Element Content (Type):

Name (common:TextType) - min. 0 - max. unbounded
Contact (ContactType) - min. 0 - max. unbounded

Attribute: id (xs:NMTOKEN) - required

ContactType: ContactType provides defines the contact information about a party. The Name provides a human-readable name.

Element Content (Type):

Name (common:TextType) - min. 0 - max. unbounded
Department (common:TextType) - min. 0 - max. unbounded
Role (common:TextType) - min. 0 - max. unbounded
Telephone (xs:string) *[Choice]*
Fax (xs:string) *[Choice]*
X400 (xs:string) *[Choice]*
URI (xs:anyURI) *[Choice]*
Email (xs:string) *[Choice]*

5.1.3 Simple Types

HeaderTimeType: Provides a union type of xs:date and xs:dateTime for the header fields in the message.

5.2 SDMX Structure Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/structure

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

5.2.1 Complex Types

OrganisationSchemesType: OrganisationSchemesType contains one or more OrganisationSchemes.

Element Content (Type):

OrganisationScheme (OrganisationSchemeType) - max. unbounded

OrganisationSchemeType: OrganisationSchemeType contains the details of an OrganisationScheme. In OrganisationSchemes, the organisation roles of agency, data provider, and data consumer may be specified. A single organisation may play more than one role. Name is an element which provides for a human-readable name for the organization. Description may be used to provide a longer, human-readable description. the is attribute provides a formal ID for the organisation scheme; the version attribute specifies a particular version. If blank, it is assumed that the version is "1.0". The uri attributre specifies the location of a valid SDMX Structure Message containing the full details of the organisation sc`heme, and is required if the isExternalReference attribute has a value of true. If isExternalReference has a value of false, full details must be provided in the current instance of the

OrganisationScheme element. The urn attribute provides a formal SDMX Registry URL - see the Logical Registry Specification for specific requirements. An agencyID must be provided, identifying the maintenance agency of the organisation scheme. Also, if the organisation scheme is final, the isFinal attribute must have a value of true - otherwise, it will be assumed to be non-final. (All production schemes must be made final - that is, unchangeable without versioning.) The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Agencies (AgenciesType) - min. 0 - max. unbounded
DataProviders (DataProvidersType) - min. 0 - max. unbounded
DataConsumers (DataConsumersType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: agencyID (common:IDType) - required

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

DataProvidersType: DataProvidersType contains one or more data providers. Data providers are those who report or disseminate data sets or metadata sets.

Element Content (Type):

DataProvider (OrganisationType) - max. unbounded

DataConsumersType: DataConsumersType contains one or more data consumers. Data consumers collect or use disseminated data sets and metadata sets.

Element Content (Type):

DataConsumer (OrganisationType) - max. unbounded

AgenciesType: AgenciesType contains one or more Agencies. Agencies are those organisations which act as the maintainers of structural definitions of various types. Agencies are often supplied as part of an organisation scheme, but may also be supplied independently using this element.

Element Content (Type):

Agency (OrganisationType) - max. unbounded

OrganisationType: OrganisationType provides a structure for describing agencies, data providers, and data consumers and their contact information. The id attribute carries a code identifying the agency. The version attribute indicates the version of the agency description. The uri attribute provides a uri for an alternate way of identifying the agency information (typically a URL resolving to an agency described in SDMX-ML). Name is an element which provides for a human-readable name for the organization. Description provides for a longer human-readable description of the organisation, which may be provided in multiple, parallel language-equivalent forms. MaintenanceContact provides contact information for the agency when acting as a MaintenanceAgency; CollectorContact does the same when the agency is acting as a statistics collector; DisseminatorContact for when the agency functions as a statistics disseminator; and ReporterContact for when the Agency is functioning as a statistics reporter. OtherContact is used to describe any other role. Note that the Role field in the contact information structure should only be specified for OtherContact. It is allowable to reference full Agency information by using (at a minimum) only the id, name, and uri fields, with the uri pointing to an external description in a valid SDMX-ML Structure message which provides more complete information. (This is termed an "external reference".) If an external reference is being made, the isExternalReference attribute must be set to "true". The urn attribute holds a valid SDMX Registry URN (see SDMX Registry Specification). The parentOrganisation attribute holds the id of a parent organisation of the same type from the same scheme, indicating that the organisation in question is a department or other sub-division of the parent organisation. Annotations may be provided using the Annotations element, in multiple, parallel-language form.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
MaintenanceContact (ContactType) - min. 0
CollectorContact (ContactType) - min. 0

DisseminatorContact (ContactType) - min. 0
ReporterContact (ContactType) - min. 0
OtherContact (ContactType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: parentOrganisation (common:IDType) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

ContactType: ContactType provides defines the contact information about a party. The id element is used to carry user id information for the contact, whereas Name provides a human-readable name.

Element Content (Type):

Name (common:TextType) - min. 0 - max. unbounded
id (common:IDType) - min. 0
Department (common:TextType) - min. 0 - max. unbounded
Role (common:TextType) - min. 0 - max. unbounded
Telephone (xs:string) *[Choice]*
Fax (xs:string) *[Choice]*
X400 (xs:string) *[Choice]*
URI (xs:anyURI) *[Choice]*
Email (xs:string) *[Choice]*

DataflowsType: DataflowsType contains one or more data flows.

Element Content (Type):

Dataflow (DataflowType) - max. unbounded

DataflowType: DataflowType describes the structure of a data flow. A human-readable name must be provided, and may be given in several language-specific variations. A longer human-readable description (also in multiple language-specific versions) may be provided. A reference must be made to a key family, and to a category within a category scheme, using the KeyFamilyRef and CategoryRef elements, unless the Dataflow is a reference

to an external data flow, in which case a url must be provided in the uri attribute, and the isExternalReference attribute must be set to true.. Annotations may be provided in the Annotations element. An id unique to the maintaining agency (identified in the agencyID attribute) must be supplied in the "id" attribute; a version may be specified, and is assumed to be "1.0" if not supplied. The urn attribute may contain a valid registry URN (as per the SDMX Registry Specification). If the dataflow is final, the isFinal attribute must have a value of true - any production dataflow must be final (that is, it cannot be changed without versioning). The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
KeyFamilyRef (KeyFamilyRefType) - min. 0
CategoryRef (CategoryRefType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: agencyID (common:IDType) - required

Attribute: isFinal (xs:boolean) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

KeyFamilyRefType: KeyFamilyRefType provides a reference to a key-family (data set structure definition). At a minimum, either (a) The key family ID must be provided, as assigned to the key family by the agency whose ID is the value of KeyFamilyAgencyID. A version must also be provided; OR (b) a valid SDMX Registry URN must be provided in the URN element (see SDMX Registry Specification)

Element Content (Type):

URN (xs:anyURI) - min. 0
KeyFamilyID (common:IDType) - min. 0
KeyFamilyAgencyID (common:IDType) - min. 0
Version (xs:string) - min. 0

CategoryRefType: CategoryRefType provides a reference to a category. At a minimum, either a value for CategorySchemeAgencyID, CategorySchemeID, and CategoryID must be provided, or a valid SDMX Registry URN must be provided in the URN element (see SDMX Registry Specification).

Element Content (Type):

URN (xs:anyURI) - min. 0
CategorySchemeID (common:IDType) - min. 0
CategorySchemeAgencyID (common:IDType) - min. 0
CategorySchemeVersion (xs:string) - min. 0
CategoryID (CategoryIDType) - min. 0

CategoryIDType: CategoryIDType describes a structure which can provide a path inside a hierarchical category scheme. Each node (category) of the referenced scheme is represented by a CategoryID element, with sub-categories represented by the child CategoryID element. Each CategoryID element must be given a node identifier in the ID field, which corresponds to the ID of the category. It is not necessary to represent the full category path with the nesting structure if each node within the hierarchical category scheme has a unique id.

Element Content (Type):

ID (common:IDType)
CategoryVersion (xs:string) - min. 0
CategoryID (CategoryIDType) - min. 0

MetadataflowsType: MetadataflowsType contains one or more metadata flows.

Element Content (Type):

Metadataflow (MetadataflowType) - max. unbounded

MetadataflowType: MetadataflowType describes the structure of a metadata flow. A human-readable name must be provided, and may be given in several language-specific variations. A longer human-readable description (also in multiple language-specific versions) may be provided. A reference must be made to a metadata structure definition, and to a category within a category scheme, using the MetadataStructureRef and CategoryRef elements. If the Metadataflow is an external reference, this is indicated by setting the

isExternalReference attribute to true, and providing a url where the full description can be found in the form of a valid SDMX-ML structure message. In this case, only the id and name must be provided. Annotations may be provided in the Annotations element. An id unique to the maintaining agency (identified in the agencyID attribute) must be supplied in the "id" attribute; a version may be specified, and is assumed to be "1.0" if not supplied. The urn attribute may contain a valid registry URN (as per the SDMX Registry Specification). If the metadata flow is final, the isFinal attribute must have a value of true - any production metadata flow must be final (that is, it cannot be changed without versioning). The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
MetadataStructureRef (MetadataStructureRefType) - min. 0
CategoryRef (CategoryRefType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: agencyID (common:IDType) - required

Attribute: isFinal (xs:boolean) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

MetadataStructureRefType: MetadataStructureRefType provides a reference to a metadata structure definition. The ID must be provided, as assigned to the metadata structure definition by the agency whose ID is the value of MetadataStructureAgencyID. A version must also be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
MetadataStructureID (common:IDType) - min. 0

MetadataStructureAgencyID (common:IDType) - min. 0
Version (xs:string) - min. 0

CategorySchemesType: CategorySchemesType contains one or more category schemes.

Element Content (Type):

CategoryScheme (CategorySchemeType) - max. unbounded

CategorySchemeType: CategorySchemeType describes the structure of a category scheme. This is a simple, levelled hierarchy. The scheme itself is given a human-readable name (which may be in multiple language-specific versions), and may optionally have a human-readable description (also in multiple, landuage-specific versions). Annotations may be provided in the Annotations element. The Category element represents a set of nested categories which describe a simple classification hierarchy. The CategoryScheme must have an agency specified in teh agency attribute, and a unique ID provided for all of the category schemes of that agency in the id attribute. A version may also be supplied - if ommitted, the version is understood to be "1.0". If the isFinal attribute has a value of true, the category scheme is final and cannot be changed without versioning. All production category schemes must be final. The urn attribute holds a valid registry URN (see the SDMX Registry Specification). If the isExternalReference attribute has a value of true, then the uri attribute must have a value which provides the location of a valid SDMX Structure message providing full details of the Category Scheme. Otherwise, all details must be provided here. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Category (CategoryType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

930 *Attribute:* isFinal (xs:boolean) - optional

931 *Attribute:* validFrom (common:TimePeriodType) - optional

932 *Attribute:* validTo (common:TimePeriodType) - optional

933 **CategoryType:** The category is given a human-readable name (which may
934 be in multiple language-specific versions), and may optionally have a human-
935 readable description (also in multiple, landuage-specific versions).
936 Annotations may be provided in the Annotations element. References to
937 dataflows and metadataflows may be provided. The Category element
938 represents a set of nested categories which are child categories. The
939 Category must have a unique ID within the Category Scheme provided in the
940 id attribute. A version may also be supplied - if ommitted, the version is
941 understood to be "1.0". The urn attribute holds a valid registry URN (see the
942 SDMX Registry Specification).

943 *Element Content (Type):*

944
945 Name (common:TextType) - max. unbounded
946 Description (common:TextType) - min. 0 - max. unbounded
947 DataflowRef (DataflowRefType) - min. 0 - max. unbounded
948 MetadataflowRef (MetadataflowRefType) - min. 0 - max. unbounded
949 Category (CategoryType) - min. 0 - max. unbounded
950 Annotations (common:AnnotationsType) - min. 0

951 *Attribute:* id (common:IDType) - required

952 *Attribute:* version (xs:string) - optional

953 *Attribute:* urn (xs:anyURI) - optional

954 *Attribute:* uri (xs:anyURI) - optional

955 *Attribute:* isExternalReference (xs:boolean) - optional

956 **CodeListsType:** CodelistsType contains one or more codelists. It also
957 defines uniqueness constraints for codelist IDs.

958 *Element Content (Type):*

959
960 CodeList (CodeListType) - min. 0 - max. unbounded

961 **CodeListType:** CodeListType defines the contents of a codelist. This includes
962 an ID, the agency which maintains the codelist, its version, and a URL where
963 it is located. Elements are provided for supplying a name and the codes. It is
964 acceptable to provide only the id, name, and uri fields at a minimum, with the
965 uri pointing to an SDMX Structure message containing complete details on the

codelist. (This is termed an "external reference".) If an external reference is being made, the isExternalReference attribute must be set to "true". The urn attribute holds a valiud SDMX Registry URN (see SDMX Registry Specification). The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Code (CodeType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

CodeType: CodeType defines the structure of a code. This allows for plain-text descriptions as element content, and the coded value as the value attribute. (Short descriptions or other presentational information may be added using Annotations with an indicative type field [eg, "ShortDescription"]). The urn attribute supplies a valid SDMX Registry URN (see the SDMX Registry Specification).The parentCode attribute provides the ability to describe simple hierarchies within a single codelist, by referenceing the id value of another code in the same codelist.

Element Content (Type):

Description (common:TextType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: value (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

Attribute: parentCode (common:IDType) - optional

HierarchicalCodelistsType: HierarchicalCodelistsType contains one or more sets of structural information about the hierarchies within a codelist (hierarchical codelists). This corresponds to complex hierarchical codelists within the SDMX Information Model - very simple hierarchies can be described within the regular Codelist, using the parentCode attribute.

Element Content (Type):

HierarchicalCodelist (HierarchicalCodelistType) - max. unbounded

HierarchicalCodelistType: A hierarchical codelist references a Codelist, and supplies the extra structural metadata to assemble the codes into a hierarchy. A human-readable name must be supplied, and multiple language-specific variants may be provided. A longer human-readable description may be provided, and may also be presented as a set of language-specific variants. The CodelistRef element references a codelist, and may indicate more than one. Annotations may be provided in the Annotaions element. An ID unique for the agency specified in the agency attribute must be assigned, using the id attribute. A version may be provided using the version attribute - if no value is provided, it is assumed to be "1.0". A valid SDMX Registry URN may be provided in the urn attribute, as specified in the SDMX Registry Specification. If the isExternalReference attribute has a value of true, the uri attribute must specify the location of a valid SDMX Structure Message which provides the full details of the hierarchical codelist; otherwise, all details must be present. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
CodelistRef (CodelistRefType) - min. 0 - max. unbounded
Hierarchy (HierarchyType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

HierarchyType: The recursive CodeRef element is used to assemble the codes in the codelist(s) referenced by the parent hierarchical codelist into a hierarchy. The Level element is used to describe the levels of a levelled hierarchy, which may be referenced from each of the CodeRefs in the Hierarchy. A human-readable name must be assigned, which may be provided in multiple, parallel-language versions. A longer, human-readable Description may also be provided, which can also have multiple parallel-language versions. Annotations may be provided with the Annotations element. The id attribute provides a unique id for the hierarchy. The urn attribute can be used to specify the hierarchy with a valid SDMX Registry URN (see the SDMX Registry Specification). The version attribute specifies a version (understood to be "1.0" if not specified), and isFinal, once given a value of true, indicates that nothing may be changed without also changing the version number. validFrom and validTo are inclusive dates indicating the relevant period of the hierarchy.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
CodeRef (CodeRefType) - min. 0 - max. unbounded
Level (LevelType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

Attribute: version (xs:string) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

LevelType: LevelType describes a level in a hierarchical codelist. The Order element specifies where the level is in a levelled hierarchy, starting with the

value "1" for the top level, and going sequentially from there using whole integers. CodingType specifies the text formatting of the codes at that level. A human-readable name must be assigned, which may be provided in multiple, parallel-language versions. A longer, human-readable Description may also be provided, which can also have multiple parallel-language versions. Annotations may be provided with the Annotations element. The id attribute provides a unique id for the hierarchy. The urn attribute can be used to specify the hierarchy with a valid SDMX Registry URN (see the SDMX Registry Specification).

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Order (xs:integer)
CodingType (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

CodelistRefType: The CodelistRefType provides the structure for a codelist reference. (Note that this is structured differently than a similarly-named type in the Registry namespace.) At a minimum, either: AgencyID has the ID of an agency as a value; CodelistID takes the ID of a codelist maintained by that agency; and Version specifies the version of the codelist; or URN supplies a valid SDMX Registry URN (see the SDMX Registry Specification). Alias is used to carry the identifier for the referenced codelist, so that codes from that list can be easily referenced by the CodeRefs contained in the parent Hierarchy, without having to repeat the agency and version for each reference. The Alias must be unique within the parent Hierarchical Codelist.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
CodelistID (common:IDType) - min. 0
Version (xs:string) - min. 0
Alias (common:IDType) - min. 0

CodeRefType: The CodeRefType provides the structure for a codelist reference. At a minimum, either a URN value (a valid SDMX Registry URN as specified in teh SDMX Registry Specification) must be supplied, or a ColdelistAliasRef and a CodeID must be specified. CodelistAliasRef references an alias assigned in a CodelistRef element in the containing hierarchical codelist.CodeRef references a code from the codelist identified at the level of the parent hierarchical codelist. Codes are arranged in a hierarchy

by reference. Note that it is possible to reference a single code such that it has multiple parents within the hierarchy. Further, the hierarchy may or may not be a levelled one. CodeID holds the ID of the code in the codelist referenced by the hierarchical codelist. CodeRef references a code. LevelRef holds the id of a Level described in the same parent Hierarchical Codelist. NodeAliasID allows for an ID to be assigned to the use of the particular code at that specific point in the hierarchy. This value is unique within the hierarchy being created, and is used to map the hierarchy against external structures. Version holds the version number of the referenced code, to support management of complex hierarchies. Along with this field are the ValidFrom and ValidTo dates, which are inclusive dates during which the code is valid within the parent hierarchy.

Element Content (Type):

URN (xs:anyURI) - min. 0
CodelistAliasRef (common:IDType) - min. 0
CodeID (common:IDType) - min. 0
CodeRef (CodeRefType) - min. 0 - max. unbounded
LevelRef (common:IDType) - min. 0
NodeAliasID (xs:string) - min. 0
Version (xs:string) - min. 0
ValidFrom (common:TimePeriodType) - min. 0
ValidTo (common:TimePeriodType) - min. 0

ConceptsType: The ConceptsType describes an XML type which contains information about sets of concepts and their relationships, each of which is described in a ConceptScheme element. This section replaces the section of the version 1.0 SDMXStructure message which provides details about concepts. As such, it is backward-compatible, and may be used to contain a simple list of concepts as per the 1.0 SDMX-ML specification.

Element Content (Type):

Concept (ConceptType) - min. 0 - max. unbounded
ConceptScheme (ConceptSchemeType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

ConceptType: ConceptType specifies the information provided for a single concept. This includes a name, as element content, and an ID. It is possible to use the uri field to point to the location of an SDMX-ML Structure message which contains a more detailed version of the concept. (This is termed an "external reference".) If an external reference is being made, the isExternalReference attribute must be set to "true". In this case, all details of the concept are assumed to be found externally, and inline characteristics provided through child elements and the coreRepresentation and coreRepresentationAgency attributes are to be ignored. The coreRepresentation and coreRepresentationAgency attributes can identify a codelist which is a default representation of the concept. Uncoded default

representations (or information about the textual aspects of coded default representations) can be provided with the TextFormat child element of the concept. Semantic relationships between concepts which occur within a single concept scheme can be captured with the parent and parentAgency attributes - these identify the concept of which the current concept is a qualification (in the ISO 11179 sense) or subclass. When used outside of a containing ConceptScheme, these attributes may be ignored. If a coreRepresentation and core RepresentationAgency are not provided, but are provided in the indicated parent, then the default representation is inherited from the specified parent concept. Note that all concepts within a concept scheme must be uniquely identified by their id - each concept scheme has only one agency for all its concepts. The agency attribute here is provided for backward-compatibility with version 1.0 of the standards, and it must not be used for concepts which are child elements of a concept scheme.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
TextFormat (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - optional

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: coreRepresentation (common:IDType) - optional

Attribute: coreRepresentationAgency (common:IDType) - optional

Attribute: parent (common:IDType) - optional

Attribute: parentAgency (common:IDType) - optional

ConceptSchemeType: ConceptSchemeType describes the structure of a ConceptScheme element, which is the preferred form (as of version 2.0) of presenting the concepts used in other SDMX constructs. ConceptSchemes may be included inline (that is, with all details provided in the instance or may be referenced externally. It is possible to use the uri field to point to the

location of an SDMX-ML Structure message which contains a more detailed version of the concept. (This is termed an "external reference".) If an external reference is being made, the isExternalReference attribute must be set to "true". A Name may be provided as a child element (in multiple parallel language versions) and an ID and version and agency information may be provided. ConceptSchemes represent a collection of concepts which are used to describe a meaningful set of distinct concepts, to be used in the reporting of data or metadata. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Concept (ConceptType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

MetadataStructureDefinitionsType: MetadataStructureDefinitionsType describes one or more metadata structure definitions.

Element Content (Type):

MetadataStructureDefinition (MetadataStructureDefinitionType) - max. unbounded

MetadataStructureDefinitionType: A metadata structure definition performs several functions: it groups sets of objects into "targets" against which reference metadata may be reported. Targets define the structure of the reference metadata "keys" which identify specific types of reported metadata,

and describe the valid values for populating the keys. Also, metadata structure definitions provide a presentational organization of concepts for reporting purposes. The structure of a reference metadata report is derived from this presentational structure. Also, representations - unless defaults from the concepts are used - must be indicated as part of this presentational structure. Attributes allow the assignment of an ID, an agency, a version, and a uri. It is possible to use the uri field to point to the location of an SDMX-ML Structure message which contains a more detailed version of the metadata structure definition. (This is termed an "external reference".) If an external reference is being made, the isExternalReference attribute must be set to "true". When an external reference is being made, none of the child elements should be included. Otherwise, both TargetIdentifiers and at least one ReportStructure must be included. The urn attribute holds a valid SDMX registry URN (see the SDMX Registry Specification). The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
TargetIdentifiers (TargetIdentifiersType) - min. 0
ReportStructure (ReportStructureType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

TargetIdentifiersType: TargetIdentifiers are the set of objects against which reference metadata is reported (data providers, data flows, data or metadata structures, etc.). There are two types of TargetIdentifiers: the "full" target identifier, which lists every object used to attach metadata to in the metadata structure definition, and the partial target identifiers, which indicate sub-sets of

those concepts against which reference metadata may be reported. It is sometimes the case that metadata will also be reported against the full target identifier. An example of this is as follows: we might wish to collect some metadata concepts such as contact information for some of the objects described by the SDMX Information Model - for each instance of a metadata flow or a data provider, for example. Our concepts would be the concepts associated with contact information: CONTACT_NAME, CONTACT_PHONE_NUMBER, etc. We would determine how these concepts are associated with the objects in the model: do we want a contact for each data provider broken out by data flow? Or do we want only a single contact reported for each data provider (who might provide several data flows)? Each object or combination of objects we need to have metadata reported for becomes a partial target identifier, unless it happens to contain the full target identifier, in which case we use that instead. Thus, our valid partial target identifiers here would be "data flow" and "data provider", because "data flow by data provider" is a full target identifier. For each target identifier, we could have some set of our concepts reported.

Element Content (Type):

FullTargetIdentifier (FullTargetIdentifierType)
PartialTargetIdentifier (PartialTargetIdentifierType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

FullTargetIdentifierType: The full target identifier provides details on all of the objects against which metadata can be reported. The full target identifier is made up of a set of identifier components - each getting its own child element - which are similar to the dimensions of a key family: each one indicates that a value will be provided by the metadata reporter to identify and describe the metadata being reported. A human-readable name must be provided, which may be provided in multiple, parallel-language versions. A longer, human-readable name may also be provided in multiple, language-parallel versions. Annotations may be provided.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
IdentifierComponent (IdentifierComponentType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

IdentifierComponentType: An identifier component describes the use of an object within the full target identifier set. An identifier component must be one of the formally-recognized object classes found in the SDMX Information Model: the sub-element TargetObjectClass provides a way of indicating which objects will be used in reporting metadata, and will indicate how those objects are to be identified by the metadata reporters (which value sets will be allowed for each identification field for each object). The RepresentationScheme child element is used to indicate the valid range of values for the providing taget identifiers in reported metadata.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
TargetObjectClass (ObjectIDType)
RepresentationScheme (RepresentationSchemeType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

PartialTargetIdentifierType: Partial target identifiers are subsets of the full target identifier. They are themselves given an identifier, so that they can be referenced by the metadata attributes of a report. A human-readable name must be provided, which may be provided in multiple, parallel-language versions. A longer, human-readable name may also be provided in multiple, language-parallel versions. Annotations may be provided.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
IdentifierComponentRef (common:IDType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

RepresentationSchemeType: Representation schemes indicated which values are valid for identifying objects within each class. For any given representation scheme, two IDs must be provided: the RepresentationScheme must have an ID as assigned to it by it

representationSchemeAgency, whose ID must also be provided. The type of the representation scheme is expressed in the representationSchemeType attribute.

Attribute: representationScheme (common:IDType) - required

Attribute: representationSchemeAgency (common:IDType) - required

Attribute: representationSchemeType (RepresentationSchemeTypeType) - required

ReportStructureType: The report structure describes the presentation of the reported concepts, and associates them with target identifiers, full or partial. It can be given a name and/or annotations. It must be given an ID, using the id attribute, which must be unique within the MetadataStructureDefinition element. It contains one or more MetadataAttribute elements, each of which may either hold a value, or may have subordinate MetadataAttribute elements. The target attribute holds the ID of a full or partial identifier, which is the identifier of the target against which the metadata attributes are reported.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
MetadataAttribute (MetadataAttributeType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: target (common:IDType) - required

MetadataAttributeType: Metadata attributes are those concepts - whether taking a coded or uncoded value, or made up of child concepts, or both - which are reported against a full or partial target identifier. If there are nested metadata attributes, these concepts are subordinate to the parent metadata attribute - that is, for the purposes of presentation, the parent concept is made up of the child concepts. This hierarchy is strictly presentational, for the purposes of structuring reports. If the metadata attribute can have a coded or uncoded value, then the charateristics of the value are indicated with the TextFormat child element. If the value is coded, then the representationScheme and representationSchemeAgency attributes must hold values: the representationScheme attribute takes the ID of a representation scheme, and the representationSchemeAgency takes the ID of

the agency which maintains that scheme. The conceptRef attribute holds the ID of the metadata attribute's concept. The conceptAgency attribute takes the agency ID of the concept referenced in conceptRef. The conceptSchemeRef attribute holds the ID value of the concept scheme from which the concept is taken, and the conceptSchemeAgency holds the ID of the agency that maintains the concept scheme referenced in the conceptSchemeRef attribute. The useageStatus attribute indicates whether provision of the metadata attribute is conditional or mandatory.

Element Content (Type):

MetadataAttribute (MetadataAttributeType) - min. 0 - max. unbounded
TextFormat (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptRef (common:IDType) - required

Attribute: conceptVersion (xs:string) - optional

Attribute: conceptAgency (common:IDType) - optional

Attribute: conceptSchemeRef (common:IDType) - optional

Attribute: conceptSchemeAgency (common:IDType) - optional

Attribute: representationScheme (common:IDType) - optional

Attribute: representationSchemeAgency (common:IDType) - optional

Attribute: usageStatus (UsageStatusType) - required

TextFormatType: TextFormatType defines the information for describing a text format. If the TextType attribute is not specified, any valid characters may be included in the text field. (It corresponds to the xs:string datatype of W3C XML Schema.) The textType attribute provides a description of the data type, and may place restrictions on the values of the other attributes, referred to as "facets". The isSequence attribute indicates whether the values are intended to be ordered, and it may work in combination with the interval attribute. The minLength and maxLength attributes specify the minimum and maximum lengths of the value in characters. startValue and endValue are used for inclusive and exclusive ranges, indicating what the bounds of the range are. The interval attribute specifies the permitted interval between two values. The timeInterval attribute indicates the permitted duration between two time expressions. The decimals attribute indicates the number of characters allowed after the decimal separator. The pattern attribute holds any regular expression permitted in the simila facet in W3C XML Schema.

Attribute: textType (TextTypeType) - optional

Attribute: isSequence (xs:boolean) - optional

Attribute: minLength (xs:integer) - optional

Attribute: maxLength (xs:integer) - optional

Attribute: startValue (xs:double) - optional

Attribute: endValue (xs:double) - optional

Attribute: interval (xs:double) - optional

Attribute: timeInterval (xs:duration) - optional

Attribute: decimals (xs:integer) - optional

Attribute: pattern (xs:string) - optional

KeyFamiliesType: KeyFamiliesType defines the structure for describing one or more key families. It also provides uniqueness constraints for each of the key family IDs.

Element Content (Type):

KeyFamily (KeyFamilyType) - max. unbounded

KeyFamilyType: KeyFamilyType defines the structure of a key-family description. This includes the name and a set of components (attributes and dimensions) as element content, and an ID, agency, version, and the URL where located as attributes. The urn attribute holds a valid SDMX Registry URN, as per the SDMX Registry spec. The isFinal attribute, once set to true, indicates that no changes may be made without versioning. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version. If the isExternalReference attribute is true, then the uri attribute must be provided, giving a location where a valid structure message can be found containing the full details of the key family.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Components (ComponentsType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

ComponentsType: ComponentsType describes the dimensions, groups, attributes, and measures of the key family. If TimeDimension is included in the key family - which it must be if time series formats for the data (GenericData, CompactData, and UtilityData formats) are to be used - then there must also be a frequency dimension.

Element Content (Type):

Dimension (DimensionType) - min. 0 - max. unbounded
TimeDimension (TimeDimensionType) - min. 0
Group (GroupType) - min. 0 - max. unbounded
PrimaryMeasure (PrimaryMeasureType)
CrossSectionalMeasure (CrossSectionalMeasureType) - min. 0 - max. unbounded
Attribute (AttributeType) - min. 0 - max. unbounded

DimensionType: DimensionType describes the structure of non-Time dimensions. The order of their declaration is significant: it is used to describe the order in which they will appear in data formats for which key values are supplied in an ordered fashion (exclusive of the Time dimension, which is not represented as a member of the ordered key). Some types of non-Time dimensions have un-coded values: the TextFormat element must be provided, to indicate what type of values are permissible. The attributes isFrequencyDimension and isEntityDimension may have a "true" value for any instance of the Dimension element, indicating that it is a dimension of the stated type. The attributes isCountDimension, isNonObservationalTimeDimension, isMeasureDimension, and is IdentityDimension may occur multiple times, and take a true value to indicate that the diemsnion in question is of that type. Note that only one dimension in the key family may be of the following types: Frequency dimension and Entity dimension, and only if there is not also an attribute in the key family of the same type. Any given dimension may only have a true value for one of the set

of attributes isFrequencyDimension, isCountDimension, is measureDimension, isEntityDimension, isNonObservationalTimeDimension, and is IdentityDimension. The definitions and limits on representation of each dimension type are as follows: Frequency dimension describes the period between observations, and is coded; Count dimensions are represented by values which are sequential, incrementing numbers - representations are always of the Increment or Count type; measureType dimensions are always coded, and they enumerate the set of possible measures declared for the key family; Entity dimensions describe the subject of the data set (ie, a country) - they can be coded or represented in any other form; Non-Observational Time dimensions must have a representation which contains a time; Identity dimensions may be coded or uncoded, but must be represented by a scheme which refers to the identifiers of external entites. (Conventionally, it is the first dimension in the ordered set of dimensions - the key.) If a key family describes cross-sectional data, then for each dimension, the crossSectionalAttachDataSet, crossSectionalAttachGroup, crossSectionalAttachSection, and crossSectionalAttachObservation attributes must be given values. A value of "true" for any of these attributes indicates that the dimension may be provided a value at the indicated level within the cross-sectional structure. Note that these attributes do not need to be provided for any dimension with the isFrequencyDimension set to "true", as these dimensions are always attached only at the group level, as is time. A key family designed for cross-sectional use must be structured such that any observation's complete key can be unambiguously described by taking each dimension value from its observation level, section level, group level, and data set level, and ordered according to the sequence given in the key family. For any dimension, the id of the referenced concept must be unique acrss the entire key family, including all dimensions, attributes and measures.

Element Content (Type):

TextFormat (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptRef (common:IDType) - required

Attribute: conceptVersion (xs:string) - optional

Attribute: conceptAgency (common:IDType) - optional

Attribute: conceptSchemeRef (common:IDType) - optional

Attribute: conceptSchemeAgency (common:IDType) - optional

Attribute: codelist (common:IDType) - optional

Attribute: codelistVersion (xs:string) - optional

Attribute: codelistAgency (common:IDType) - optional

Attribute: isMeasureDimension (xs:boolean) - default: false

Attribute: isFrequencyDimension (xs:boolean) - default: false

Attribute: isEntityDimension (xs:boolean) - default: false

Attribute: isCountDimension (xs:boolean) - default: false

Attribute: isNonObservationTimeDimension (xs:boolean) - default: false

Attribute: isIdentityDimension (xs:boolean) - default: false

Attribute: crossSectionalAttachDataSet (xs:boolean) - optional

Attribute: crossSectionalAttachGroup (xs:boolean) - optional

Attribute: crossSectionalAttachSection (xs:boolean) - optional

Attribute: crossSectionalAttachObservation (xs:boolean) - optional

TimeDimensionType: TimeDimensionType describes the special Time dimension. Any key family which will be used for time-series formats (GenericData, CompactData, and UtilityData) must include the time dimension. Any key family which uses the time dimension must also declare a frequency dimension, conventionally the first dimension in the key (the set of ordered non-time dimensions). A TextFormat element may be included for indicating the representation of time. The concept attribute must contain the concept name of the time concept. The codelist attribute may provide the value of the concept name of a codelist if needed. If a key family describes cross-sectional data, then for each dimension, the crossSectionalAttachDataSet, crossSectionalAttachGroup, crossSectionalAttachSection, and crossSectionalAttachObservation attributes must be given values. A value of "true" for any of these attributes indicates that the dimension may be provided a value at the indicated level within the cross-sectional structure. Note that these attributes do not need to be provided for any dimension with the isFrequencyDimension set to "true", as these dimensions are always attached only at the group level, as is time. A key family designed for cross-sectional use must be structured such that any observation's complete key can be unambiguously described by taking each dimension value from its observation level, section level, group level, and data set level, and ordered according to the sequence given in the key family.

Element Content (Type):

TextFormat (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptRef (common:IDType) - required

Attribute: conceptVersion (xs:string) - optional

Attribute: conceptAgency (common:IDType) - optional

Attribute: conceptSchemeRef (common:IDType) - optional

Attribute: conceptSchemeAgency (common:IDType) - optional

Attribute: codelist (common:IDType) - optional

Attribute: codelistVersion (xs:string) - optional

Attribute: codelistAgency (common:IDType) - optional

Attribute: crossSectionalAttachDataSet (xs:boolean) - optional

Attribute: crossSectionalAttachGroup (xs:boolean) - optional

Attribute: crossSectionalAttachSection (xs:boolean) - optional

Attribute: crossSectionalAttachObservation (xs:boolean) - optional

GroupType: GroupType declares any useful groupings of data, based on a selection of the declared (non-Time) dimensions (indicated with the DimensionRef element) which form partial keys to which attributes may be attached. The value of the DimensionRef element is the concept of the dimension - that is, the value of the dimension's concept attribute. Thus, if data is to be presented as a set of time series which vary only according to their differing frequencies, a "sibling group" would be declared, with all dimensions except the frequency dimension in it. If data is commonly grouped as a set of all countries, then a "Country Group" could be declared, with all dimensions except the country dimension forming part of the partial key. Any dimension which is not part of a group has a value which varies at the series level (for time series formats). There is no requirement to have only a single dimension ommitted from a partial key - it can be any subset of the set of ordered dimensions (that is, all dimensions except the time dimension, which may never be declared as belonging to a group partial key). All groups declared in the key family must be unique - that is, you may not have duplicate partial keys. All groups must also be given unique names (id attributes). Although it is conventional to declare dimensions in the same order as they are declared in the ordered key, there is no requirement to do so

- the ordering of the values of the key are taken from the order in which the dimensions are declared. The Description element provides a human-readable description (potentially in multiple, parallel languages) of the group. Note that for cross-sectional formats, the named group mechanism is not used, but is instead replaced by a generic group which carries time and frequency values with it, and allows for any available group-level attributes to be specified if desired.

Element Content (Type):

DimensionRef (common:IDType) *[Choice]* - max. unbounded
AttachmentConstraintRef (common:IDType) *[Choice]*
Description (common:TextType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

AttachmentConstraintRefType: AttachmentConstraintRefType describes a reference to an attachment constraint. This includes a reference to a dataflow, metadataflow, data provider, or provision agreement plus the ID of the attachment constraint, as assigned within the constraints associated with the referenced object, in the ConstraintRef element.

Element Content (Type):

DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
DataProviderRef (DataProviderRefType) *[Choice]*
ProvisionAgreementRef (ProvisionAgreementRefType) *[Choice]*
ConstraintRef (common:IDType)

ProvisionAgreementRefType: ProvisionAgreementRef allows for the identification of a provision agreement. At a minimum, either the URN element - holding a valid registry URN - or the set of OragnisationSchemeAgencyID, OrganisationSchemeID, DataProviderID, DataflowAgencyID, and DataflowID must be specified. Constraint can be used to express constraints associated with the provision agreement. This type differs from the similar type in the Registry namespace package by not providing information about the datasource or constraints.

Element Content (Type):

URN (xs:anyURI) - min. 0
OrganisationSchemeAgencyID (common:IDType) - min. 0
OrganisationSchemeID (common:IDType) - min. 0
DataProviderID (common:IDType) - min. 0
DataProviderVersion (xs:string) - min. 0
DataflowAgencyID (common:IDType) - min. 0
DataflowID (common:IDType) - min. 0

DataflowVersion (xs:string) - min. 0
Constraint (common:ConstraintType) - min. 0

DataProviderRefType: The DataProviderRef type structures a reference to a data provider. This requires that IDs be provided for an organisation scheme, its maintenance agency, and the data provider as identified in the referenced organisation scheme. The Version element may be used to specify the version of the indicated data provider. If absent, the most recent version is assumed. The URN element is used to provide the registry-specific urn as an alternate means of identification. At a minimum, either the URN element or OrgansisationSchemeID, OrganisationSchemeAgencyID, DataProviderID, and (optionally) Version must be supplied. When used in a response document of any type, the URN must always be provided. Constraints can be used to specify constraints associated with the data provider. This type differs from the similar type in the Registry namespace by not describing the datasource.

Element Content (Type):

URN (xs:anyURI) - min. 0
OrganisationSchemeAgencyID (common:IDType)
OrganisationSchemeID (common:IDType)
DataProviderID (common:IDType)
Version (xs:string) - min. 0
Constraint (common:ConstraintType) - min. 0

AttributeType: AttributeType describes the structure of attributes declared in the key family. If the codelist attribute is not used, then the attribute is uncoded. You may use the TextFormat element to specify constraints on the value of the uncoded attribute. The concept attribute contains the name of a concept. The codelist attribute supplies the id value of a codelist. The attachmentLevel attribute indicates the level to which the attribute is attached in time-series formats (GenericData, CompactData, and UtilityData formats). The assignmentStatus attribute indicates whether a value must be provided for the attribute when sending documentation along with the data. The AttachmentGroup element is included only when the attribute is attached at the Group level, to indicate which declared group or groups the attribute may be attached to. For each such group, an AttachmentGroup element should appear, with the content of the element being the name of the group. The AttachmentMeasure element is similar, indicating for cross-sectional formats which declared measure or measures the attribute attached at the observation level may be attached to. The isTimeFormat attribute indicates that the attribute represents the concept of time format (typically a mandatory series-level attribute with a codelist representation taken from ISO 8601). For key families not used to structure cross-sectional formats, this element may be ommitted. Each such element contains the name of the declared measure. The attributes crossSectionalAttachDataSet, crossSectionalAttachGroup, crossSectionalAttachSection, and crossSectionalAttachObservation indicate

what the attachment level or levels are for cross-sectional data formats, and may be ommitted if the key family will not be used to structure them. A value of "true" indicates that it is permissible to provide a value for the attribute at the specified level within the structure. Note that all groups in cross-sectional formats are replaced by a generic group which has any values for time and frequency, and allows any group-level attributes to be attached to it. An attribute which is an Entity attribute has a true value for the isEntityAttribute attribute - you may have either one Entity dimension or one Entity Attribute in a key family; a non-observational time has a true value for isNonObservationalTimeAttribute; and a Count attribute has a true value for the isCountAttribute attribute. The attributes isFrequencyAttribute and isEntityAttribute are mutually exclusive - that is, only one of them may have a "true" value for any instance of the Attribute element. The definitions and limits on representation of each attribute type are as follows: Frequency attribute describes the period between observations, and is coded; Count attributes are represented by values which are sequential, incrementing numbers - representations are always of the Increment or Count type; Entity attributes describe the subject of the data set - they can be coded or represented in any other form; Non-Observational Time attributes must have a representation which contains a time; Identity attributes may be coded or uncoded, but must be represented by a scheme which refers to the identifiers of external entities. Any given instance of an attribute may only have a true value for this set of attributes, and if so may not have a true value for the isTimeFormat attribute.

Element Content (Type):

TextFormat (TextFormatType) - min. 0
AttachmentGroup (common:IDType) - min. 0 - max. unbounded
AttachmentMeasure (common:IDType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptRef (common:IDType) - required

Attribute: conceptVersion (xs:string) - optional

Attribute: conceptAgency (common:IDType) - optional

Attribute: conceptSchemeRef (common:IDType) - optional

Attribute: conceptSchemeAgency (common:IDType) - optional

Attribute: codelist (common:IDType) - optional

Attribute: codelistVersion (xs:string) - optional

Attribute: codelistAgency (common:IDType) - optional

Attribute: attachmentLevel (structure:AttachmentLevelType) - required

Attribute: assignmentStatus (structure:AssignmentStatusType) - required

Attribute: isTimeFormat (xs:boolean) - default: false

Attribute: crossSectionalAttachDataSet (xs:boolean) - optional

Attribute: crossSectionalAttachGroup (xs:boolean) - optional

Attribute: crossSectionalAttachSection (xs:boolean) - optional

Attribute: crossSectionalAttachObservation (xs:boolean) - optional

Attribute: isEntityAttribute (xs:boolean) - default: false

Attribute: isNonObservationalTimeAttribute (xs:boolean) - default: false

Attribute: isCountAttribute (xs:boolean) - default: false

Attribute: isFrequencyAttribute (xs:boolean) - default: false

Attribute: isIdentityAttribute (xs:boolean) - default: false

PrimaryMeasureType: PrimaryMeasureType describes the observation values for all presentations of the data, except those cross-sectional formats which have multiple measures (for which a set of cross-sectional measures are used instead). The concept attribute points to the unique concept represented by the measure. The PrimaryMeasure is conventionally associated with the OBS-VALUE concept. The TextFormat element allows description of the contents of the observation value. The codelist attribute references a codelist if the observation value is coded. If this attribute is used, then codelistAgencyID must contain the ID of the maintenance agency of the referenced codelist. The codelistVersion attribute may be specified - if not, then the version of the referenced codelist is assumed to be "1.0".

Element Content (Type):

TextFormat (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptRef (common:IDType) - required

Attribute: conceptVersion (xs:string) - optional

Attribute: conceptAgency (common:IDType) - optional

Attribute: conceptSchemeRef (common:IDType) - optional

Attribute: conceptSchemeAgency (common:IDType) - optional

Attribute: codelist (common:IDType) - optional

Attribute: codelistVersion (xs:string) - optional

Attribute: codelistAgency (common:IDType) - optional

CrossSectionalMeasureType: CrossSectionalMeasureType describes the observation values for multiple-measure cross-sectional data formats. For non-cross sectional key families, it is not necesary to specify any cross-sectional measures.The concept attribute points to the unique concept represented by the measure. The measureDimension attribute contains the concept name of the measure dimension. The code attribute contains the value of its corresponding code in the codelist used to represent the measure dimension. A CrossSectionalMeasure must be declared for each code in the codelist used to represent the measure dimension - these will replace the primary measure for multiple-measure cross-sectional data formats.The TextFormat element allows description of the contents of the observation value. The codelist attribute references a codelist if the observation value is coded. If this attribute is used, then codelistAgencyID must contain the ID of the maintenance agency of the referenced codelist. The codelistVersion attribute may be specified - if not, then the version of the referenced codelist is assumed to be "1.0".

Element Content (Type):

TextFormat (TextFormatType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptRef (common:IDType) - required

Attribute: conceptVersion (xs:string) - optional

Attribute: conceptAgency (common:IDType) - optional

Attribute: conceptSchemeRef (common:IDType) - optional

Attribute: conceptSchemeAgency (common:IDType) - optional

Attribute: codelist (common:IDType) - optional

Attribute: codelistVersion (xs:string) - optional

Attribute: codelistAgency (common:IDType) - optional

Attribute: measureDimension (common:IDType) - required

Attribute: code (common:IDType) - required

StructureSetsType: StructureSetsType contains one or more structure sets.

Element Content (Type):

StructureSet (StructureSetType) - max. unbounded

StructureSetType: StructureSetType describes the relationships between two or more key families and/or metadata structure definitions, including the mapping between category schemes and concept schemes, to provide for the mapping of representations. This can include inheritance and extension of properties, or total or partial equivalencies. It also includes mapping of concepts existing in metadata structure definitions to those used in key families, and vice-versa. A human-readable name is provided in the Name element, which may include several language-specific variants. A longer human-readable description may also be provided, in the Description element, which may also have language-specific variants provided. The Annotations element may be used to provide annotations. The StructureRefs element references all of the key families and/or metadata structure definitions included in the Structure Set - these must be provided if a StructureMap element is used, but is not required if the structure set is only used to provide codelist mappings, concept mappings, or category mappings. The StructureMap element indicates which components in the included data and metadata structures are equivalent; CodelistMap indicates which codes map to other codelists. CategorySchemeMap indicates which categories in one scheme map to those in another scheme. ConceptSchemeMap indicates which concepts in one scheme map to those in another scheme. OrganisationSchemeMap describes how one organisation scheme maps to another. The id attribute takes an id which is unique to all structure sets maintained by the agency specified in the agency attribute. version specifies a version number (by default "1.0"). The uri attribute holds a URL where a valid SDMX Structure messgae can be found which provides full details of the StructureSet, and it must be used if the isExternalReference attribute has a value of true. The urn attribute holds a valid SDMX Registry URN as described in the SDMX Registry specification. A true value in the isFinal attribute indicates that the contents of the structure set may not be changed without versioning. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
RelatedStructures (RelatedStructuresType) - min. 0
StructureMap (StructureMapType) - min. 0
CodelistMap (CodelistMapType) - min. 0
CategorySchemeMap (CategorySchemeMapType) - min. 0
ConceptSchemeMap (ConceptSchemeMapType) - min. 0
OrganisationSchemeMap (OrganisationSchemeMapType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: agencyID (common:IDType) - optional

Attribute: version (xs:string) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: isFinal (xs:boolean) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

RelatedStructuresType: RelatedStructuresType includes references to key families (in the KeyFamilyRef element) and/or metadata structure definitions (In the MetadataStructureRef element). Any mapped CategorySchemes, ConceptSchemes, or Organisation Schemes should also be referenced. HierarchicalCodelistRef allows for HierarchicalCodelists which describe relationships between pertinent codelists to be referenced and included in the structure set - this must be used if the CodelistMap in the StructureSet refers to any hierarchical codelists.

Element Content (Type):

KeyFamilyRef (KeyFamilyRefType) - min. 0 - max. unbounded
MetadataStructureRef (MetadataStructureRefType) - min. 0 - max. unbounded
ConceptSchemeRef (ConceptSchemeRefType) - min. 0 - max. unbounded
CategorySchemeRef (CategorySchemeRefType) - min. 0 - max. unbounded
OrganisationSchemeRef (OrganisationSchemeRefType) - min. 0 - max. unbounded
HierarchicalCodelistRef (HierarchicalCodelistRefType) - min. 0 - max. unbounded

CategorySchemeRefType: CategorySchemeRef allows for references to specific category schemes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
CategorySchemeID (common:IDType) - min. 0
Version (xs:string) - min. 0

ConceptSchemeRefType: ConceptSchemeRef allows for references to specific concept schemes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
ConceptSchemeID (common:IDType) - min. 0
Version (xs:string) - min. 0

OrganisationSchemeRefType: OrganisationSchemeRef allows for references to specific organisation schemes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
OrganisationSchemeID (common:IDType) - min. 0
Version (xs:string) - min. 0

HierarchicalCodelistRefType: HierarchicalCodelistRef allows for references to specific hierarchical codelists. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
HierarchicalCodelistID (common:IDType) - min. 0
Version (xs:string) - min. 0

StructureMapType: StructureMapType describes the structure of the mapping of components between a referenced key family or metadata structure and a target key family or metadata structure. Components include any dimension, attribute, or reported concept. The Name element is used to provide a human-readable name for the component map; the Description element is used to provide a longer human-readable description. Both of these elements may be provided in multiple, language-specific variations. The StructureMapType provides for Annotations with the Annotations element. Either a KeyFamilyRef or a MetadataStructureRef must be provided; and also a TargetKeyFamilyRef or a TargetMetadataStructureRef. A series of map components are then specified using the ComponentMap element, each of which specifies the equivalence of a concept in the referenced strauctur definition to one in the target structure definition. If the isExtension attribute has a value of true, then the target structure definition inherits all properties of the referenced structure definition, and may have additional components. Note that this attribute may only be set to true if the component map has as a referenced structure definition and a target structure definition either two key families or two metadata structure definition. You cannot inherit concepts between the two type of structure definitions using this mechanism. The id attribute allows for an id to be assigned to the component map - it must be unique within its StructureSet.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
KeyFamilyRef (KeyFamilyRefType) *[Choice]*
MetadataStructureRef (MetadataStructureRefType) *[Choice]*
TargetKeyFamilyRef (KeyFamilyRefType) *[Choice]*
TargetMetadataStructureRef (MetadataStructureRefType) *[Choice]*
ComponentMap (ComponentMapType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: isExtension (xs:boolean) - optional

Attribute: id (common:IDType) - required

CodelistMapType: CodelistMap allows the description of how the codes in a codelist are represented in a target codelist or associated hierarchical codelist. A human-readable Name is provided, and a longer, human-readable description may be provided as well, in the Name and Description elements respectively. These may be supplied in multiple, language-specific versions.CodelistRef references the codelist or hierarchical codelist being mapped; TargetCodelistRef indicates the codelist to which it will be mapped. CodeMap is the element which indicates the equivalence of codes in the referenced codelist to those in the target codelist. Any codes not mapped are assumed to lack equivalence. The CodelistMap may be annotated using the

Annotations element. The id attribute is used to assign an identifier which is unique within the StructureSet for all CodelistMaps.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
CodelistRef (CodelistRefType) *[Choice]*
HierarchicalCodelistRef (HierarchicalCodelistRefType) *[Choice]*
TargetCodelistRef (CodelistRefType) *[Choice]*
TargetHierarchicalCodelistRef (HierarchicalCodelistRefType) *[Choice]*
CodeMap (CodeMapType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

CodeMapType: CodeMap describes the equivalence of the codes in the referenced codelist or hierarchical codelist indicated in the CodelistRef element of the containing CodelistMap to those in the referenced TargetCodelist in the containing CodelistMap. The CodeAlias attribute is used to assign an alias code to the equivalence for querying the structure set.

Element Content (Type):

MapCodeRef (common:IDType)
MapTargetCodeRef (common:IDType)

Attribute: CodeAlias (common:IDType) - optional

ComponentMapType: ComponentMapType describes how a component (that is, dimension, attribute, or reported concept) in a referenced metadata structure definition or key family maps to a component in a referenced "target" metadata structure definition or key family. The MapConceptRef contains the id of the concept in the metadata structure definition or key family referenced in the KeyFamilyRef or MetadataStructureRef element of the containing ComponentMap element. The MapTargetConceptRef contains the id of the concept in the metadata structure definition or key family referenced in the TargetKeyFamilyRef or TargetMetadataStructureRef element of the containing ComponentMap element. The RepresentationMapRef element contains a reference to the CodelistMap which describes how the coded representation of the referenced component maps to the coded representation of the target component. If the target component has an uncoded representation, then ToTextFormat is used to describe the un-coded representation to which the code of the referenced component should be transformed. The ToValueType element tells you whether the value, name, or description of the source value should be used in the resulting text field. The componentAlias attribute assigns a new ID to the relationship between these components. Note that of three components from different key families and/or

metadata structure definitions are all equivalent, the two component maps can share a single alias. Note also that for metadata concepts which are represented not by codelists but rather by other types of item schemes (OrganisationSchemes or CategorySchemes), these can also be referenced using the RepresentationMapRef element. The preferredLanguage attribute specifies the language to use when translating coded values into their names or descriptions, if available, in the same form as xml:lang.

Element Content (Type):

MapConceptRef (common:IDType)
MapTargetConceptRef (common:IDType)
RepresentationMapRef (RepresentationMapRefType) *[Choice]*

Attribute: componentAlias (common:IDType) - optional

Attribute: preferredLanguage (xs:language) - default: en

RepresentationMapRefType: RepresentationMapRefType describes the structure of a reference to a codelist, category scheme, or organisation scheme map. RepresentationMapAgencyID takes the id value of the maintenance agency of the codelist, category scheme, or organisation scheme map; RepresentationMapID takes the id attribute value of the codelist, category scheme, or organisation scheme map.

Element Content (Type):

RepresentationMapAgencyID (common:IDType)
RepresentationMapID (common:IDType)

Attribute: representationType (RepresentationTypeType) - default: Codelist

CategorySchemeMapType: CategorySchemeMap provides for the mapping of categories in one scheme against those in another. It requires a human-readable Name, and can have a longer human-readable Description, both of which can be supplied in multiple, parallel-language form. It may be annotated using Annotations. The id attribute carries a unique ID for CategorySchemeMaps within the StructureSet. CategorySchemeRef identifies the source CategoryScheme; TargetCategorySchemeRef identifies the target CategoryScheme.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
CategorySchemeRef (CategorySchemeRefType)

TargetCategorySchemeRef (CategorySchemeRefType)
CategoryMap (CategoryMapType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

CategoryMapType: CategoryMap allows for the mapping of a category in one scheme against a category in another, target scheme. The categoryAlias attribute allows for an alias to be assigned to the mapping for searching across the set of mapped categories. Note that the Category IDs are recursive, and can therefore describe a path through a category scheme.

Element Content (Type):

CategoryID (CategoryIDType)
TargetCategoryID (CategoryIDType)

Attribute: categoryAlias (common:IDType) - optional

ConceptSchemeMapType: ConceptSchemeMap provides for the mapping of concepts in one scheme against those in another. It requires a human-readable Name, and can have a longer human-readable Description, both of which can be supplied in multiple, parallel-language form. It may be annotated using Annotations. The id attribute carries a unique ID for ConceptSchemeMaps within the StructureSet. ConceptSchemeRef identifies the source ConceptScheme; TargetConceptSchemeRef identifies the target ConceptScheme.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
ConceptSchemeRef (ConceptSchemeRefType)
TargetConceptSchemeRef (ConceptSchemeRefType)
ConceptMap (ConceptMapType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

ConceptMapType: ConceptMap allows for the mapping of a concept in one scheme against a concept in another, target scheme. The conceptAlias attribute allows for an alias to be assigned to the mapping for searching across the set of mapped concepts.

Element Content (Type):

ConceptID (common:IDType)
TargetConceptID (common:IDType)

Attribute: conceptAlias (common:IDType) - optional

OrganisationSchemeMapType: OrganisationSchemeMap provides for the mapping of Organisations in one scheme against those in another. It requires a human-readable Name, and can have a longer human-readable Description, both of which can be supplied in multiple, parallel-language form. It may be annotated using Annotations. The id attribute carries a unique ID for OrganisationSchemeMaps within the StructureSet. OrganisationSchemeRef identifies the source OrganisationScheme; TargetOrganisationSchemeRef identifies the target OrganisationScheme.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
OrganisationSchemeRef (OrganisationSchemeRefType)
TargetOrganisationSchemeRef (OrganisationSchemeRefType)
OrganisationMap (OrganisationMapType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

OrganisationMapType: OrganisationMap allows for the mapping of an organisation in one scheme against an organisation in another, target scheme. The organisationAlias attribute allows for an alias to be assigned to the mapping for searching across the set of mapped organisations.

Element Content (Type):

OrganisationID (common:IDType)
TargetOrganisationID (common:IDType)

Attribute: organisationAlias (common:IDType) - optional

ReportingTaxonomiesType: ReportingTaxonomiesType holds on or more ReportingTaxonomy elements.

Element Content (Type):

ReportingTaxonomy (ReportingTaxonomyType) - max. unbounded

ReportingTaxonomyType: ReportingTaxonomyType groups data flows and/or metadata flows for the purposes of assembling "reports" made up of data from disparate sources. It is a maintainable object, and thus has a mandatory human-readable Name and optional Description containing a longer human-readable description. Annotations may be included. All of these fields may be provided in multiple, parallel languages. The id attribute

assignes a unique ID to the Reporting Taxonomy, version provides a version number, uri contains a URL where the SDMX-ML expression of the Reporting taxonomy can be found, and must be included if the isExternalReference attribute has a value of true. The urn attribute holds the value of a valid SDMX Registry URN as per the SDMX Registry specification. The isExternalReference attribute, if set to true, indicates that the uri attribute points to an external location for the ReportingTaxonomy, with only the id, Name element, and version supplied in addition. The agencyID attribute holds the ID of the Reporting Taxonomies' maintenance agency. Also, if the Reporting Taxonomy is final, the isFinal attribute must`have a value of true - otherwise, it will be assumed to be non-final. (All production versions must be made final - that is, unchangeable without versioning.) The sub-element Category may be used to group dataflows and metadataflows into useful sub-packages. DataflowRef and MetadataFlowRef are references to the flows which make up the reporting taxonomy at the top level. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
DataflowRef (DataflowRefType) - min. 0 - max. unbounded
MetadataflowRef (MetadataflowRefType) - min. 0 - max. unbounded
Category (CategoryType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: agencyID (common:IDType) - required

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

MetadataflowRefType: The MetadataflowRef type structures a reference to a metadataflow definition. This requires that ID are provided for a pre-existing Agency and Metadataflow Definition in the registry. The Version element may

be used to specify the version of the indicated dataflow. If absent, the most recent version is assumed. The URN element is used to provide the registry-specific URN as an alternate means of identification. When used in a response document of any type, the URN must always be provided. At a minimum, either the URN element or AgencyID, MetadataflowID, and (optionally) version must be supplied. Datasource may be used to specify a datasource. Constraint can be used to provide constraints associated with the metadataflow. Note that this is similar, but not identical to the MetadataflowRefType found in the SDMX-ML registry namespace package - it lacks references to the datasource and the constraints.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
MetadataflowID (common:IDType) - min. 0
Version (xs:string) - min. 0

DataflowRefType: The DataflowRef type structures a reference to a dataflow definition. This requires that ID are provided for a pre-existing Agency and Dataflow Definition in the registry. The Version element may be used to specify the version of the indicated dataflow. If absent, the most recent version is assumed. The URN element is used to provide the registry-specific URN as an alternate means of identification. At a minimum, either the URN element or AgencyID, DataflowID, and (optionally) version must be supplied. When used in a response document of any type, the URN must always be provided. Datasource may be used to specify a datasource. Constraints can be used to specify constraints associated with the dataflow. Note that this is similar, but not identical to the DataflowRefType found in the SDMX-ML registry namespace package - it lacks references to the datasource and the constraints.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
DataflowID (common:IDType) - min. 0
Version (xs:string) - min. 0

ProcessesType: ProcessesType describes a list of Processes.

Element Content (Type):

Process (ProcessType) - max. unbounded

ProcessType: ProcessType generically describes a statistical process. In this version of the SDMX Technical Specifications, it is not meant to support

process automation, but serves as a description of how processes occur. The primary use for this structural mechanism is the attachment of reference metadata regarding statistical processing. A process has a human-readable Name, which may be provided in multiple, parallel-language versions. It also has an optional human-readable Description, which also may be provided with multiple, parallel-language versions. The Annotations element allows for it to be annotated. The id attribute takes a unique id within the set of processes maintained by the agency identified in the agencyID attribute. The version attribute indicated the version of teh process description. The uri value is a URL where a complete description of the Process may be found; the urn attribute takes the valid registry URN of the Process, as described in the SDMX Registry Specification. If isFinal is set to true, the process description cannot be changed without versioning. If isExternalReference is true, then only the id, agency, Name, and uri (or URN) need be supplied - all other fields are assumed to be found in a valid SDMX Structure message found at the uri attribute location. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
ProcessStep (ProcessStepType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

Attribute: version (xs:string) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: isExternalReference (xs:boolean) - optional

Attribute: agencyID (common:IDType) - required

Attribute: isFinal (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

ProcessStepType: ProcessStepType describes a single step in a statistical process. ProcessSteps may be recursive. The Input element specifies the type of object(s) which serve as inputs to the process; the Output element specifies the type of objects which are the result of the process. Computation elements describe the computations involved in the process, in any form

desired by the user (these are informational rather than machine-actionable), and so may be supplied in multiple, parallel-language versions. Transitions describe the process steps to which a process is connected - that is, which processes happen next. The process step maust be given a Name, and may be given a Description. These are human-readable, and may be supplied in multiple, parallel-language versions. Annotations may be supplied. The id attribute takes the unique identifier of the process step within the parent process.

Element Content (Type):

Name (common:TextType) - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
Input (ObjectIDType) - min. 0 - max. unbounded
Output (ObjectIDType) - min. 0 - max. unbounded
Computation (common:TextType) - min. 0 - max. unbounded
Transition (TransitionType) - min. 0 - max. unbounded
ProcessStep (ProcessStepType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - required

TransitionType: TransitionType describes the Condition and next step in a transition. The Condition text is informational, and may be supplied in multiple, parallel-language form. The TargetStep holds the id of the next step in the process if the condition is met.

Element Content (Type):

TargetStep (common:IDType) - min. 0
Condition (common:TextType) - min. 0

5.2.2 Simple Types

ObjectIDType: The Object ID is used to reference a particular Object within the SDMX Information Model's formalization of statistical exchanges.

Restricts xs:NMTOKEN

Code: Agency - Agency

Code: ConceptScheme - Concept scheme

Code: Concept - Concept

Code: Codelist - Codelist

Code: Code - Code

2259 Code: KeyFamily - Key family

2260 Code: Component - Component

2261 Code: KeyDescriptor - Key descriptor

2262 Code: MeasureDescriptor - Measure descriptor

2263 Code: AttributeDescriptor - Attribute descriptor

2264 Code: GroupKeyDescriptor - Group key descriptor

2265 Code: Dimension - Dimension

2266 Code: Measure - Measure

2267 Code: Attribute - Attribute

2268 Code: CategoryScheme - Category scheme

2269 Code: ReportingTaxonomy - Reporting taxonomy

2270 Code: Category - Category

2271 Code: OrganisationScheme - Organisation scheme

2272 Code: DataProvider - Data or metadata provioder

2273 Code: MetadataStructure - Metadata structure definition

2274 Code: FullTargetIdentifier - Full target identifier

2275 Code: PartialTargetIdentifier - Partial target identifier

2276 Code: MetadataAttribute - Metadata attribute

2277 Code: DataFlow - Data flow

2278 Code: ProvisionAgreement - Data or metadata provision agreement

2279 Code: MetadataFlow - Metadata flow

2280 Code: ContentConstraint - Content constraint

2281 Code: AttachmentConstraint - Attachment constraint

2282 Code: DataSet - Data set

2283 Code: XSDataSet - Cross-sectional data set

2284 Code: MetadataSet - Metadata set

2285 Code: HierarchicalCodelist - Hierarchical codelist

2286 Code: Hierarchy - Hierarchy

Code: StructureSet - Structure set

Code: StructureMap - Structure map

Code: ComponentMap - Component map

Code: CodelistMap - Codelist map

Code: CodeMap - Code map

Code: CategorySchemeMap - Category scheme map

Code: CategoryMap - Category map

Code: OrganisationSchemeMap - Organisation scheme map

Code: OrganisationRoleMap - Organisation role map

Code: ConceptSchemeMap - Concept scheme map

Code: ConceptMap - Concept map

Code: Process - Process

Code: ProcessStep - Process step

TextTypeType: TextTypeType provides an enumerated list of the types of characters allowed in a TextFormat field.

Restricts xs:NMTOKEN

Code: String - A string datatype corresponding to W3C XML Schema's xs:string datatype.

Code: BigInteger - An integer datatype corresponding to W3C XML Schema's xs:integer datatype.

Code: Integer - An integer datatype corresponding to W3C XML Schema's xs:int datatype.

Code: Long - A numeric datatype corresponding to W3C XML Schema's xs:long datatype.

Code: Short - A numeric datatype corresponding to W3C XML Schema's xs:short datatype.

Code: Decimal - A numeric datatype corresponding to W3C XML Schema's xs:decimal datatype.

Code: Float - A numeric datatype corresponding to W3C XML Schema's xs:float datatype.

Code: Double - A numeric datatype corresponding to W3C XML Schema's xs:double datatype.

Code: Boolean - A datatype corresponding to W3C XML Schema's xs:boolean datatype.

Code: DateTime - A time datatype corresponding to W3C XML Schema's xs:dateTime datatype.

Code: Date - A time datatype corresponding to W3C XML Schema's xs:date datatype.

Code: Time - A time datatype corresponding to W3C XML Schema's xs:time datatype.

Code: Year - A time datatype corresponding to W3C XML Schema's xs:gYear datatype.

Code: Month - A time datatype corresponding to W3C XML Schema's xs:gMonth datatype.

Code: Day - A time datatype corresponding to W3C XML Schema's xs:gDay datatype.

Code: MonthDay - A time datatype corresponding to W3C XML Schema's xs:gMonthDay datatype.

Code: YearMonth - A time datatype corresponding to W3C XML Schema's xs:gYearMonth datatype.

Code: Duration - A time datatype corresponding to W3C XML Schema's xs:duration datatype.

Code: URI - A datatype corresponding to W3C XML Schema's xs:anyURI datatype.

Code: Timespan - A complex datatype specifying a start date (xs:dateTime) and a duration (xs:duration). Note that this is not allowed as thre text type representing a dimension.

Code: Count - A simple incrementing Integer type. The isSequence facet must be set to true, and the interval facet must be set to "1".

Code: InclusiveValueRange - This value indicates that the startValue and endValue attributes provide an inclusive numeric range of type xs:double.

Code: ExclusiveValueRange - This value indicates that the startValue and endValue attributes provide an exclusive numeric range, of type xs:double.

Code: Incremental - This value indicates that the value increments according to the value provided in the interval facet, and has a true value for the isSequence facet.

Code: ObservationalTimePeriod - This is a time datatype, and is the conventional representation of time in SDMX formats. It is a union of W3C XML Schema time datatypes and a set of codes for indicating quarterly, tri-annual, bi-annual, and weekly time periods. See common:TimePeriodType for specifics.

UsageStatusType: UsageStatus provides a list of enumerated types for indicating whether reporting a given metadata attribute is mandatory or conditional.

Restricts xs:NMTOKEN

Code: Mandatory - Reporting the associated attribute is mandatory - a value must be supplied.

Code: Conditional - Reporting the associated attribute is not mandatory - a value may be supplied, but is not required.

RepresentationSchemeTypeType: Representation scheme type provides an enumerated list of valid types of representation schemes.

Restricts xs:NMTOKEN

Code: Codelist - Representation scheme is a codelist.

Code: Concept - Representation scheme is a concept scheme.

Code: Category - Representation scheme is a category scheme.

Code: Organisation - Representation scheme is an organisation scheme.

Code: External - Representation scheme is "external" to the known model - that is, it cannot be enumerated at the time the report is designed. This will only be valid if some maintained and changing object is to have metadata reported against it: for example, if the concepts of dimension objects are to be reported against for all of an agencies' key families, then it is not possible at design time to enumerate all of the concepts which will be used by that agencies' key families into the future. This value should not be used unless absolutely necessary, as it reduces the processability of the metadata report generated.

AttachmentLevelType:

Restricts xs:NMTOKEN

Code: DataSet - Data set level

Code: Group - Group level

Code: Series - Series level

Code: Observation - Observation level

AssignmentStatusType:

Restricts xs:NMTOKEN

Code: Mandatory - Providing attribute value is mandatory

Code: Conditional - Providing attribute value is optional

ToValueTypeType: ToValueTypeType provides an enumeration of available text-equivalents for translation of coded values into textual formats.

Restricts xs:NMTOKEN

Code: Value - Code or other tokenized value, as provided in the representation scheme.

Code: Name - The human-readable name of the Value, as provided in the representation scheme.

Code: Description - The human-readable description of the Value, as provided in the representation scheme.

RepresentationTypeType: RepresentationTypeType provides an enumeration of representation scheme types useful for the mapping of reference metadata concepts to one another.

Restricts xs:NMTOKEN

Code: Codelist - Codelist

Code: CategoryScheme - CategoryScheme

Code: OrganisationScheme - OrganisationScheme

5.3 SDMX Generic Data Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/generic

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

5.3.1 Global Elements

DataSet(DataSetType): The DataSet element contains one or more groups that comprise the data set.

5.3.2 Complex Types

DataSetType: DataSetType defines the structure of a data set. This consists of a key family reference which contains the ID of the key family, and the attribute values attached at the data set level. A DataSet may be used to transmit documentation (that is, only attribute values), data, or a combination of both. If providing only documentation, you need not send the complete set of attributes. If transmitting only data, the Group may be omitted if desired. Uniqueness constraints are defined for the attributes of the data set. If

dataset-level attributes are sent in a delete message, then any valid attribute value will indicate that the current attribute value should be deleted. The keyFamilyURI attribute is provided to allow a URI (typically a URL) to be provided, pointing to an SDMX-ML Structure message describing the key family. Attributes are provided for describing the contents of a data or metadata set, which are particularly important for interactions with the SDMX Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID, and dataflowID all take the IDs specified by the attribute names. The action attribute indicates whether the file is appending, replacing, or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate, and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year, and publicationPeriod specifies the period of publication of the data in terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if that is the time of publication for a data set published on a quarterly basis).

Element Content (Type):

KeyFamilyRef (common:IDType)
Attributes (ValuesType) - min. 0
Group (GroupType) *[Choice]* - min. 0 - max. unbounded
Series (SeriesType) *[Choice]* - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: keyFamilyURI (xs:anyURI) - optional

Attribute: datasetID (common:IDType) - optional

Attribute: dataProviderSchemeAgencyId (common:IDType) - optional

Attribute: dataProviderSchemeId (common:IDType) - optional

Attribute: dataProviderID (common:IDType) - optional

Attribute: dataflowAgencyID (common:IDType) - optional

Attribute: dataflowID (common:IDType) - optional

Attribute: action (common:ActionType) - optional

Attribute: reportingBeginDate (common:TimePeriodType) - optional

Attribute: reportingEndDate (common:TimePeriodType) - optional

Attribute: validFromDate (common:TimePeriodType) - optional

Attribute: validToDate (common:TimePeriodType) - optional

Attribute: publicationYear (xs:gYear) - optional

Attribute: publicationPeriod (common:TimePeriodType) - optional

GroupType: The key values at the group level may be stated explicitly, and all which are not wildcarded listed in GroupKey - they must also all be given a value at the series level. It is not necessary to specify the group key, however, as this may be inferred from the values repeated at the series level. If only documentation (group-level attributes) are being transmitted, however, the GroupKey cannot be omitted. The type attribute contains the name of the declared group in the key family. If any group-level attributes are specified in a delete message, then any valid value supplied for the attribute indicates that the current attribute value should be deleted for the specified attribute.

Element Content (Type):

GroupKey (ValuesType) - min. 0
Attributes (ValuesType) - min. 0
Series (SeriesType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: type (xs:NMTOKEN) - required

SeriesType: SeriesType specifies the structure of a series. This includes all of the key values, values for all the attributes, and the set of observations making up the series content. Messages may transmit only attributes, only data, or both. Regardless, the series key is always required. Key values appear at the Series level in an ordered sequence which corresponds to the key sequence in the key family. A series in a delete message need not supply more than the key, indicating that the entire series identified by that key should be deleted. If series attributes are sent in a delete message, any valid value specified for an attribute indicates that the attribute should be deleted.

Element Content (Type):

SeriesKey (SeriesKeyType)
Attributes (ValuesType) - min. 0
Obs (ObsType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

SeriesKeyType: SeriesKeyType defines the contents of a series key. Each non-time dimension must have a value supplied for it, in the order in which the dimensions are specified in the key family.

Element Content (Type):

Value (ValueType) - max. unbounded

ObsType: ObsType defines the structure of an observation. This includes a time and observation value, as well as values for each of the attributes assigned at the observation level by the key family. In a delete message, only the time need be given, indicating that the observation identified by the key and time should be deleted. For an update message, both time and observation value are required. If any attributes appear in a delete message, any valid value supplied for an attribute indicates that the current value should be deleted.

Element Content (Type):

Time (common:TimePeriodType)
ObsValue (ObsValueType) - min. 0
Attributes (ValuesType) - min. 0
Annotations (common:AnnotationsType) - min. 0

ValuesType:

Element Content (Type):

Value (ValueType) - max. unbounded

ValueType: ValueType is used to assign a single value to a concept, as for attribute values and key values. It has no element content. The startTime attribute is only used if the textFormat of the attribute is of the Timespan type in the key family (in which case the value field takes a duration).

Attribute: concept (common:IDType)

Attribute: value (xs:string)

Attribute: startTime (xs:dateTime) - optional

ObsValueType: ObsValueType describes the information set for an observation value. This is associated with the primary measure concept declared in the key family. The startTime attribute is only used if the textFormat of the observation is of the Timespan type in the key family (in which case the value field takes a duration).

Attribute: value (xs:double)

Attribute: startTime (xs:dateTime) - optional

5.4 SDMX Generic Metadata Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/genericmetadata

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

5.4.1 Global Elements

MetadataSet(MetadataSetType):

5.4.2 Complex Types

MetadataSetType: The Metadata Set is a set of reported metadata against a set of values for a given full or partial target identifier, as described in a metadata structure definition. Child elements include identification of the relevant metadata structure definition using the MetadataStructureRef and MetadataStructureAgencyRef elements. The ReportRef element includes the ID of the report structure as described in the metadata structure definition. AttributeValueSet is a repeatable child element which allows target identifier keys and their associated metadata attribute values to be reported (this functions like a series element does for data sets). An optional name and annotations may also be supplied. The metadataStructureURI allows for a URI to be provided, pointing to the SDMX-ML Structure Message representation of the referenced metadata structure definition. Attributes are provided for describing the contents of a data or metadata set, which are particularly important for interactions with the SDMX Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID, and dataflowID all take the IDs specified by the attribute names. The action attribute indicates whether the file is appending, replacing, or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate, and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year, and publicationPeriod specifies the period of publication of the data in terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if that is the time of publication for a data set published on a quarterly basis).

Element Content (Type):

Name (common:TextType) - min. 0 - max. unbounded
MetadataStructureRef (common:IDType)
MetadataStructureAgencyRef (common:IDType)
ReportRef (common:IDType)
AttributeValueSet (AttributeValueSetType) - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: metadataStructureURI (xs:anyURI) - optional

Attribute: datasetID (common:IDType) - optional

Attribute: dataProviderSchemeAgencyId (common:IDType) - optional

Attribute: dataProviderSchemeId (common:IDType) - optional

Attribute: dataProviderID (common:IDType) - optional

Attribute: dataflowAgencyID (common:IDType) - optional

Attribute: dataflowID (common:IDType) - optional

Attribute: action (common:ActionType) - optional

Attribute: reportingBeginDate (common:TimePeriodType) - optional

Attribute: reportingEndDate (common:TimePeriodType) - optional

Attribute: validFromDate (common:TimePeriodType) - optional

Attribute: validToDate (common:TimePeriodType) - optional

Attribute: publicationYear (xs:gYear) - optional

Attribute: publicationPeriod (common:TimePeriodType) - optional

AttributeValueSetType: The attribute value set provides the values for a set of metadata attributes reported against a target identifier key. The TargetRef element contains the value of the metadata attribute's target attribute in the metadata structure definition (that is, the ID of the full or partial target identifier which is the target of the metadata report). TargetValues is an element substructure which provides the specific full or partial target identifier component values, and the ReportedAttribute sub-element allows for values to be reported against the metadata attributes as described in the referenced metadata structure definition for the referenced full or partial targets.

Element Content (Type):

TargetRef (common:IDType)
TargetValues (TargetValuesType)
ReportedAttribute (ReportedAttributeType) - max. unbounded

TargetValuesType: Target values contains the specific values for each concept in the full or partial target identifier as described in a metadata structure definition. These values typically come from codelists or other item schemes. Each such value should be presented in the order given in the metadata structure definition, and must use a valid representation for that concept. Concepts are those referenced by the identifier components of the target identifiers.

Element Content (Type):

ComponentValue (ComponentValueType) - max. unbounded

ComponentValueType: Component values have an object attribute with an object type value as provided in the metadata structure definition, a component attribute which takes the ID of the identifier component in the metadata structure definition's full target identifier, and a value, which must be a valid value for that concept's representation as described in the metadata structure definition.

[data] (xs:NMTOKEN)

ReportedAttributeType: Reported attributes hold the values which are to be reported against the target specified in the metadata structure definition, and according to the metadata attributes specified for the target referenced in the TargetRef element. Each reported attribute may have Value sub-elements (one per language) if it takes a text or numeric value. The StartTime element is only used if the attribute being represented is of the Timespan type (as described in the corresponding TextFormat element in the metadata structure definition). In this case, the Value takes a duration. Only one such value is allowed in the ReportedAttribute in this case. The types of these values must conform to the limitations described in the metadata structure definition. Also - if permitted by the metadata structure definition - there may be one or more child ReportedAttribute elements. These must be arranged in the nesting hierarchy given in the metadat structure definition. The conceptID attribute provides the id of the concept given in the metadata structure definition to which the reported attribute corresponds.

Element Content (Type):

Value (common:TextType) - min. 0 - max. unbounded
StartTime (xs:dateTime) - min. 0
ReportedAttribute (ReportedAttributeType) - min. 0 - max. unbounded
Annotations (common:AnnotationsType) - min. 0

Attribute: conceptID (common:IDType) - required

5.4.3 Simple Types

ObjectIDType: The Object ID is used to reference a particular Object within the SDMX Information Model's formalization of statistical exchanges.

Restricts xs:NMTOKEN

Code: Agency - Agency

Code: ConceptScheme - Concept scheme

Code: Concept - Concept

Code: Codelist - Codelist

Code: Code - Code

Code: KeyFamily - Key family

Code: Component - Component

Code: KeyDescriptor - Key descriptor

Code: MeasureDescriptor - Measure descriptor

Code: AttributeDescriptor - Attribute descriptor

Code: GroupKeyDescriptor - Group key descriptor

Code: Dimension - Dimension

Code: Measure - Measure

Code: Attribute - Attribute

Code: CategoryScheme - Category scheme

Code: ReportingTaxonomy - Reporting taxonomy

Code: Category - Category

Code: OrganisationScheme - Organisation scheme

Code: DataProvider - Data or metadata provioder

Code: MetadataStructure - Metadata structure definition

Code: FullTargetIdentifier - Full target identifier

Code: PartialTargetIdentifier - Partial target identifier

Code: MetadataAttribute - Metadata attribute

Code: DataFlow - Data flow

Code: ProvisionAgreement - Data or metadata provision agreement

Code: MetadataFlow - Metadata flow

Code: ContentConstraint - Content constraint

Code: AttachmentConstraint - Attachment constraint

Code: DataSet - Data set

Code: XSDataSet - Cross-sectional data set

Code: MetadataSet - Metadata set

Code: HierarchicalCodelist - Hierarchical codelist

Code: Hierarchy - Hierarchy

Code: StructureSet - Structure set

Code: StructureMap - Structure map

Code: ComponentMap - Component map

Code: CodelistMap - Codelist map

Code: CodeMap - Code map

Code: CategorySchemeMap - Category scheme map

Code: CategoryMap - Category map

Code: OrganisationSchemeMap - Organisation scheme map

Code: OrganisationRoleMap - Organisation role map

Code: ConceptSchemeMap - Concept scheme map

Code: ConceptMap - Concept map

Code: Process - Process

Code: ProcessStep - Process step

5.5 SDMX Query Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/query

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

5.5.1 Global Elements

Query(QueryType): The Query message allows standard querying of SDMX-compliant databases and web services. It is intended to be used in non-registry exchanges, and is focused on data sets and metadata sets. It allows queries to retrieve data, metadata, key families, metadata structure definitions, codelists, concepts, and other structural metadata. Note that date and time formats are structured according to the common:TimePeriodType, rather than being specified in the query. The response documents for this query mesage are data formats (for data queries), metadata formats (for metadata queries), and the SDMX Structure MEssage (for all other queries).

5.5.2 Complex Types

QueryType: The Query element is a top-level element for this namespace, which is referenced by the SDMX message envelope, or could be put inside another envelope, such as SOAP. It contains a query. The defaultLimit attribute is the suggested maximum response size in kilobytes.

Element Content (Type):

DataWhere (DataWhereType) - min. 0 - max. unbounded
MetadataWhere (MetadataWhereType) - min. 0 - max. unbounded
KeyFamilyWhere (KeyFamilyWhereType) - min. 0 - max. unbounded
MetadataStructureWhere (MetadataStructureWhereType) - min. 0 - max. unbounded
CodelistWhere (CodelistWhereType) - min. 0 - max. unbounded
ConceptWhere (ConceptWhereType) - min. 0 - max. unbounded
AgencyWhere (AgencyWhereType) - min. 0 - max. unbounded
DataProviderWhere (DataProviderWhereType) - min. 0 - max. unbounded
HierarchicalCodelistWhere (HierarchicalCodelistWhereType) - min. 0 - max. unbounded
ReportingTaxonomyWhere (ReportingTaxonomyWhereType) - min. 0 - max. unbounded
DataflowWhere (DataflowWhereType) - min. 0 - max. unbounded
MetadataflowWhere (MetadataflowWhereType) - min. 0 - max. unbounded
StructureSetWhere (StructureSetWhereType) - min. 0 - max. unbounded
ProcessWhere (ProcessWhereType) - min. 0 - max. unbounded
OrganisationSchemeWhere (OrganisationSchemeWhereType) - min. 0 - max. unbounded
ConceptSchemeWhere (ConceptSchemeWhereType) - min. 0 - max. unbounded
CategorySchemeWhere (CategorySchemeWhereType) - min. 0 - max. unbounded

Attribute: defaultLimit (xs:integer) - optional

DataWhereType: The DataWhere element representes a query for data. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
DataSet (xs:string) *[Choice]*
KeyFamily (xs:string) *[Choice]*
Dimension (DimensionType) *[Choice]*
Attribute (AttributeType) *[Choice]*
Codelist (CodelistType) *[Choice]*
Time (TimeType) *[Choice]*
Category (CategoryType) *[Choice]*
Concept (xs:string) *[Choice]*
DataProvider (xs:string) *[Choice]*
Dataflow (xs:string) *[Choice]*
Version (xs:string) *[Choice]*
Or (OrType) *[Choice]*
And (AndType) *[Choice]*

MetadataWhereType: The MetadataWhere element representes a query for metadata. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
MetadataSet (xs:string) *[Choice]*
MetadataStructure (xs:string) *[Choice]*
StructureComponent (StructureComponentType) *[Choice]*
Attribute (AttributeType) *[Choice]*
Codelist (CodelistType) *[Choice]*
Time (TimeType) *[Choice]*
Category (CategoryType) *[Choice]*
Concept (xs:string) *[Choice]*
DataProvider (xs:string) *[Choice]*
Metadataflow (xs:string) *[Choice]*
Version (xs:string) *[Choice]*
Or (OrType) *[Choice]*
And (AndType) *[Choice]*

AndType: For the And element, each of its immediate child elements represent clauses all of which represent conditions which must be satisfied. If children are A, B, and C, then any legitimate response will meet conditions A, B, and C. Values are the IDs of the referenced object.

Element Content (Type):

DataSet (xs:string) - min. 0 - max. unbounded
MetadataSet (xs:string) - min. 0 - max. unbounded
KeyFamily (xs:string) - min. 0 - max. unbounded
MetadataStructure (xs:string) - min. 0 - max. unbounded
Dimension (DimensionType) - min. 0 - max. unbounded
StructureComponent (StructureComponentType) - min. 0 - max. unbounded
Attribute (AttributeType) - min. 0 - max. unbounded
Codelist (CodelistType) - min. 0 - max. unbounded
Time (TimeType) - min. 0 - max. unbounded

Category (CategoryType) - min. 0 - max. unbounded
Concept (xs:string) - min. 0 - max. unbounded
AgencyID (xs:string) - min. 0 - max. unbounded
DataProvider (xs:string) - min. 0 - max. unbounded
Dataflow (xs:string) - min. 0 - max. unbounded
Metadataflow (xs:string) - min. 0 - max. unbounded
Version (xs:string) - min. 0 - max. unbounded
Or (OrType) - min. 0 - max. unbounded
And (AndType) - min. 0 - max. unbounded

OrType: The Or element's immediate children represent clauses in the query any one of which is sufficient to satisfy the query. If these children are A, B, and C, then any result which meets condition A, or condition B, or condition C is a match for that query. Values are the IDs of the referenced object.

Element Content (Type):

DataSet (xs:string) - min. 0 - max. unbounded
MetadataSet (xs:string) - min. 0 - max. unbounded
KeyFamily (xs:string) - min. 0 - max. unbounded
MetadataStructure (xs:string) - min. 0 - max. unbounded
Dimension (DimensionType) - min. 0 - max. unbounded
StructureComponent (StructureComponentType) - min. 0 - max. unbounded
Attribute (AttributeType) - min. 0 - max. unbounded
Codelist (CodelistType) - min. 0 - max. unbounded
Time (TimeType) - min. 0 - max. unbounded
Category (CategoryType) - min. 0 - max. unbounded
Concept (xs:string) - min. 0 - max. unbounded
AgencyID (xs:string) - min. 0 - max. unbounded
DataProvider (xs:string) - min. 0 - max. unbounded
Dataflow (xs:string) - min. 0 - max. unbounded
Metadataflow (xs:string) - min. 0 - max. unbounded
Version (xs:string) - min. 0 - max. unbounded
Or (OrType) - min. 0 - max. unbounded
And (AndType) - min. 0 - max. unbounded

DimensionType: Dimension elements contain the (single) value being searched on within the key of the data set. The id attribute holds the ID of the dimension. If the content is empty, then the query is for any dimension with the given name. If the name attribute is not supplied, then the query is for the given key value within any dimension.

[data] (xs:string)

StructureComponentType: StructureComponent elements contain the (single) value being searched on within the key of data set, but this value can be either a code value or the alias assigned to a set of equivalent code values. The id attribute holds the ID of the dimension, attribute, or alias assigned to a component in a structure set. If the content is empty, then the query is for any component with the given name or alias. If the name attribute

is not supplied, then the query is for the given code value or alias within any component or component alias.

[data] (xs:string)

AttributeType: Attribute elements contain the (single) value of an attribute being queried for. The id attribute contains the id of the attribute. The attachmentLevel attribute specifies the attachment level of the attribute. If the content of Attribute is empty, then the search is for the specified attribute (and attachment level). If the name attribute is not specified, then the search is on any attribute. If the attachmentLevel attribute is not specified, then the query is for an attribute at any attachment level, as the value defaults to "Any".

[data] (xs:string)

CodelistType: The Codelist element allows queries to specify a (single) value found within a codelist as the element content, and the agency-qualified name of the codelist being queried for in the name attribute. If no content is supplied, then the query is for the named codelist. If the id attribute is left empty, then the value is searched for in any codelist.

[data] (xs:string)

CategoryType: The Category element allows for a search to be made on the values within a specific category, which is specified (in agency-qualified form) with the name attribute. If there is no element content, then the search is for the named Category; if the name is not supplied, then the category value supplied as content should be sought-for in all available categories.

[data] (xs:string)

KeyFamilyWhereType: The KeyFamilyWhere element representes a query for a key family or key families. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
KeyFamily (xs:string) *[Choice]*
Dimension (DimensionType) *[Choice]*
Attribute (AttributeType) *[Choice]*
Codelist (CodelistType) *[Choice]*
Category (CategoryType) *[Choice]*
Concept (xs:string) *[Choice]*
AgencyID (xs:string) *[Choice]*
Version (xs:string) *[Choice]*

Or (OrType) *[Choice]*
And (AndType) *[Choice]*

MetadataStructureWhereType: The MetadataStructureWhere element representes a query for a metadata structure or structures. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
KeyFamily (xs:string) *[Choice]*
MetadataStructure (xs:string) *[Choice]*
StructureSet (xs:string) *[Choice]*
Dimension (DimensionType) *[Choice]*
StructureComponent (StructureComponentType) *[Choice]*
Attribute (AttributeType) *[Choice]*
Codelist (CodelistType) *[Choice]*
Category (CategoryType) *[Choice]*
Concept (xs:string) *[Choice]*
AgencyID (xs:string) *[Choice]*
Version (xs:string) *[Choice]*
Or (OrType) *[Choice]*
And (AndType) *[Choice]*

CodelistWhereType: The CodelistWhere element representes a query for a codelist or codelists. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
Codelist (CodelistType) *[Choice]*
AgencyID (xs:string) *[Choice]*
Version (xs:string) *[Choice]*
Or (OrType) *[Choice]*
And (AndType) *[Choice]*

ConceptWhereType: The ConceptWhere element representes a query for a concept or concepts. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
Concept (xs:string) *[Choice]*
AgencyID (xs:string) *[Choice]*
Version (xs:string) *[Choice]*
Or (OrType) *[Choice]*
And (AndType) *[Choice]*

AgencyWhereType: The AgencyWhere element representes a query for details for an Agency. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
KeyFamily (xs:string) *[Choice]* - min. 0 - max. unbounded
MetadataStructure (xs:string) *[Choice]* - min. 0 - max. unbounded
StructureSet (xs:string) *[Choice]* - min. 0 - max. unbounded
Codelist (CodelistType) *[Choice]* - min. 0 - max. unbounded
Category (CategoryType) *[Choice]* - min. 0 - max. unbounded
Concept (xs:string) *[Choice]* - min. 0 - max. unbounded
AgencyID (xs:string) *[Choice]* - min. 0 - max. unbounded
Or (OrType) *[Choice]* - min. 0 - max. unbounded
And (AndType) *[Choice]* - min. 0 - max. unbounded

DataProviderWhereType: The DataProviderWhere element representes a query for details for a provider of data or metadata sets. It contains all of the clauses in that query, represented by its child elements. Values are the IDs of the referenced object.

Element Content (Type):

(Choice)
DataSet (xs:string) *[Choice]* - min. 0 - max. unbounded
MetadataSet (xs:string) *[Choice]* - min. 0 - max. unbounded
KeyFamily (xs:string) *[Choice]* - min. 0 - max. unbounded
MetadataStructure (xs:string) *[Choice]* - min. 0 - max. unbounded
StructureSet (xs:string) *[Choice]* - min. 0 - max. unbounded
Codelist (CodelistType) *[Choice]* - min. 0 - max. unbounded
Category (CategoryType) *[Choice]* - min. 0 - max. unbounded
Concept (xs:string) *[Choice]* - min. 0 - max. unbounded
AgencyID (xs:string) *[Choice]* - min. 0 - max. unbounded
Or (OrType) *[Choice]* - min. 0 - max. unbounded
And (AndType) *[Choice]* - min. 0 - max. unbounded

TimeType: TimeType contains the time point or period for which results should be supplied. When StartTime and EndTime are used, these must be understood as inclusive.

Element Content (Type):

(Choice)
Time (common:TimePeriodType) *[Choice]*

StructureSetWhereType: The StructureSetWhere element represents a query for a structure set or structure sets. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

HierarchicalCodelistWhereType: The HierarchicalCodelistWhere element represents a query for a hierarchical codelist or codelists. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

ReportingTaxonomyWhereType: The ReportingTaxonomyWhere element represents a query for a reporting taxonomy or taxonomies. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

DataflowWhereType: The DataflowWhereType element represents a query for a dataflow or dataflows. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

MetadataflowWhereType: The MetadataflowWhereType element represents a query for a metadataflow or metadataflows. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0

ID (xs:string) - min. 0
Version (xs:string) - min. 0

ProcessWhereType: The ProcessWhere element represents a query for a process or processes. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

OrganisationSchemeWhereType: The OrganisationSchemeWhere element represents a query for an organisation scheme or schemes. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

ConceptSchemeWhereType: The ConceptSchemeWhere element represents a query for a concept scheme or schemes. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

CategorySchemeWhereType: The CategorySchemeWhere element represents a query for a category scheme or schemes. Like other maintainable objects, it must be queried for using information about its agency, ID, and/or version. Any field not supplied will be taken as matching all of that type.

Element Content (Type):

AgencyID (xs:string) - min. 0
ID (xs:string) - min. 0
Version (xs:string) - min. 0

5.5.3 Simple Types

AttachmentLevelType: This type supplies an enumeration of attachment levels corresponding to those in the SDMX Information Model, plus a value of "Any" where the search is wildcarded.

Restricts xs:NMTOKEN

Code: DataSet - Attached at the Data Set level

Code: Group - Attached at the Group level

Code: Series - Attached at the Series level

Code: Observation - Attached at the Observation level

Code: Any - Attached at any attachment level

5.6 SDMX Common Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common

Imports: http://www.w3.org/XML/1998/namespace (xml.xsd)

5.6.1 Complex Types

ConstraintType: Constraint specifies the object to which constraints are attached. Note that if the constraint is that for a Data Provider, then only ReleaseCalendar information is relevant, as there is no reliable way of determining which key family is being used to frame constraints in terms of cube regions or key sets. ReferencePeriod is used to report start date and end date constraints. MetadataConceptSet allows for conten t constraints to be described for metadata sets.

Element Content (Type):

ConstraintID (IDType)
CubeRegion (CubeRegionType) - min. 0 - max. unbounded
MetadataConceptSet (MetadataConceptSetType) - min. 0
KeySet (KeySetType) - min. 0 - max. unbounded
ReleaseCalendar (ReleaseCalendarType) - min. 0
ReferencePeriod (ReferencePeriodType) - min. 0

Attribute: ConstraintType (ConstraintTypeType) - required

CubeRegionType: CubeRegion describes the portion(s) of the possible combinations of all components within a key family or metadata structure definition by providing valid values on a per-component basis. This does not guarantee that data will be available for all possible combinations, but describes the portion of the cube in which it is useful to query for data. The isIncluded attribute, if true, indicates that the described area is the one in which it is useful to search/expect to find data. If false, this means that the portions of the cube outside the described region are useful to search/where you may expect to find data.

Element Content (Type):

Member (MemberType) - max. unbounded

Attribute: isIncluded (xs:boolean) - required

MetadataConceptSetType: The isIncluded attribute, if true, indicates that the described concepts - of those described as possibilities in the relevant metadata structure definition - are reported. If the value is false, then the specified concepts are not reported.

Element Content (Type):

Member (MemberType) - max. unbounded

Attribute: isIncluded (xs:boolean) - required

MemberType: Member describes the constrained component - which can be a dimension, an attribute, a metadata attribute, or a measure. This must agree with the metadata structure definition or key family referenced by the Provision Agreement's Dataflow or Metadataflow. The isIncluded attribute indicates whether the Member is listing included or excluded values for each component, as seen against the full valid set described in the key family. When used to describe reported metadata, the MemberValue may be omitted in cases where no specification is made regarding the representation of the concept (as is the case with un-coded metadata attributes). Otherwise, MemberValue must be included.

Element Content (Type):

ComponentRef (IDType)
MemberValue (MemberValueType) - min. 0 - max. unbounded

Attribute: isIncluded (xs:boolean) - required

MemberValueType: MemberValue specifies the value of the specified component, which must be a valid value as described in the appropriate structure definition (key family).

Element Content (Type):

Value (xs:string)

KeySetType: KeySet describes a set of keys. The isIncluded attribute, if true, indicates that the specified keys are valid keys within the constraint. If false, the set of keys described are not valid - all other possible keys are the valid ones.

Element Content (Type):

Key (KeyType)

Attribute: isIncluded (xs:boolean) - required

KeyType: Key allows for sets of component references - holding the name of the component's concept - and a permitted value for that component. This conctruct can be repeated as many times as dessired, but must describe complete keys according to teh relevant structure definition (key family).

Element Content (Type):

ComponentRef (IDType)
Value (xs:string)

ReleaseCalendarType: The ReleaseCalendar holds information about the timing of releases of the constrained data. Periodicity is the period between releases of the data set. Offset is the interval between January first and the first release of data within the year. Tolerance is the period after which the release of data may be deemed late. All of these values use the standard "P7D"-style format.

Element Content (Type):

Periodicity (xs:string)
Offset (xs:string)
Tolerance (xs:string)

ReferencePeriodType: Specifies the inclusive start and end times for a registry query.

Attribute: startTime (xs:dateTime) - required

Attribute: endTime (xs:dateTime) - required

TextType: TextType provides for a set of language-specific alternates to be provided for any human-readable construct in the instance.

[data] (xs:string)

AnnotationType: AnnotationType provides for non-documentation notes and annotations to be embedded in data and structure messages. It provides optional fields for providing a title, a type description, a URI, and the text of the annotation.

Element Content (Type):

AnnotationTitle (xs:string) - min. 0
AnnotationType (xs:string) - min. 0
AnnotationURL (xs:anyURI) - min. 0
AnnotationText (TextType) - min. 0 - max. unbounded

AnnotationsType: AnnotationsType provides for a list of annotations to be attached to data and structure messages.

Element Content (Type):

Annotation (AnnotationType) - max. unbounded

5.6.2 Simple Types

ConstraintTypeType: ConstraintType provides an enumeration of values of the types of constraints.

Restricts xs:NMTOKEN

Code: Content - Content constraint.

Code: Attachment - Attachment constraint.

PeriodType: PeriodType provides a list of tokens for specifying common periods: Quarterly: Q1, Q2, Q3, Q4; Weekly: W1 - W52; Triannual: T1, T2, T3; Biannual: B1, B2. These values appear after a four-digit year indicator, followed by a dash (ie, 2005-Q1).

Restricts xs:string

TimePeriodType: TIME_PERIOD is not completely expressable in XML Schema's date type: instead we use the union of dateTime, date,

gYearMonth, and gYear. The default name for the concept is TIME_PERIOD. Bi-annual, tri-annual, quarterly, and weekly periods have special formats (see PeriodType, above), but other periods would be described in terms of their beginning date or time (eg, a period of a decade is identified with a four-digit year corresponding to the decades' first year).

ActionType: ActionType provides a list of actions, describing the intention of the data transmission from the sender's side. Each action provided at the dataset or metadataset level applies to the entire dataset for which it is given. Note that the actions indicated in the Message Header are optional, and used to summarize specific actions indicated with this data type for all registry interactions. The "Informational" value is used when the message contains information in response to a query, rather than being used to invoke a maintenance activity.

Restricts xs:NMTOKEN

> Code: Append - Data or metadata is an incremental update for an existing data/metadata set or the provision of new data or documentation (attribute values) formerly absent. If any of the supplied data or metadata is already present, it will not replace that data or metadata. This corresponds to the "Update" value found in version 1.0 of the SDMX Technical Standards.
>
> Code: Replace - Data/metadata is to be replaced, and may also include additional data/metadata to be appended.
>
> Code: Delete - Data/Metadata is to be deleted.
>
> Code: Information - Informational

IDType: IDType provides a type which is used for restricting the characters in codes and IDs throughout all SDMX-ML messages. Valid characters include A-Z, a-z, @, 0-9, _, -, $.

Restricts xs:string

5.7 SDMX Registry Interfaces Namespace Module

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/registry

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)
Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/structure (SDMXStructure.xsd)

5.7.1 Complex Types

SubmitSubscriptionRequestType: The SubmitSubscriptionRequest element is submitted to the registry to subscribe to registration and change events for specific registry resources.

Element Content (Type):

Subscription (SubscriptionType) - max. unbounded

SubmitSubscriptionResponseType: The SubmitSubscriptionResponse element contains information which describes the success or failure of a Subscription, providing any error messages in the event of failure. It also returns the registry URN of the subscription, and the subscriber-assigned ID.

Element Content (Type):

SubscriptionURN (xs:anyURI) - min. 0
SubscriberAssignedID (common:IDType) - min. 0
SubscriptionStatus (StatusMessageType)

NotifyRegistryEventType: The NotifyRegistryEvent element is sent by the registry services to subscribers, to notify them of specific registration and change events. EventTime specifies the time of the triggering event. ObjectURN provides the URN of the object on which the event occurred. SubscriptionURN provides the registry/repository URN of the subscription. EventAction indicates the nature of the event - whether the action was an addition, a modification, or a deletion.

Element Content (Type):

EventTime (xs:dateTime)
ObjectURN (xs:anyURI)
SubscriptionURN (xs:anyURI)
EventAction (common:ActionType)
StructuralEvent (StructuralEventType) *[Choice]*
ProvisioningEvent (ProvisioningEventType) *[Choice]*
RegistrationEvent (RegistrationEventType) *[Choice]*

SubmitRegistrationRequestType: SubmitRegistrationRequest is sent to the registry by an agency or data/metadata provider to request registration for a data set or metadata set. The resource to be registered must be accessible to the registry services at an indicated URL, so that it can be processed by those services. This is the datasource, which may also have been specified for the data provider or provision agreement, in which case it need not appear here. Constraints describing the content and release calendar of the registered dataset (and, for metadata sets, the release calendar) may also be included.

Element Content (Type):

Registration (RegistrationType) - max. unbounded

SubmitRegistrationResponseType: This document is sent to the agency or data/metadata provider in response to a registration request. It indicates the success or failure of the registration request, and contains any error messages generated by the registration service.

Element Content (Type):

RegistrationStatus (RegistrationStatusType) - max. unbounded

QueryRegistrationRequestType: The QueryRegistrationRequest is used to query the contents of a registry for data sets and metadata sets. The QueryRegistrationRequest specifies whether the result set should include metadata sets, data sets, or both with the QueryType element. The constraints which characterize the search - including reference period, are contained in teh Constraints within the child object references.

Element Content (Type):

QueryType (QueryTypeType)
ProvisionAgreementRef (ProvisionAgreementRefType) *[Choice]*
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
DataProviderRef (DataProviderRefType) *[Choice]*

QueryRegistrationResponseType: The QueryRegistrationResponse is sent as a response document to anyone querying the contents of a registry. The results set contains a set of links to data and/or metadata If the result set is null, or there is some other problem with the query, then appropriate error messages and statuses will be returned.

Element Content (Type):

QueryResult (QueryResultType) - max. unbounded

SubmitStructureRequestType: SubmitStructureRequest is used to submit structure definitions - key families, metadata structures - to the repository. The structure resources (key families, agencies, concepts and concept schemes, codelists,etc.) to be submitted must be available as valid SDMX-ML Structure messages external to the registry, so that they can be retrieved by the repository submission service. A SubmitStructureResponse will be sent in response, and will indicate status and contain anyt relevant error information. StructureLocation holds the URL of the valid Structure Message. Alternately, the Structure element can contain the structural descriptions. The

SubmittedStructureType contains a reference to one of the structural maintainable artefacts detailed in the Structure Message, which is to be submitted to the repository. It does not need to be used when the structures being submitted are included in the request message, or when all objects in the referenced SDMX-ML Structure message are to submitted.

Element Content (Type):

StructureLocation (xs:anyURI) *[Choice]*
Structure (StructureType) *[Choice]*
SubmittedStructure (SubmittedStructureType) - min. 0 - max. unbounded

SubmitStructureResponseType: SubmitStructureResponse is returned by the registry when a SubmitStructure is received. It indicates the status of the submission, and carries any error messages which are generated, if relevant. For each submitted structure, a SubmissionResult will be returned.

Element Content (Type):

SubmissionResult (SubmissionResultType) - max. unbounded

QueryStructureRequestType: QueryStructureRequest is used to query the registry for any maintainable object within the repository. The response is a Structure message. In the reference elements to the queryable registry objects, a valid registry URN or a complete set of other child elements may be used to identify the objects desired in the result set. Any part of an element-based (that is, non-URN) identification of an object which is not provided will be understood as a wild-card value, referring to "all" possible values. The resolveReferences attribute is set to true if all dependent objects should also be returned as part of the result set. (For example, if you query for a key family and want to also have all codelists, concepts, and agencies, returned as well, resolveReferences should be set to true.)

Element Content (Type):

(Choice)
AgencyRef (AgencyRefType) *[Choice]*
DataProviderRef (DataProviderRefType) *[Choice]*
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
CodelistRef (CodelistRefType) *[Choice]*
CategorySchemeRef (CategorySchemeRefType) *[Choice]*
ConceptSchemeRef (ConceptSchemeRefType) *[Choice]*
OrganisationSchemeRef (OrganisationSchemeRefType) *[Choice]*
KeyFamilyRef (KeyFamilyRefType) *[Choice]*
MetadataStructureRef (MetadataStructureRefType) *[Choice]*
HierarchicalCodelistRef (HierarchicalCodelistRefType) *[Choice]*
StructureSetRef (StructureSetRefType) *[Choice]*
ProcessRef (ProcessRefType) *[Choice]*
ReportingTaxonomyRef (ReportingTaxonomyRefType) *[Choice]*

Attribute: resolveReferences (xs:boolean) - required

QueryStructureResponseType: QueryStructureResponse is sent in response to a QueryStructureRequest. It carries the status of the response, with any relevant error messages, and then also carries all information found in the rersult set.

Element Content (Type):

StatusMessage (StatusMessageType)
OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
Dataflows (structure:DataflowsType) - min. 0
Metadataflows (structure:MetadataflowsType) - min. 0
CategorySchemes (structure:CategorySchemesType) - min. 0
CodeLists (structure:CodeListsType) - min. 0
HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
Concepts (structure:ConceptsType) - min. 0
MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) - min. 0
KeyFamilies (structure:KeyFamiliesType) - min. 0
StructureSets (structure:StructureSetsType) - min. 0
ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0
Processes (structure:ProcessesType) - min. 0

SubmitProvisioningRequestType: This document is sent to the registry services to submit provisioning information. A provision agreement is typically sent, wghich has internal references to existing data rpvodiers and dataflows/metadataflows. These elements are also included as possible separate submissions, because it may be necessary to provide datasource and constraint information independent of the establishment of a provision agreement.

Element Content (Type):

(Choice)
ProvisionAgreement (ProvisionAgreementType) *[Choice]*
DataProviderRef (DataProviderRefType) *[Choice]*
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*

SubmitProvisioningResponseType: The ProvisioningResponse element is returned by the registry services in response to a provisioning request. It contains information about the status of the submitted provisioning information, and any relevant error messages in case of failure.

Element Content (Type):

ProvisioningStatus (ProvisioningStatusType) - max. unbounded

QueryProvisioningRequestType: QueryProvisioningRequest is used to query the repository for provisioning metadata. The response is a QueryProvisioningResponse document, carrying either the result set of the query or relevant error messages. Note that whatever information is presented here, regarding provision agreements, data flow, metadataflow, or data providers, is taken to be the search criteria - the query is for all provision agreements which match the supplied criteria. If any of provision agreement, metadataflow, dataflow, or data provider are omitted, the search will apply to all values for those objects in the repository.

Element Content (Type):

ProvisionAgreementRef (ProvisionAgreementRefType) - min. 0
DataflowRef (DataflowRefType) - min. 0
MetadataflowRef (MetadataflowRefType) - min. 0
DataProviderRef (DataProviderRefType) - min. 0

QueryProvisioningResponseType: The QueryProvisioningResponse element is returned in response to queries regarding provisioning information. It carries either the provisioning information making up the result set, or relevant status messages containing errors or warnings, or both. The references to Dataflow, Metadataflow, and Data Provider are included in those cases where the result set has these objects, but not associated with any Provisioning Agreement.

Element Content (Type):

ProvisionAgreement (ProvisionAgreementType) - min. 0 - max. unbounded
DataflowRef (DataflowRefType) - min. 0 - max. unbounded
MetadataflowRef (MetadataflowRefType) - min. 0 - max. unbounded
DataProviderRef (DataProviderRefType) - min. 0 - max. unbounded
StatusMessage (StatusMessageType)

SubscriptionType: Subscriptions submit a subscription for a registry or repository object.Action indicates what action is being taken by sending the request. RegistryURN is used to identify the subscription in the case of deletion or modification. NotificationMAilTo holds an e-mail address (the "mailto:" protocol); NotificationHTTP holds an http address to which notifications can be addressed as POSTs. SubscriberAssignedID allows the subscriber to specify an ID which will be returned as part of the notification for the subscribed events. Validity period sets a start and end date for the subscription, EventSelector indicates an event or events for the subscription.

Element Content (Type):

Action (common:ActionType)
RegistryURN (xs:anyURI) - min. 0
NotificationMailTo (xs:anyURI) - min. 0

NotificationHTTP (xs:anyURI) - min. 0
SubscriberAssignedID (common:IDType) - min. 0
ValidityPeriod (ValidityPeriodType)
EventSelector (EventSelectorType)

ValidityPeriodType: Specifies inclusive start and end-dates for the subscription period.

Element Content (Type):

StartDate (xs:date)
EndDate (xs:date)

EventSelectorType: Allows subscribers to specify registry and repository events for which they wish to receive notifications.

Element Content (Type):

StructuralRepositoryEvents (StructuralRepositoryEventsType) - min. 0
ProvisioningRepositoryEvents (ProvisioningRepositoryEventsType) - min. 0
DataRegistrationEvents (DataRegistrationEventsType) - min. 0
MetadataRegistrationEvents (MetadataRegistrationEventsType) - min. 0

StructuralRepositoryEventsType: Contains details of the subscribed structural repository events. AgencyID specifies an agency for the object or objects indicated in the other ID fields. Note that the ID fields (including AgencyID) may hold a complete ID or Repository URN, but may also insert the "%" wildcard character, which represents 0 or more characters, in the ID string. If left empty, all objects will be matched within the other constraints (agency, object type) provided.

Element Content (Type):

AgencyID (common:IDType) - min. 0 - max. unbounded
AllEventsID (xs:string) - min. 0 - max. unbounded
KeyFamilyID (xs:string) - min. 0 - max. unbounded
ConceptSchemeID (xs:string) - min. 0 - max. unbounded
CodeListID (xs:string) - min. 0 - max. unbounded
MetadataStructureID (xs:string) - min. 0 - max. unbounded
CategorySchemeID (xs:string) - min. 0 - max. unbounded
DataflowID (xs:string) - min. 0 - max. unbounded
MetadataflowID (xs:string) - min. 0 - max. unbounded
OrganisationSchemeID (xs:string) - min. 0 - max. unbounded
HierarchicalCodelistID (xs:string) - min. 0 - max. unbounded
StructureSetID (xs:string) - min. 0 - max. unbounded
ReportingTaxonomyID (xs:string) - min. 0 - max. unbounded
ProcessID (xs:string) - min. 0 - max. unbounded

ProvisioningRepositoryEventsType: Contains details of the subscribed provisioning repository events. Note that the ID fields may hold a complete ID

or Repository URN, but may also insert the "%" wildcard character, which represents 0 or more characters, in the ID string. If left empty, all objects will be matched within the other constraints (agency, object type) provided.

Element Content (Type):

ProvisionAgreementID (common:IDType) - min. 0 - max. unbounded
DataProviderID (xs:string) - min. 0 - max. unbounded
DataflowID (xs:string) - min. 0 - max. unbounded
MetadataflowID (xs:string) - min. 0 - max. unbounded
AllEventsID (xs:string) - min. 0 - max. unbounded

DataRegistrationEventsType: Contains details of the subscribed data registry events. Note that the ID fields may hold a complete ID or Registry URN, but may also insert the "%" wildcard character, which represents 0 or more characters, in the ID string. If left empty, all objects will be matched within the other constraints (agency, object type) provided.

Element Content (Type):

AllEventsID (xs:string) - min. 0 - max. unbounded
DataProviderID (xs:string) - min. 0 - max. unbounded
ProvisionAgreementID (xs:string) - min. 0 - max. unbounded
DataflowID (xs:string) - min. 0 - max. unbounded
KeyFamilyID (xs:string) - min. 0 - max. unbounded
CategoryID (xs:string) - min. 0 - max. unbounded
CategorySchemeID (xs:string) - min. 0 - max. unbounded
CategorySchemeAgencyID (xs:string) - min. 0 - max. unbounded

MetadataRegistrationEventsType: Contains details of the subscribed metadadata registry events. Note that the ID fields may hold a complete ID or Registry URN, but may also insert the "%" wildcard character, which represents 0 or more characters, in the ID string. If left empty, all objects will be matched within the other constraints (agency, object type) provided.

Element Content (Type):

AllEventsID (xs:string) - min. 0 - max. unbounded
DataProviderID (xs:string) - min. 0 - max. unbounded
ProvisionAgreementID (xs:string) - min. 0 - max. unbounded
MetadataflowID (xs:string) - min. 0 - max. unbounded
MetadatastructureID (xs:string) - min. 0 - max. unbounded
CategoryID (xs:string) - min. 0 - max. unbounded

StructuralEventType: This provides the details of a structural repository event for the purposes of notification.

Element Content (Type):

OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
Dataflows (structure:DataflowsType) - min. 0
Metadataflows (structure:MetadataflowsType) - min. 0
CategorySchemes (structure:CategorySchemesType) - min. 0
CodeLists (structure:CodeListsType) - min. 0
HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
Concepts (structure:ConceptsType) - min. 0
MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) - min. 0
KeyFamilies (structure:KeyFamiliesType) - min. 0
StructureSets (structure:StructureSetsType) - min. 0
Processes (structure:ProcessesType) - min. 0
ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0

ProvisioningEventType: This provides the details of a provisioning event for the purposes of notification.

Element Content (Type):

DataProviderRef (DataProviderRefType) - min. 0
DataflowRef (DataflowRefType) - min. 0
MetadataflowRef (MetadataflowRefType) - min. 0
ProvisionAgreement (ProvisionAgreementType) - min. 0

RegistrationEventType: This provides the details of a data or metadata registration event for the purposes of notification.

Element Content (Type):

Registration (RegistrationType)

ProvisionAgreementType: Provision agreements contain a reference to a pre-existing data flow or metadata flow definition and a pre-existing data (or metadata) provider in the registry. They also must have the action attribute set, indicating whether this is an addition, a modification, or a deletion of a provision agreement. They may also supply boolean values which describe how the registry must behave: if indexTimeSeries is true, then the registry must index all time series when a data set is registered against this provision agreement; if indexDataSet is true, then the registry must index the range of actual (present) values for each dimension of the Dataset (as indicated in the dataset's key family); if indexReportingPeriod is true, then the registry must index the time period ranges for which data are present in the dataset(s) registered against the provision agreement. Note that the values for these attributes are not needed when a Delete action is indicated. As for all identifiable objects, provision agreements have Name and Description elements, which are repeatable to provide language-specific forms. These may be ommitted if the provision agreement already exists (as is the case for modification and deletion); descriptions are always optional. The id attribute holds the unique id of the provision agreement as derived (according to teh

logical registry specifcation.) If specified, the uri attribute points to a location (typically a URL) where the provision agreement is described in a valid QueryProvisioningResponse message. The urn attribute holds the reserved registry URN assigned to the provision agreement - this must be included when the ProvisionAgreement structure is used in a response document of any type. The action attribute must be specified when the provisionAgreement is used in a request document of any type.. Datasource is used to describe the data source associated with the provider agreement. Annotations may be provided in the Annotations element. The validFrom and validTo attributes provide inclusive dates for providing supplemental validity information about the version.

Element Content (Type):

Name (common:TextType) - min. 0 - max. unbounded
Description (common:TextType) - min. 0 - max. unbounded
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
DataProviderRef (DataProviderRefType)
Datasource (QueryableDatasourceType) - min. 0
Constraint (common:ConstraintType) - min. 0
Annotations (common:AnnotationsType) - min. 0

Attribute: id (common:IDType) - optional

Attribute: uri (xs:anyURI) - optional

Attribute: urn (xs:anyURI) - optional

Attribute: action (common:ActionType) - optional

Attribute: indexTimeSeries (xs:boolean) - optional

Attribute: indexDataSet (xs:boolean) - optional

Attribute: indexReportingPeriod (xs:boolean) - optional

Attribute: validFrom (common:TimePeriodType) - optional

Attribute: validTo (common:TimePeriodType) - optional

DatasourceType: Datasource specifies the properties of a data or metadata source. A SimpleDatasource requires only the URL of the data. A QueryableDatasource must be able to accept an SDMX-ML Query Message, and respond appropriately. Either or both may be specified.

Element Content (Type):

SimpleDatasource (xs:anyURI) - min. 0
QueryableDatasource (QueryableDatasourceType) - min. 0

QueryableDatasourceType: If the data provider uses a single, queryable data source for all provision agreements contained in the ProvisionRequest element, then this element should be used to describe the queryable datasource. DataUrl contains the URL of the datasource, with WSDLUrl optionally providing the location of a WSDL instance on the internet which deascribes the queryable datasource. The attributes isRESTDatasource and isWebServiceDatasource indicate, if true, that the queryable datasource is accessible via the REST protocol and/or Web Services protocols, respectively.

Element Content (Type):

DataUrl (xs:anyURI)
WSDLUrl (xs:anyURI) - min. 0

Attribute: isRESTDatasource (xs:boolean) - required

Attribute: isWebServiceDatasource (xs:boolean) - required

ProvisioningStatusType: For each provision agreement, dataflow reference, metadataflow reference, or data provider reference submitted in a provisioning request, a provisioning status will be returned, providing a status and any warnings or errors.

Element Content (Type):

ProvisionAgreementRef (ProvisionAgreementRefType) *[Choice]*
DataProviderRef (DataProviderRefType) *[Choice]*
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
StatusMessage (StatusMessageType)

RegistrationType: Registration provides the information needed for data and reference metadata set registration. LastUpdated can provide a time stamp; ValidFrom and ValidTo allow for effectivity, so that data visibility from the registry can be controlled by the registrant. A Datasource must be supplied here if not already provided in the provision agreement. The data set or metadata set must be associated with a provision agreement, a metadataflow, or a dataflow definition. If possible, the provision agreement should be specified. Only in cases where this is not possible should the dataflow or metadataflow be used.

Element Content (Type):

LastUpdated (xs:dateTime) - min. 0
ValidFrom (xs:dateTime) - min. 0
ValidTo (xs:dateTime) - min. 0
Action (common:ActionType)
Datasource (DatasourceType) - min. 0
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
ProvisionAgreementRef (ProvisionAgreementRefType) *[Choice]*

RegistrationStatusType: Each RegistrationStatus reports the status of a submitted data set or metadata set registration, and carries any error messages. If successful, the Datasource which has been registered is returned, and a reference to the provision agreement, dataflow, metadataflow, or data provider is returned.

Element Content (Type):

StatusMessage (StatusMessageType)
Datasource (DatasourceType) - min. 0
DataProviderRef (DataProviderRefType) - min. 0
DataflowRef (DataflowRefType) - min. 0
MetadaflowRef (MetadataflowRefType) - min. 0
ProvisionAgreementRef (ProvisionAgreementRefType) - min. 0

QueryResultType: QueryResult contains the results of a specific registry query for a single datasource. If a successful result is a registered dataset, the DataResult element is used. If a successful result is a registered metadataset, the MetadataResult is used. If the query failed, then StatusMessage is included. The timeSeriesMatch attribute is true when the result is an exact match with the key found in the registry - that is, when the registered datasource provides a matching key. It is set to false when the datasource is registered with cube-region constraints, or in any other circumstance when it cannot be established that the sought-for keys have been exactly matched. This is always true when the resulting datasource is the source of a metadata set.

Element Content (Type):

(Choice)
DataResult (ResultType) *[Choice]*
MetadataResult (ResultType) *[Choice]*
StatusMessage (StatusMessageType) *[Choice]*

Attribute: timeSeriesMatch (xs:boolean) - required

ResultType: Result contains the information about either a data or metadata source, being returned as part of a QueryResult element. If the

Element Content (Type):

Datasource (DatasourceType) - min. 0
ProvisionAgreementRef (ProvisionAgreementRefType) *[Choice]*
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (DataflowRefType) *[Choice]*
DataProviderRef (DataProviderRefType) *[Choice]*

StructureType: Holds the structure information for submission to the structural repository.

Element Content (Type):

OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
Dataflows (structure:DataflowsType) - min. 0
Metadataflows (structure:MetadataflowsType) - min. 0
CategorySchemes (structure:CategorySchemesType) - min. 0
CodeLists (structure:CodeListsType) - min. 0
HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
Concepts (structure:ConceptsType) - min. 0
MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) - min. 0
KeyFamilies (structure:KeyFamiliesType) - min. 0
StructureSets (structure:StructureSetsType) - min. 0
Processes (structure:ProcessesType) - min. 0
ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0

SubmittedStructureType: SubmittedStructure holds a reference to a structural object to be stored in the repository. The externalDependencies attribute should be set to true if the repository is expected to use URLs in the structure Message to retrieve objects on which the stored object has dependencies. (Thus, if a key family is being submitted to the repository, and the structure message has URLKs which point to the locations of the codelists it uses, then the externalDependencies attribute should be set to true.) the action attribute specifies whether the Structure being submitted is intended to be added or deleted from the repository. The "modify" action is not applicable to final structures in the repository, and will produce an error condition, as these can be versioned but not modified.To submit a later version of a structure, the structure message should include the incremented version number. The externalDependencies and action attributes need not be specified in a response document. The isFinal attribute indicates whether the structure being submitted to the repository is final or not - this can also be specified on the structures themselves.

Element Content (Type):

(Choice)
DataflowRef (DataflowRefType) *[Choice]*
MetadataflowRef (MetadataflowRefType) *[Choice]*
CodelistRef (CodelistRefType) *[Choice]*
HierarchicalCodelistRef (HierarchicalCodelistRefType) *[Choice]*
CategorySchemeRef (CategorySchemeRefType) *[Choice]*
ConceptSchemeRef (ConceptSchemeRefType) *[Choice]*

OrganisationSchemeRef (OrganisationSchemeRefType) *[Choice]*
KeyFamilyRef (KeyFamilyRefType) *[Choice]*
MetadataStructureRef (MetadataStructureRefType) *[Choice]*
ProcessRef (ProcessRefType) *[Choice]*
StructureSetRef (StructureSetRefType) *[Choice]*
ReportingTaxonomyRef (ReportingTaxonomyRefType) *[Choice]*

Attribute: externalDependencies (xs:boolean) - optional

Attribute: action (common:ActionType) - optional

Attribute: isFinal (xs:boolean) - optional

SubmissionResultType: For each Structure object submitted to the repository in a SubmitStructure, a SubmissionResult will be returned. It will identify the object submitted, report back the action requested, and convey the status and any error messages which are relevant to the submission.

Element Content (Type):

SubmittedStructure (SubmittedStructureType)
StatusMessage (StatusMessageType)

ProvisionAgreementRefType: ProvisionAgreementRef allows for the identification of a provision agreement. At a minimum, either the URN element - holding a valid registry URN - or the set of OragnisationSchemeAgencyID, OrganisationSchemeID, DataProviderID, DataflowAgencyID, and DataflowID must be specified. When used in a response document of any type, the URN must always be provided. Datasource can be used to specify a datasource for the provision agreement. Constraint can be used to express constraints associated with the provision agreement.

Element Content (Type):

URN (xs:anyURI) - min. 0
OrganisationSchemeAgencyID (common:IDType) - min. 0
OrganisationSchemeID (common:IDType) - min. 0
DataProviderID (common:IDType) - min. 0
DataProviderVersion (xs:string) - min. 0
DataflowAgencyID (common:IDType) - min. 0
DataflowID (common:IDType) - min. 0
DataflowVersion (xs:string) - min. 0
Datasource (DatasourceType) - min. 0
Constraint (common:ConstraintType) - min. 0

MetadataflowRefType: The MetadataflowRef type structures a reference to a metadataflow definition. This requires that ID are provided for a pre-existing Agency and Metadataflow Definition in the registry. The Version element may be used to specify the version of the indicated dataflow. If absent, the most

recent version is assumed. The URN element is used to provide the registry-specific URN as an alternate means of identification. When used in a response document of any type, the URN must always be provided. At a minimum, either the URN element or AgencyID, MetadataflowID, and (optionally) version must be supplied. Datasource may be used to specify a datasource. Constraint can be used to provide constraints associated with the metadataflow.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
MetadataflowID (common:IDType) - min. 0
Version (xs:string) - min. 0
Datasource (DatasourceType) - min. 0
Constraint (common:ConstraintType) - min. 0

DataflowRefType: The DataflowRef type structures a reference to a dataflow definition. This requires that ID are provided for a pre-existing Agency and Dataflow Definition in the registry. The Version element may be used to specify the version of the indicated dataflow. If absent, the most recent version is assumed. The URN element is used to provide the registry-specific URN as an alternate means of identification. At a minimum, either the URN element or AgencyID, DataflowID, and (optionally) version must be supplied. When used in a response document of any type, the URN must always be provided. Datasource may be used to specify a datasource. Constraints can be used to specify constraints associated with the dataflow.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
DataflowID (common:IDType) - min. 0
Version (xs:string) - min. 0
Datasource (DatasourceType) - min. 0
Constraint (common:ConstraintType) - min. 0

DataProviderRefType: The DataProviderRef type structures a reference to a data provider. This requires that IDs be provided for an organisation scheme, its maintenance agency, and the data prov ider as identified in the referenced organisation scheme. The Version element may be used to specify the version of the indicated data provider. If absent, the most recent version is assumed. The URN element is used to provide the registry-specific urn as an alternate means of identification. At a minimum, either the URN element or OrgansisationSchemeID, OrganisationSchemeAgencyID, DataProviderID, and (optionally) Version must be supplied. When used in a response document of any type, the URN must always be provided. Datasource may be used to specify a datasource. Constraints can be used to specify constraints associated with the data provider.

Element Content (Type):

URN (xs:anyURI) - min. 0
OrganisationSchemeAgencyID (common:IDType)
OrganisationSchemeID (common:IDType)
DataProviderID (common:IDType)
Version (xs:string) - min. 0
Datasource (DatasourceType) - min. 0
Constraint (common:ConstraintType) - min. 0

AgencyRefType: The AgencyRef type structures a reference to an Agency. This requires that IDs be provided for an organisation scheme, its maintenance agency, and the agency as identified in the referenced organisation scheme. The Version element may be used to specify the version of the indicated agency. If absent, the most recent version is assumed. The URN element is used to provide the registry-specific urn as an alternate means of identification. At a minimum, either the URN element or OrgansisationSchemeID, OrganisationSchemeAgencyID, AgencyID, and (optionally) Version must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
OrganisationSchemeAgencyID (common:IDType)
OrganisationSchemeID (common:IDType)
AgencyID (common:IDType)
Version (xs:string) - min. 0

CodelistRefType: KeyFamilyRef allows for references to specific codelists. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
CodelistID (common:IDType) - min. 0
Version (xs:string) - min. 0

CategorySchemeRefType: CategorySchemeRef allows for references to specific category schemes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
CategorySchemeID (common:IDType) - min. 0
Version (xs:string) - min. 0

ConceptSchemeRefType: ConceptSchemeRef allows for references to specific concept schemes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
ConceptSchemeID (common:IDType) - min. 0
Version (xs:string) - min. 0

OrganisationSchemeRefType: OrganisationSchemeRef allows for references to specific organisation schemes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
OrganisationSchemeID (common:IDType) - min. 0
Version (xs:string) - min. 0

KeyFamilyRefType: KeyFamilyRef allows for references to specific key families (data structure definitions). At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
KeyFamilyID (common:IDType) - min. 0
Version (xs:string) - min. 0

MetadataStructureRefType: MetadataStructureRef allows for references to specific metadata structure definitions. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements

must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
MetadataStructureID (common:IDType) - min. 0
Version (xs:string) - min. 0

HierarchicalCodelistRefType: HierarchicalCodelistRef allows for references to specific hierarchical codelists. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
HierarchicalCodelistID (common:IDType) - min. 0
Version (xs:string) - min. 0

StructureSetRefType: StructureSetRef allows for references to specific structure sets. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
StructureSetID (common:IDType) - min. 0
Version (xs:string) - min. 0

ProcessRefType: ProcessRef allows for references to specific processes. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
ProcessID (common:IDType) - min. 0
Version (xs:string) - min. 0

ReportingTaxonomyRefType: ReportingTaxonomyRef allows for references to specific reporting taxonomies. At a minimum, either the URN - which contains a valid Registry/Repository URN - or the rest of the child elements must be supplied. When used in a response document of any type, the URN must always be provided.

Element Content (Type):

URN (xs:anyURI) - min. 0
AgencyID (common:IDType) - min. 0
ReportingTaxonomyID (common:IDType) - min. 0
Version (xs:string) - min. 0

StatusMessageType: This carries the text of error messages and/or warnings in response to queries or requests. The Status attribute carries the staus of the query or request.

Element Content (Type):

MessageText (common:TextType) - min. 0 - max. unbounded

Attribute: status (StatusType) - required

5.7.2 Simple Types

ConstraintTypeType: ConstraintType provides an enumeration of values of the types of constraints.

Restricts xs:NMTOKEN

Code: Content - Content constraint.

Code: Attachment - Attachment constraint.

StatusType: StatusType provides an enumeration of values that detail the status of queries or requests.

Restricts xs:NMTOKEN

Code: Success - Query or request successful.

Code: Warning - Query or request successful, but with warnings.

Code: Failure - Query or request not successful.

QueryTypeType: QueryType provides an enumeration of values which specify the objects in the result-set for a registry query.

Restricts xs:NMTOKEN

Code: DataSets - Only references data sets should be returned.

Code: MetadataSets - Only references to metadata sets should be returned.

Code: AllSets - References to both data sets and metadata sets should be returned.

5.8 Data Formatting and Character Encoding

In all SDMX-ML documents – whether key-family-specific or not - the character encoding must be UTF-8. To simplify the exchange of statistical data and metadata globally, restrictions also apply to the expression of numeric formats: the decimal separator is always a period (“.”). There is no character used to separate thousands in data.

5.9 Missing Observation Values

In some of the SDMX-ML documents, an Observation is required (as in the Utility format) or it is desirable to indicate that a numerical value does not exist. While this information may be captured in an Observation-level attribute such as OBS_STATUS, with a code indicating that the value for the observation is missing, there is also a way to reliably indicate this state in the data itself. For this purpose, missing observation values – when included in an SDMX-ML data file – should be indicated using “NaN”. In XML, this indicates “not a number”, but is still valid in numeric fields. This avoids having to use a number (such as “-9999999” or “0”), along with a status code of “missing” (or similar construct) to indicate missing numeric values.

6 KEY-FAMILY- AND METADATA-STRUCTURE-DEFINITION-SPECIFIC SCHEMAS: CORE STRUCTURES & STANDARD MAPPINGS

Some schemas are specific to key families and metadata structure definitions, and therefore there is no single schema for all users. In these cases, standard mappings are provided so that even though one schema cannot be published, the schemas can be predicted from an examination of SDMXStructure messages that describe the key families/metadata structure definitions on which they are based. Automatic creation of these structure-specific schemas according to these mappings is a natural consequence of this correspondence, and free tools to enable this creation of structure-specific schemas is envisioned.

It is important to note that all key-family- and metadata-structure-definition-specific schemas are based on a core of identical constructs, allowing the smallest possible number of tags to differ from schema to schema. This section first documents these “core” structures, each in their own SDMX-maintained namespace module, and then discusses the mappings from a key family to the key-family-specific schema, and from a metadata-structure-definition to a metadata-structure-definition-specific schema.

These schemas are all as similar as possible. They vary according to where in the common structure key values and attributes may be specified. A less obvious difference is seen in the Utility and Metadata Report schemas, which are designed to carry as much structural metadata as possible in order to allow “typical” XML tools (such as schema-guided editors and parsers) to benefit from the availability of this data - such tools are generally incapable of consulting the key family or metadata structure definition for structural metadata.

Note that for all key-family-specific and metadata-structure-definition-specific schemas, the namespaces must be constructed following the rules for registry URN identifiers, as described in section 5.2 of the SDMX Registry Interfaces specification, with the addition of a single field at the end of the URN:

- For Utility schemas: “:utility”
- For Compact schemas: “:compact”
- For Cross-Sectional schemas: “:cross”
- For Metadata Report schemas: “:metadatareport”

6.1 Compact Data Message Core Structure

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/compact

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

6.1.1 Global Elements

DataSet(DataSetType): The DataSet element contains the data set.

Group(GroupType): The Group element contains the group.

Series(SeriesType): The Series element contains the series.

Obs(ObsType): The Obs element contains the observations.

6.1.2 Complex Types

DataSetType: DataSetType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element. Attributes are provided for describing the contents of a data or metadata set, which are particularly important for interactions with the SDMX Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID, and dataflowID all take the IDs specified by the attribute names. The action attribute indicates whether the file is appending, replacing, or deleting.

Attributes reportingBeginDate, reportingEndDate, validFromDate, and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year, and publicationPeriod specifies the period of publication of the data in terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if that is the time of publication for a data set published on a quarterly basis).

Attribute: keyFamilyURI (xs:anyURI) - optional

Attribute: datasetID (common:IDType) - optional

Attribute: dataProviderSchemeAgencyId (common:IDType) - optional

Attribute: dataProviderSchemeId (common:IDType) - optional

Attribute: dataProviderID (common:IDType) - optional

Attribute: dataflowAgencyID (common:IDType) - optional

Attribute: dataflowID (common:IDType) - optional

Attribute: action (common:ActionType) - optional

Attribute: reportingBeginDate (common:TimePeriodType) - optional

Attribute: reportingEndDate (common:TimePeriodType) - optional

Attribute: validFromDate (common:TimePeriodType) - optional

Attribute: validToDate (common:TimePeriodType) - optional

Attribute: publicationYear (xs:gYear) - optional

Attribute: publicationPeriod (common:TimePeriodType) - optional

GroupType: GroupType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

SeriesType: SeriesType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

ObsType: ObsType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

6.2 Utility Data Message Core Structure

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/utility

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

6.2.1 Global Elements

DataSet(DataSetType): DataSet exists to act as the head of a substitution group to which key-family-specific attributes and elements are bound.

Group(GroupType): Group exists to act as the head of a substitution group to which key-family-specific attributes and elements are bound.

Series(SeriesType): Series exists to act as the head of a substitution group to which key-family-specific attributes and elements are bound.

Key(KeyType): Key is an element which serves as the head of a substitution group containing the key-family-specific key values.

Obs(ObsType): Obs exists to act as the head of a substitution group to which key-family-specific attributes and elements are bound.

6.2.2 Complex Types

DataSetType: DataSetType acts as a structural base, which is extended through the addition of attributes and elements to reflect the particular needs of a specific key family using the xs:extends element. Attributes are provided for describing the contents of a data or metadata set, which are particularly important for interactions with the SDMX Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID, and dataflowID all take the IDs specified by the attribute names. The action attribute indicates whether the file is appending, replacing, or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate, and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year, and publicationPeriod specifies the period of publication of the data in terms of

whatever provisioning agreements might be in force (ie, "Q1 2005" if that is the time of publication for a data set published on a quarterly basis).

Attribute: keyFamilyURI (xs:anyURI) - optional

Attribute: datasetID (common:IDType) - optional

Attribute: dataProviderSchemeAgencyId (common:IDType) - optional

Attribute: dataProviderSchemeId (common:IDType) - optional

Attribute: dataProviderID (common:IDType) - optional

Attribute: dataflowAgencyID (common:IDType) - optional

Attribute: dataflowID (common:IDType) - optional

Attribute: action (common:ActionType) - optional

Attribute: reportingBeginDate (common:TimePeriodType) - optional

Attribute: reportingEndDate (common:TimePeriodType) - optional

Attribute: validFromDate (common:TimePeriodType) - optional

Attribute: validToDate (common:TimePeriodType) - optional

Attribute: publicationYear (xs:gYear) - optional

Attribute: publicationPeriod (common:TimePeriodType) - optional

GroupType: GroupType acts as a structural base, which is renamed and extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

SeriesType: SeriesType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

KeyType: KeyType describes the abstract type which defines the Key element.

ObsType: ObsType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

6.3 Cross-Sectional Data Message Core Structure

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

6.3.1 Global Elements

DataSet(DataSetType): DataSet contains the data set.

Group(GroupType): Group contains the group.

Section(SectionType): Section contains the data section.

Obs(ObsType): Obs contains the observation, with one or more measures.

6.3.2 Complex Types

DataSetType: DataSetType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element. Attributes are provided for describing the contents of a data or metadata set, which are particularly important for interactions with the SDMX Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID, and dataflowID all take the IDs specified by the attribute names. The action attribute indicates whether the file is appending, replacing, or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate, and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year, and publicationPeriod specifies the period of publication of the data in terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if that is the time of publication for a data set published on a quarterly basis).

Attribute: keyFamilyURI (xs:anyURI) - optional

Attribute: datasetID (common:IDType) - optional

Attribute: dataProviderSchemeAgencyId (common:IDType) - optional

Attribute: dataProviderSchemeId (common:IDType) - optional

Attribute: dataProviderID (common:IDType) - optional

Attribute: dataflowAgencyID (common:IDType) - optional

Attribute: dataflowID (common:IDType) - optional

Attribute: action (common:ActionType) - optional

Attribute: reportingBeginDate (common:TimePeriodType) - optional

Attribute: reportingEndDate (common:TimePeriodType) - optional

Attribute: validFromDate (common:TimePeriodType) - optional

Attribute: validToDate (common:TimePeriodType) - optional

Attribute: publicationYear (xs:gYear) - optional

Attribute: publicationPeriod (common:TimePeriodType) - optional

GroupType: GroupType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

SectionType: SectionType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element.

ObsType: ObsType acts as a structural base, which is extended through the addition of attributes to reflect the particular needs of a specific key family using the xs:extends element. It is capable of expressing the value and attributes of any single available cross-sectional measure (when extended).

6.4 Metadata Report Core Structure

http://www.SDMX.org/resources/SDMXML/schemas/v2_0/metadatareport

Imports: http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common (SDMXCommon.xsd)

6.4.1 Global Elements

MetadataSet(MetadataSetType): The MetadataSet element contains metadata-structure-specific report(s)described in a single metadata structure definition. Attributes are provided for describing the contents of a data or metadata set, which are particularly important for interactions with the SDMX Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID, and dataflowID all take the IDs specified by the attribute names. The action attribute indicates whether the file is appending, replacing, or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate, and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year, and publicationPeriod specifies the period of publication of the data in terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if that is the time of publication for a data set published on a quarterly basis).

6.4.2 Complex Types

MetadataSetType: MetadataReportType defines the structure of a metadata structure definition-specific Metadata Report. This consists of a MetadataStructureRef which holds the ID fo the metadata structure, and MetadataStructureAgencyRef, which holds the ID of the maintraenance agency of the metadata structure, and Version, which provides the version number of the referenced metadata structure definition. If not provided, version is assumed to be "1.0". This type is designed to be extended to hold the metadata-structure-sepcific fields needed to validate a specific metadata report.

Element Content (Type):

MetadataStructureRef (common:IDType)
MetadataStructureAgencyRef (common:IDType)
Version (xs:string) - min. 0

Attribute: MetadataStructureURI (xs:anyURI) - optional

Attribute: datasetID (common:IDType) - optional

Attribute: dataProviderSchemeAgencyId (common:IDType) - optional

Attribute: dataProviderSchemeId (common:IDType) - optional

Attribute: dataProviderID (common:IDType) - optional

Attribute: dataflowAgencyID (common:IDType) - optional

Attribute: dataflowID (common:IDType) - optional

Attribute: action (common:ActionType) - optional

Attribute: reportingBeginDate (common:TimePeriodType) - optional

Attribute: reportingEndDate (common:TimePeriodType) - optional

Attribute: validFromDate (common:TimePeriodType) - optional

Attribute: validToDate (common:TimePeriodType) - optional

Attribute: publicationYear (xs:gYear) - optional

Attribute: publicationPeriod (common:TimePeriodType) - optional

---

6.5 Mappings to Key-Family-Specific Data Schemas

6.5.1 General Rules:

For all key-family-specific schemas (Compact, Utility, and Cross-Sectional) SDMX provides a namespace to be used as the extension base for key-family-specific schemas of that type. The key-family-specific schema will be created in its own target name space, owned and maintained by the creating agency. It will use the targetNamespace attribute of the schema element to identify the namespace which contains the key-family-specific schema. The namespace module provided by SDMX for that class of key-family-specific schema will be incorporated using the import element in the key-family-specific schema. The SDMX Common namespace module must also be imported into the schema. Other xml:namespace attributes may be added to the schema element as needed.

The elementFormDefault attribute on the schema element will be given a value of "qualified", and the attributeFormDefault attribute on the schema element will be given a value of "unqualified".

All additions to the SDMX module will be made using the extends element from W3C XML Schema. The term "levels of structure," when referring to the imported SDMX modules, include the following:

- DataSet level
- Group level
- Series level
- Observation level

These levels normally refer to the element provided by the SDMX module to which attributes and elements may be assigned. In some cases, specific named constructs in the key family will become members of a set of elements corresponding to one of the levels named above.

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

Note that for all of the following mappings the term "concept name" is the value of the id attribute of the concept as found in the SDMX-ML message describing the key family.

6.5.2 Representations and Datatypes

For all key-family-specific schemas, the information about permitted datatypes found in the `structure:TextFormat` element for dimensions and attributes (including observation values, which are technically an attribute) are represented with a standard set of datatypes from W3C XML Schema. The table below shows many of these – the first column corresponds to the `structure:TextType` attribute of the `structure:TextFormat` element, and the second column shows how this value will be represented in the generated schemas.

SDMX-ML Data Type	XML Schema Data Type
`String`	`xs:string`
`Big Integer`	`xs:integer`
`Integer`	`xs:int`
`Long`	`xs.long`
`Short`	`xs:short`
`Decimal`	`xs:decimal`
`Float`	`xs:float`
`Double`	`xs:double`
`Boolean`	`xs:boolean`
`DateTime`	`xs:dateTime`
`Time`	`xs:time`
`Date`	`xs:date`
`Year, Month, Day, MonthDay, YearMonth`	`xs:gYear, xs:gMonth, xs:gDay, xs:gMonthDay, xs:gYearMonth`

SDMX-ML Data Type	XML Schema Data Type
	(respectively)
Duration	xs:duration
URI	xs:anyURI

There are a set of additional text types which cannot be expressed with a simple correspondence to W3C XML Schema:

Timespan: This is a complex datatype, requiring both a startDate expressed as an `xs:dateTime` and a duration, expressed as an `xs:duration` type in the generated schemas. Depending on which type of generated schema is being discussed, these two fields will be either XML attributes or XML elements in the generated schema. Note that Timespan may not be used as the representation of a key value (that is, as the representation of a dimension).

Count: This is represented in the generated XML Schema as `xs:interval.` Note, however, that it represents a sequential number, as indicated in the facets of the `structure:TextType` element.

InclusiveValueRange/ExclusiveValueRange: These text types require a single value which must fall between the specified start and end values as per the facets on the `structure:TextType` element. The value is expressed in the generated schema as `xs:double`.

Incremental: This is again a single value which is of type xs:double, but must be expressed in increments as per the interval facet of the `structure:TextFormat` element.

Note also that the facets of the structure:TextFormat element may provide further restrictions on the values described in the key family. These should be bound into the generated XML schema as facets on the simple types declared to represent the contents of elements where present.

In all cases where a facet must be expressed, a global XSD simple type will be declared which has the name of the concept it is representing followed by the string “Type”, as described below. This type can then have XSD facets expressed as per the following:

minLength: XSD restriction is used to set the minLength value.
maxLength: XSD restriction is used to set the maxLength value.
startValue: XSD restriction is used to set the minInclusive value.
endValue: XSD restriction is used to set the maxInclusive value.
decimals: XSD restriction is used to set the fractionDigits value.
pattern: This is expressed as a pattern facet on the declared simple type.

Other facets are informational, and are not expected to be expressed in the generated schema for validation.

6.5.3 Use of W3C Schema Extension in XML Mappings

It is permissible to use schema extension and refinement (that is, xs:extends and xs:restricts) as well as xs:include to organize a set of SDMX-ML schemas. While this is not required, it is sometimes useful to organize a large set of similar schemas.

If these techniques are used, however, the resulting XML instance must be identical to those instances marked-up according to schemas which do not employ them. This rule includes the use of XML namespaces – that is, they must be identical in all respects between instances marked-up according to XML schemas which use xs:extends, xs:restricts, and xs:include, and those which do not.

6.5.4 Compact Schemas:

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

A global element named “DataSet” will be declared, with an XSD substitutionGroup attribute which has a value referencing the DataSet element in the “compact:” namespace. Its type attribute will reference DataSetType in the key-family-specific namespace.

An XSD complexType will be declared named “DataSetType”. It will have XSD complexContent containing an XSD extension element, with a base attribute of DataSetType in the “compact:” namespace. The extension will consist of an XSD choice element, with a minOccurs attribute with a value of “0” and a maxOccurs value of “unbounded”. The choice will contain an XSD element reference for each named group declared in the key family. They will each have an XSD ref attribute with a value of the group id provided in the key family. (These elements will take the names of the group ids declared in the key family.) Additionally, an XSD element will be declared in the choice with a ref attribute with a value of Series. Further, an element named Annotations will be added to the choice, with a type of AnnotationsType from the “common:” namespace.

For each attribute declared in the key family with an attachmentLevel of “DataSet”, an XML attribute will also be declared in the extension. It will have the same name as the attribute’s concept in the key family. It will have a “use” attribute value of “optional”. For coded attributes, the XML attribute will be given a type value which is the name of the codelist which represents it. In the key-family-specific namespace, this codelist will be represented by a simpleType declaration which contains a list of enumerations, equivalent to the values of the codelist, as described in the key family. These will be extensions of the XSD "string" datatype. The enumerated values will be the values of the codes. The descriptions of the codes will be placed inside XSD "documentation" elements, contained in XSD "annotation" elements, which are themselves contained in the XSD "enumeration" elements as the first instance of the XSD documentation element. No other text shall occur within this particular instance of the XSD documentation element, although other XSD documentation elements may occur within any given XSD enumeration element.

Uncoded attributes will also be represented with XSD simpleType elements declared in the key-family-specific namespace, with names formed by taking the name of the

attribute in the key family and appending “Type” to them. If unrestricted, these will be of the W3C XML Schema primitive type “string”. Otherwise, bindings will be as described above in the Representations and Datatypes section.

For each dimension declared in the key family, an XML attribute will be declared, with a name which is the name of the concept referenced by the dimension. For coded dimensions, the XML attribute will be given a type value which is the name of the codelist which represents it. In the key-family-specific namespace, this codelist will be represented by a simpleType declaration which contains a list of enumerations, equivalent to the values of the codelist, as described in the key family (and for coded attributes, above). All data typing with the TextFormat element will be implemented as provided for attributes, above. The “use” attribute for the dimension XML attribute declaration will have a value of “optional”.

For each named Group in the key family, a global XSD element will be declared, taking the id of the group. Its XSD type attribute will have a value formed by taking the name of the element and adding “Type” to the end of it. It will have a substitutionGroup attribute which references the Group element declared in the “compact:” namespace.

An XSD complexType will be declared for each named group declared in the key family, with a name formed by taking the name of the group in the key family and appending “Type” to it. It will have an XSD complexContent element which contains an XSD extends with a base attribute value of compact:GroupType. The extends will contain an XSD sequence element. An element named Annotations will be added to the end of the sequence, with a type of AnnotationsType from the “common:” namespace. It will also have a minOccurs value of “0”.

For each attribute in the key family with an attachmentLevel of “Group”, an XSD attribute element will be added to the extends element, with a use attribute set to “optional” and a type attribute defined as for the DataSet level, above. The name will be the concept name of the attribute in the key family.

For each dimension referenced by DimensionRef element in the named Group declaration in the key family XML , an XSD attribute element will also be added to the extends element, with a use attribute set to “required” and a type defined as for coded attributes for the dataset level, above. The name will be the concept name of the dimension in the key family.

A XSD global element named Series will be declared in the key-family-specific namespace, with a type of SeriesType and a substitutionGroup attribute referencing compact:Series.

An XSD complexType will then be declared with a name of SeriesType. It will have XSD complexContent, with an XSD extension element that has a base attribute value of compact:SeriesType. The extends element will contain an XSD sequence element, which will contain an XSD element with a ref attribute whose value is “Obs”. Its minOccurs attribute will have a value of “0” and a maxOccurs value of “unbounded”. An element named Annotations will be added to the end of the sequence, with a type of AnnotationsType from the “common:” namespace. It will also have a minOccurs value of “0”.

For each attribute in the key family with an attachmentLevel of “Series”, an XSD attribute element will be added to the extends element, with a use attribute set to “optional” and a type attribute defined as for the DataSet level, above. The name will be the name of the attribute’s concept in the key family. The exception is where an attribute has an isTimeFormat attribute value of “true” – in this case, it is treated the same as other series-level attributes except that its use attribute has a value of “required”.

For each dimension declared in the key family, an XML attribute will be declared, with a name which is the name of the concept referenced by the dimension. For coded dimensions, the XML attribute will be given a type value which is the name of the codelist which represents it. In the key-family-specific namespace, this codelist will be represented by a simpleType declaration which contains a list of enumerations, equivalent to the values of the codelist, as described in the key family (and for coded attributes, above). All data typing with the TextFormat element will be implemented as provided for attributes, above. The “use” attribute for the dimension attribute declaration will have a value of “optional”.

An XSD global element will be declared named “Obs”. It will have a substitutionGroup attribute with a value “compact:Obs”. It will have a type of “ObsType”.

An XSD complexType element will be declared with a name “ObsType” and an XSD complexContent. This will contain an XSD extends element with a base attribute of “compact:ObsType”. It will contain an XSD sequence element. The sequence element will contain an element named Annotations, with a type of AnnotationsType from the “common:” namespace. It will have a minOccurs value of “0”.

The extension element will also have an XSD attribute element in it, which will have a name attribute whose value is the name of the TimeDimension concept from the key family. It will have a use attribute of “optional” and a type of “common:TimePeriodType”.

The extension element will also have an XSD attribute element in it, which will have a name attribute whose value is the concept name of the primary measure from the key family. It will have a use attribute of “optional” and a type as described for attributes, above. If the codelist attribute references a codelist, then a simple type must be declared as indicated above. Otherwise, data typing should be done as for other constructs using the TextFormat element to describe the data format.

For each attribute declared in the key family with an attachmentLevel of “Observation”, an XSD attribute will be added to the extends. Each XSD attribute will take the name of the attribute’s concept declared in the key family, and will have a use attribute of “optional”. Its type will be defined as for the DataSet-level attributes described above.

No other declarations or constructs will be added to the schemas created using this mapping.

Time Ranges in CompactData: Unlike any other SDMX-ML data format, the key-family-specific CompactData format can express a set of observation values without having to provide a time for each of them. If a Series has a time provided for the first

observation, subsequent observations in the series may omit the time, and only provide an observation value (a value for the attribute named after the primary measure), and whatever attributes are needed (see below). The times of the subsequent observations can be calculated according to the frequency specified by the relevant time format attribute value (or, failing that, the frequency dimension value), which can be calculated by the application. Note that support for this functionality is not mandatory for applications which do not claim this support in their conformance statements. It is also permissible to supply a time value for the last observation in the series, to permit double-checking of the calculation, although this is not mandatory.

Delete and Update Messages in CompactData: In the Header element and action attribute at the DataSet level, the action field specifies whether a message is an update message (to append or replace) or a delete message for thr purposes of bilateral exchange. If it is an update message, it is used to send new information or updated information, which may include only data, only documentation (that is, attribute values as described in the key family), or both. (Agreements regarding the use of update messages should be specified between counterparties.) For a delete message, the requirements are that a complete series key always be sent for the deletion of data, which is identified either as an entire series by the absence of any specified time periods, or for a specific set of time periods, by the inclusion of those time periods. Attribute values may be deleted by sending a complete or partial set of attributes, with any valid value for the attribute (according to the XSD schema) being taken as an indication that the current attribute value should be deleted.

6.5.5 Cross-Sectional Schemas

Key-family-specific cross-sectional schemas express all non-time-series-based presentations of the data which are made possible in the key family. They also are capable of expressing statistical data for which time is not a concept – that is, they can provide the only SDMX-ML format for data which is inherently only cross-sectional. As with the CompactData format, key values and attribute values are attached to a four-level structure as XML attributes. For cross-sectional data, however, the term “Series” – an abbreviation of “time series” – is replaced by the equivalent “Section” construct.

Please note that named groups declared in the key family are ignored for the purposes of the cross-sectional data format. They are replaced by a generic Group element, leaving it up to the writing or processing application to enforce the validity of attribute values for groups of Sections. This is true also because a single SDMX-ML cross-sectional schema may be described in the key family such that it allows for more than one dimension to be expressed at the observation level, replacing the role of time in time-series-oriented formats, and therefore allows key values and attribute values to be attached at more than one level.

A global element named “DataSet” will be declared, with an XSD substitutionGroup attribute which has a value referencing the DataSet element in the “cross:” namespace. Its type attribute will reference DataSetType in the key-family-specific namespace.

An XSD complexType will be declared named “DataSetType”. It will have XSD complexContent containing an XSD extension element, with a base attribute of

DataSetType in the “cross:” namespace. The extension will consist of an XSD choice element, with a minOcurs of “0” and a maxOccurs of ”unbounded”. The choice element will contain an XSD element reference with a value of “Group”. Additionally, an XSD element will be declared in the choice with a ref attribute, whose value is Section. Further, an element named Annotations will be added to the choice, with a type of AnnotationsType from the “common:” namespace. It will have a minOccurs attribute of “0”.

For each attribute or dimension declared in the key family with a crossSectionalAttachDataSet of “true”, an XML attribute will also be declared in the extension. It will have the same name as the attribute concept or dimension concept in the key family. It will have a “use” attribute value of “optional”. For coded attributes and dimensions, the XML attribute will be given a type value which is the name of the codelist which represents it. In the key-family-specific namespace, this codelist will be represented by a simpleType declaration which contains a list of enumerations, equivalent to the values of the codelist, as described in the key family. These will be extension of the XSD "string" datatype. The enumerated values will be the values of the codes. The descriptions of the codes will be placed inside XSD "documentation" elements, contained in XSD "annotation" elements, which are themselves contained in the XSD "enumeration" elements as the first instance of the XSD documentation element. No other text shall occur within this particular instance of the XSD documentation element, although other XSD documentation elements may occur within any given XSD enumeration element.

Un-coded dimensions will have XML attributes declared as above, but will have their data-typing mapped differently. All data typing with the TextFormat element will be implemented as provided for uncoded attributes, below, with the exception that Timespan is not permitted as the representation of a dimension.

Uncoded attributes will also be represented with XSD simpleType elements declared in the key-family-specific namespace, with names formed by taking the name of the attribute concept in the key family and appending “Type” to them. If unrestricted, these will be of the W3C XML Schema primitive type “string”; otherwise, mappings will be as per the Representations and Datatypes section, above. If the textType of an attribute value is a Timespan, then two attributes will be declared – one as per usual, which will be of type xs:duration, and the other will have a name value of the attribute’s concept with “StartTime” appended to it, and it will have a value of “xs:duration”.

A Global XSD element will be declared named Group. Its XSD type attribute will have a value of GroupType. It will have a substitutionGroup attribute which references the Group element declared in the “cross:” namespace.

An XSD complexType named GroupType will be declared. It will have an XSD complexContent element which contains an XSD extends with a base attribute value of compact:GroupType. The extends will contain an XSD sequence element, which will contain an XSD element with a reference to the element Section. Its minOccurs attribute will have a value of “0” and a maxOccurs value of “unbounded”. An element named Annotations will be added to the end of the sequence, with a type of AnnotationsType from the “common:” namespace. It will also have a minOccurs value of “0”.

For each attribute or dimension in the key family with a crossSectionalAttachGroup value of “true” or an isFrequencyDimension value of “true”, an XSD attribute element will be added to the extends element, with a use attribute set to “optional” and a type attribute defined as for the DataSet level, above. The name will be the name of the attribute concept or dimension concept in the key family.

A XSD global element named Section will be declared in the key-family-specific namespace, with a type of SectionType and a substitutionGroup attribute referencing compact:Section.

An XSD complexType will then be declared with a name of SectionType. It will have XSD complexContent, with an XSD extension element that has a base attribute value of cross:SectionType. The extends element will contain an XSD choice element with a minOccurs of “0” and a maxOccurs of “unbounded”, which will contain an XSD element for each CrossSectionalMeasure declared in the key family, with a ref attribute whose value is the name of the measure’s concept. An element named Annotations will be added to the end of the choice, with a type of AnnotationsType from the “common:” namespace.

For each attribute or dimension in the key family with a crossSectionalAttachSection value of “true”, an XSD attribute element will be added to the extends element, with a use attribute set to “optional” and a type attribute defined as for the DataSet level, above. The name will be the name of the attribute concept or dimension concept in the key family.

An XSD global element will be declared for each CrossSectionalMeasure declared in the key family, with the name of the measure’s concept. It will have a substitutionGroup attribute with a value “cross:Obs”. It will have a type of “ObsType”. If no CrossSectionalMeasures have been declared, use the PrimaryMeasure instead.

An XSD complexType element will be declared for each CrossSectionalMeasure declared in the key family with a name created by appending “Type” to the concept of the measure. These declarations will contain an XSD complexContent. This will contain an XSD extends element with a base attribute of “cross:ObsType”. It will contain an XSD sequence element. The sequence element will contain an element named Annotations, with a type of AnnotationsType from the “common:” namespace. It will have a minOccurs value of “0”.

The extension element will also have an XSD attribute element in it for each attribute or dimension which has a crossSectionalAttachObservation value of “true” and lists the name of the measure’s concept in an AttachmentMeasure element in its declaration. The XSD attribute will take its name value from the name of the attribute’s concept. It will have a use attribute of optional, and a type as described for the DataSet level, above. Additionally, an attribute will be declared with a name of “value” and a type created by appending the string “SimpleType” to the name of the containing Measure. Its use attribute will be “optional”. (Note that the dimension whose coded representation corresponds to the CrossSectionalMeasures should never have its crossSectionalAttachObservation attribute set to “true”.) For each of the “value” attributes, a global XSD simple type will be declared, with a name created by appending “SimpleType” to the Measure corresponding to the value attribute. The base of the simple type will be xs:string. The type will be restricted as per the

Representations and datatypes section, above, with the one exception that if the type of any given attribute is “Timespan,” an additional attribute will be declared as a sibling to the “value” attribute, with a name of “startTime”, a value of xs:dateTime, and a use attribute of optional. The type of the generated simple type will in this case be “xs:duration”.

If no CrossSectionalMeasures were declared in the key family, there will be an XSD attribute element added to the extension, which will have a name attribute whose value is the concept name of the PrimaryMeasure concept from the key family. It will have a use attribute of “optional” and a type value as described for attributes and dimensions.

In this case, for each attribute declared in the key family with an attachmentLevel of “Observation”, an XSD attribute will be added to the extends. Each XSD attribute will take the name of the attribute’s concept declared in the key family, and will have a use attribute of “optional”. Its type will be defined as for the DataSet-level attributes described above. Additionally, an attribute will be declared with a name of value and a type value as described for attributes and dimensions. Its use attribute is “optional”.

No other declarations or constructs will be added to the schemas created using this mapping.

Delete and Messages in CrossSectionalData: In the Header element and in the action attribute at DataSet level, the action field specifies whether a message is an update message (Append, Replace) or a delete message for the purposes of bilateral exchange. If it is an update message, it is used to send new information or updated information, which may include only data, only documentation (that is, attribute values as described in the key family), or both. (Agreements regarding the use of update messages should be specified between counterparties.) For a delete message, the requirements are that a complete key always be sent for the deletion of data, which is identified either as an entire series by the absence of any specified time periods, or for a specific set of time periods, by the inclusion of those time periods. Attribute values may be deleted by sending a complete or partial set of attributes, with any valid value for the attribute (according to the XSD schema) being taken as an indication that the current attribute value should be deleted.

6.5.6 Utility Schemas

Utility schemas are different from the Compact and Cross-Sectional schemas because they differentiate between the expression of the attributes and dimensions established in the key family. This design serves to preserve the ordering of the keys - the design provides much of the key-family structural metadata without requiring the processor to access the XML structure message describing the key family. This makes the rules inherent in the structure of the key family available to such tools as schema-guided XML editors, which are part of the primary reason for the Utility schema format.

The Utility schema employs a technique similar to the Compact and Cross-Sectional schemas by creating substitution groups which are headed by elements at the DataSet, Group, Series, and Observation levels. This is done in such a way that the messages can be more completely validated with a generic XML parser but are considerably larger in size than the CompactData or CrossSectionalData formats.

A global element named “DataSet” will be declared, with an XSD substitutionGroup attribute which has a value referencing the DataSet element in the “utility:” namespace. Its type attribute will reference DataSetType in the key-family-specific namespace.

An XSD complexType will be declared named “DataSetType”. It will have XSD complexContent containing an XSD extension element, with a base attribute of DataSetType in the “utility:” namespace. The extension will consist of an XSD sequence element containing first an XSD choice element, with a maxOccurs value of “unbounded”. The choice will contain an XSD element reference for each named group declared in the key family. They will each have an XSD ref attribute with a value of the group name provided in the key family. (These elements will take the names of the groups declared in the key family.) If there are no named groups declared in the key family, an XSD element will be declared in the choice with a ref attribute with a value of Series. An element named Annotations will be added to the end of the sequence, with a type of AnnotationsType from the “common:” namespace and a minOccurs attribute of “0”.

For each attribute declared in the key family with an attachmentLevel of “DataSet”, an XML attribute will be declared in the extension. It will have the same name as the attribute’s concept in the key family. It will have a use attribute with a value of “required” if the attribute declared in the key family has an assgnmentStatus of “Mandatory”, and a use attribute with a value of optional if its assignmentStatus in the key family is “Conditional”. For coded attributes, the XML attribute will be given a type value which is the id of the codelist which represents it. In the key-family-specific namespace, this codelist will be represented by a simpleType declaration which contains a list of enumerations, equivalent to the values of the codelist, as described in the key family. These will be extension of the XSD "string" datatype. The enumerated values will be the values of the codes. The descriptions of the codes will be placed inside XSD "documentation" elements, contained in XSD "annotation" elements, which are themselves contained in the XSD "enumeration" elements as the first instance of the XSD documentation element. No other text shall occur within this particular instance of the XSD documentation element, although other XSD documentation elements may occur within any given XSD enumeration element.

Uncoded attributes will also be represented with XSD simpleType elements declared in the key-family-specific namespace, with names formed by taking the name of the attribute’s concept in the key family and appending “Type” to them. If unrestricted, these will be of the W3C XML Schema primitive type “string”; any restrictions as described in a TextFormat element will be implemented as per the Representations and Datatypes section, above. If any attribute is described in the TextFormat element as having a textType of “Timespan”, then an additional attribute will be added to the extension with a name formed by taking the concept name of the attribute and appending “StartTime” to it. This attribute will have a type of “xs:dateTime”; the primary attribute will be given a type of “xs:duration”.

For each named Group in the key family, a global XSD element will be declared, taking the name of the group. Its XSD type attribute will have a value formed by taking the name of the element and adding “Type” to the end of it. It will have a substitutionGroup attribute which references the Group element declared in the “utility:” namespace.

An XSD complexType will be declared for each named group declared in the key family, with a name formed by taking the name of the group in the key family and appending “Type” to it. It will have an XSD complexContent element which contains an XSD extends with a base attribute value of utility:GroupType. The extends will contain an XSD sequence element, which will contain an XSD element with a reference to the element Series. Its maxOccurs attribute will have a value of “unbounded”. An element named Annotations will be added to the end of the sequence, with a type of AnnotationsType from the “common:” namespace. It will also have a minOccurs value of “0”.

For each attribute in the key family with an attachmentLevel of “Group”, an XSD attribute element may be added to the extends element for any given group. To determine if a declared Group-level attribute in the key family is to be added to a particular named group XSD type, look at the AttachmentGroup elements in the XML of the key family. If the group element in the key-family-specific schema that is being declared appears in an AttachmentGroup element in the key family XML, then the attribute should be included in the utility schema being created. If added, this attribute should be declared as defined for the DataSet level, above. The name will be the name of the attribute’s concept in the key family.

A XSD global element named Series will be declared in the key-family-specific namespace, with a type of SeriesType and a substitutionGroup attribute referencing utility:Series.

An XSD complexType will then be declared with a name of SeriesType. It will have XSD complexContent, with an XSD extension element that has a base attribute value of utility:SeriesType. The extends element will contain an XSD sequence element, which will contain first an XSD element whose ref value is “Key”. This is followed by an XSD element with a ref attribute whose value is “Obs”. Its maxOccurs attribute will have a value of “unbounded”. An element named Annotations will be added to the end of the sequence, with a type of AnnotationsType from the “common:” namespace. It will also have a minOccurs value of “0”.

For each attribute in the key family with an attachmentLevel of “Series”, an XSD attribute element will be added to the extends element, with name, use, and type attributes defined as for the DataSet level, above.

A global XSD element named Key will be declared. It will have a type of KeyType, and a substitutionGroup attribute with a value of utility:Key.

An XSD complexType will be declared, with a name of KeyType. It will have an XSD complexContent element with an XSD extends element inside it, whose base attribute will have a value of “utility:KeyType”. The extends element will contain a XSD sequence of elements, one for each non-time dimension declared in the key family, in the order in which they appear in the XML for the key family. These elements will have names that are the same as the dimension’s concepts in the key family which they represent. Their type attributes will be the names of simpleTypes created exactly as for attributes at the DataSet level, above, with some additional mapping rules. For Time dimensions and non-observational time dimensions, the type will be set to “common:TimePeriodType”. For count dimensions, the type will be set to “xs:integer”. For entity dimensions, the type will be set to “xs:string”. All data

typing with the TextFormat element will be implemented as provided for uncoded attributes, below, and in the general rules regarding this mapping.

An XSD global element will be declared named “Obs”. It will have a substitutionGroup attribute with a value “utility:Obs”. It will have a type of “ObsType”.

An XSD complexType element will be declared with a name “ObsType” and an XSD complexContent. This will contain an XSD extends element with a base attribute of “compact:ObsType”. It will contain an XSD sequence element. The sequence element will contain an element whose name is the name of the TimeDimension concept from the key family, with a type of common:TimePeriodType. It will be followed by an element whose name is the name of the PrimaryMeasure declared in the key family, with a type created as for other attribute and dimension values. If the Primary Measure was described in a TextFormat element as being of textType “Timespan”, another element will be declared with a name of “ObsStartTime”, and it will have a declared type of “xs:dateTime”. The declared type of the primary measure element will be “xs:duration”. Last is an element named Annotations, with a type of AnnotationsType from the “common:” namespace. It will have a minOccurs value of “0”.

For each attribute declared in the key family with an attachmentLevel of “Observation”, an XSD attribute will be added to the extends. Each XSD attribute will take the name of the attribute’s concept declared in the key family, and will have a use attribute, name, and type created as defined as for the DataSet-level attributes described above.

No other declarations or constructs will be added to the schemas created using this mapping.

Note: The UtilityData key-family-specific schema does not have any mechanism for expressing time ranges across a set of observation values. The only permissible message for this schema type is an “update” message containing a complete set of attributes and observation values for the transmitted series. There is no concept of a “delete” message, and the action field in the message Header element is ignored if specified.

6.6 Mappings to Metadata Structure Definition-Specific Metadata Schemas

6.6.1 General Rules

For all metadata-structure-specific schemas SDMX provides a namespace to be used as the extension base: `SDMXMetadatReport.xsd` The metadata-structure-specific schema will be created in its own target name space, owned and maintained by the creating agency. It will use the targetNamespace attribute of the schema element to identify the namespace which contains the metadata-structure-specific schema. The `SDMXMetadatReport.xsd` namespace module provided by SDMX will be incorporated using the import element in the key-family-specific schema. The `SDMXCommon.xsd` namespace module must also be imported into the schema. Other xml:namespace attributes may be added to the schema element as needed.

The elementFormDefault attribute on the schema element will be given a value of "qualified", and the attributeFormDefault attribute on the schema element will be given a value of "unqualified".

6.6.2 Use of W3C Schema Extension in XML Mappings

These rules for Metadata Schemas are identical to those given for Data Schemas above.

6.6.3 Attribute and Observation Values

In many places, the TextFormat element is used in the SDMX Structure message to describe a data type in the schema. This is identical to the Representations and Datatypes section above.

6.6.4 Metadata Report

In the MetadataReport namespace, a global element will be declared with the name MetadataSet. This element declaration will have a substitutiongroup attribute with the value “metadatareport:MetadataSet”, and will have a type of “MetadataSetType”.

A complex type will be declared with the name “MetadataSetType”, and it will contain a complexContent element. Inside of this will be an extension element with a base attribute value of “metadatareport:MetadataSetType”. Inside of this will be a sequence element. For each ReportStructure element in the MetadataStructureDefinition, there will be an element declared which has the name of the id attribute of each report structure. These elements will have type values created by appending the string “Type” to the end of these id values. These elements will have a minOccurs attribute with a value of “0” and a maxOccurs attribute with a value of “unbounded”.

For each ReportStructure element, a complex type is declared with a name value created by appending the string “Type” to the end of the value of its id attribute. Each of these types will contain a sequence element. Inside this sequence element, an XSD element is declared with a name of the id attribute with “Target” appended to it, with a type value named by taking the id value and appending “TargetType” to it. It has no minOccurs or maxOccurs attributes.

For each top-level MetadataAttribute element in the metadata structure definition, there will be an element declaration after the “Target” element declaration. Each report-structure type only has element declarations for the top-level MetadataAttributes which it contains. These elements which correspond to the top-level MetadataAttribute elements will be named after the values of the conceptRef attributes of each one. If the useageStatus attribute has a value of “Conditional,” then the element declaration has a minOccurs attribute with a value of “0”. Each element will have a type value which has a value created by appending the string “Type” to the value of the conceptRef attribute.

For each type created by appending “Type” to the conceptRef attribute value, for each of its child MetadataAttributes an element and type will be declared, following the pattern for the top-level element, recursively. There will be no target types declared, however.

If the representationScheme attribute for any MetadataAttribute is used, then the declaration of that MetadataAttribute's type is changed: the type value will be set to a value created by appending the string "CodeType" to the value of the MetadataAttribute's conceptRef field, and a simpleType declaration which has that name will also be declared. This will contain a restriction element with a base attribute with a value of "xs:NMTOKEN", and the values of the codelist referenced by the MetadataAttributes representationScheme and representationSchemeAgency attributes will each be represented by an enumeration element. The value attribute of each enumeration element will contain the code value, and the code description will be contained in a documentation element inside an annotation element, which will form the contents of the enumeration element.

If a MetadataAttribute contains a TextFormat element, then a simple type is declared as above, but instead of having an enumeration, it is mapped to the schema as per the TextFormat bindings provided above for Key-Family-Specific schemas. If neither the representationScheme attribute not a TextFormat child element is present, then the default representation of the referenced concept should be used, as provided in the ConceptScheme.

For each ReportStructure element, a complex type will be declared which has the name of the ReportStructure id attribute with "TargetType" appended to it. This complex type will contain a sequence element. For each IdentifierComponent or IDentifierComponentRef element present in the FullTargetIdentifier or PartialTargetIdentifier referenced by the ReportStructure in its target attribute, an element will be declared, which will have a name composed of the contents of the corresponding id attribute of the IdentifierComponent (as referenced by the IdentifierComponentRef in the case of PartialTargetIdentifiers) with "Target" appended to it. The type of each such element will have a value which is the name value plus the string "Type". For each of these, a simple type will be declared, with the type name as formulated, which has a value derived from the representation of the concept as provided for MetadataAttributes. Duplicate type declarations are to be avoided. When name collisions occur, they should be resolved by pre-pending the relevant agency code to the name. For those attributes which are described in a TextFormat element as being of textType "Timespan", a second element will be made available with the same cardinality as the first: this will be named by appending "StartTime" to the attribute's concept name. It will be of type "xs:dateTime", and the original value element will be of type "xs:duration".

7 APPENDIX: SAMPLE SDMX-ML DATA MESSAGES

This appendix is presented to provide example layouts for some of the simpler SDMX-ML sample data files, allowing them to be more easily understood. For each sample data file, one or more tables are offered, to show how the data itself might be formatted. Please note that all data is fictitious, and used for demonstration purposes only. (Numbers are not consistent across samples, but are randomly generated.)

7.1 CompactSample.xml

ID: Message JD014 (Untruncated Test Message)
Name: Trans46305

Prepared: 2001-03-11T09:30:47-05:00
Sent by: GB Smith from the BIS, +000.000.0000
To: B.S. Featherstone, Statistics Division, ECB, +000.000.0001

This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks in Millions of US Dollars, Monthly at the beginning of period. (Free data)

Time	Data
2000-01	3.14
2001-02	2.29
2000-03	3.14
2000-04	5.24
2000-05	3.14
2000-06	3.78
2000-07	3.65
2000-08	2.37
2000-09	3.14
2000-10	3.17
2000-11	3.34
2000-12	1.21

External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks in Millions of US Dollars, Annually at the beginning of period. (Free data)

Time	Data
2000-01	3.14

External Debt, All Maturities, Debt Securities Issued Abroad for Mexico, expressed as Stocks in Millions of US Dollars, Monthly at the beginning of period. (Free data)

Time	Data
2000-01	5.14
2001-02	3.29
2000-03	6.14
2000-04	2.24
2000-05	3.14
2000-06	7.78
2000-07	3.65
2000-08	5.37
2000-09	3.14
2000-10	1.17
2000-11	4.34
2000-12	1.21

External Debt, All Maturities, Debt Securities Issued Abroad for Mexico, expressed as Stocks in Millions of US Dollars, Annually at the beginning of period. (Free data)

Time	Data
2000–1	4.14

7.2 UtilitySample.xml

ID: Message JD01678594 (Untruncated Test Message)
Name: Trans46304
Prepared: 2001-03-11T09:30:47-05:00
Sent by: GB Smith from the BIS, +000.000.0000
To: B.S. Featherstone, Statistics Division, ECB, +000.000.0001

This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks in Millions of US Dollars, Monthly at the beginning of period. (Free data)

Time	Data
2000-01	3.14
2001-02	3.19
2000-03	5.26
2000-04	5.12
2000-05	4.13
2000-06	3.12
2000-07	3.14
2000-08	3.79
2000-09	9.79
2000-10	3.14
2000-11	3.19
2000-12	3.14

7.3 GenericSample.xml

ID: Message JD014 (Untruncated Test Message)
Name: Trans46302
Prepared: 2001-03-11T09:30:47-05:00
Sent by: GB Smith from the BIS, +000.000.0000
To: B.S. Featherstone, Statistics Division, ECB, +000.000.0001

This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks in Millions of US Dollars, Monthly at the beginning of period. (Free data)

Time	Data
2000-01	3.14
2001-02	3.14
2000-03	4.29
2000-04	6.04
2000-05	5.18
2000-06	5.07
2000-07	3.13
2000-08	1.17
2000-09	1.14
2000-10	3.04
2000-11	1.14
2000-12	3.24

7.4 CrossSectionalSample.xml

ID: Message BIS947586 (Untruncated Test Message)
Name: Trans46305
Prepared: 2001-03-11T09:30:47-05:00
Sent by: GB Smith from the BIS, +000.000.0000
To: B.S. Featherstone, Statistics Division, ECB, +000.000.0001

This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

External Debt for Mexico, in Millions of US Dollars, at the beginning of period for 2000. (Free data)

Topic	Stocks	Flows
All Maturities, Bank Loans	3.14	1.00
All Maturities, Debt Securities Issued Abroad	6.39	2.27
All Maturities, Brady Bonds	2.34	-1.00
All Maturities, Non-Bank Trade Credits	3.19	-1.06