



**SDMX-ML:
SCHEMA AND DOCUMENTATION**

(VERSION 2.0)

November 2005



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52

© SDMX 2005
<http://www.sdmx.org/>



53 **Contents**

54

55 1 BACKGROUND 5

56 1.1 History and Version 2.0 Developments 5

57 1.2 The XML Design 5

58 1.3 Fostering the Use of a Standard SDMX-ML 6

59 2 NORMATIVE REFERENCES 6

60 3 CONFORMANCE 6

61 4 DESIGN OVERVIEW 6

62 4.1 Scope and Requirements 6

63 4.2 Design Approach 8

64 4.3 SDMX-ML Packaging: Namespace Modules 10

65 5 GENERIC (NON-STRUCTURE-DEFINITION-SPECIFIC) SCHEMAS 12

66 5.1 SDMX Message Namespace Module 12

67 5.2 SDMX Structure Namespace Module 18

68 5.3 SDMX Generic Data Namespace Module 65

69 5.4 SDMX Generic Metadata Namespace Module 69

70 5.5 SDMX Query Namespace Module 73

71 5.6 SDMX Common Namespace Module 82

72 5.7 SDMX Registry Interfaces Namespace Module 86

73 5.8 Data Formatting and Character Encoding 105

74 5.9 Missing Observation Values 105

75 6 KEY-FAMILY- AND METADATA-STRUCTURE-DEFINITION-SPECIFIC SCHEMAS:
76 CORE STRUCTURES & STANDARD MAPPINGS 105

77 6.1 Compact Data Message Core Structure 106

78 6.2 Utility Data Message Core Structure 108

79 6.3 Cross-Sectional Data Message Core Structure 110

80 6.4 Metadata Report Core Structure 111

81 6.5 Mappings to Key-Family-Specific Data Schemas 113



82	6.6	Mappings to Metadata Structure Definition-Specific Metadata Schemas	125
83	7	APPENDIX: SAMPLE SDMX-ML DATA MESSAGES	127
84	7.1	CompactSample.xml	127
85	7.2	UtilitySample.xml	129
86	7.3	GenericSample.xml	129
87	7.4	CrossSectionalSample.xml.....	130
88			



89 1 BACKGROUND

90 1.1 History and Version 2.0 Developments

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

100

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

110

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

115 1.2 The XML Design

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

120

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



- 131 • Key-family-specific schema for cross-sectional data
132 ("SDMXCrossSectionalData.xsd")
- 133 • Generic schema for registry interfaces ("SDMXRegistry.xsd")
- 134 • Generic schema for reference metadata sets ("SDMXRefMetadata.xsd")
- 135 • Metadata-structure-definition-specific schema for metadata sets
136 ("SDMXMetadataReport.xsd")

137

138 **1.3 Fostering the Use of a Standard SDMX-ML**

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

148

149 **2 NORMATIVE REFERENCES**

150 W3C XML Schema Definition Language, version 1.0 (URL:
151 <http://www.w3c.org/XML/Schema#dev>), World Wide Web Consortium
152 W3C Extensible Markup Language, version 1.0, Third Edition (URL:
153 <http://www.w3c.org/TR/2004/REC-xml-20040204/>), World Wide Web Consortium
154

155 **3 CONFORMANCE**

156 Sections V and VI of this document are normative, providing rules for the creation of
157 conformant SDMX-ML XML instances and W3C XML Schemas.
158

159 **4 DESIGN OVERVIEW**

160 **4.1 Scope and Requirements**

161 To understand the relationships between the several document types, it is important
162 to have some familiarity with the requirements they are designed to fulfil.
163

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



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



- 209 • XML formats for describing the relationships between groups of metadata
210 sets and data sets, by mapping concepts and codelists between these
211 structures, and by allowing for common querying of data and metadata
212 described with not only a single structural definition, but with a related set of
213 structural definitions, based on these mappings.
- 214 • Allow for time-related concepts which are not related to the time of the
215 observation to be used in data structures.
- 216 • Allow for simple, un-coded incremental identifiers in data structure definitions,
217 to be used to dis-ambiguate data series/observations which do not have a
218 simple 1-to-1 relationship with the time period of the observation.
- 219 • Allow for un-coded identifiers and descriptors to be associated with data
220 structure definitions which establish an external entity or identifier to
221 disambiguate between otherwise identical series/observations (ie, when a
222 data set describes a group of organisations, or a set of accounts, which might
223 otherwise have identical key values).
- 224 • Allow for non-numeric observation values (usually but not always coded)
- 225 • Allow “cube”-based systems (such as OLAP) to interoperate with less
226 sophisticated systems, without necessarily losing the richness of metadata
227 found in the more sophisticated systems.

228 This is a very broad set of requirements, and in examining these it becomes evident
229 that some of the requirements are very much at cross-purposes. It is almost
230 impossible to design a single XML document type for any single function (exchange
231 of data, exchange of reference metadata, querying, etc.) which will satisfy all of these
232 requirements. At the same time, it was very much felt that whatever design was
233 adopted should have a clear relationship with the information model.
234

235 **4.2 Design Approach**

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

251
252 The main argument against this approach is its apparent complexity, which is a
253 negative factor when launching international standards. In looking at requirements,



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

258

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

269

270 • Database exchange, update, and revision

271 • “Normal” XML use and processing for webmasters, developers, and other
272 users of typical XML tools

273 • Exchange of cross-sectional data (which could potentially be the same as the
274 Database Exchange scenario)

275 • Standard interactions with registry services

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

281

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

291

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

295

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



302 metadata expressed in the key family (eg, ordering of the key). This type of
303 difference in design stems always from the requirements for the specific XML format
304 being designed.
305

306 **4.3 SDMX-ML Packaging: Namespace Modules**

307 In the proposed XML Schema design, there is a packaging scheme based on the
308 idea that XML namespaces can be used as “modules”, so that any given user or
309 application need only be familiar with a subset of the entire library in order to use it.
310 This approach fit very well with the design described above, and is often used in
311 major XML standards for other domains.
312

313 The other major benefit of namespaces – especially in light of the requirement that
314 maintenance agencies be tracked across the potential reuse of the structures and
315 data they maintained – is that it allows SDMX to own certain namespace modules,
316 and allows other maintenance agencies to own namespaces specific to the key-
317 families or metadata structure definitions they also maintain.
318

319 The result is a set of namespace packages which agree with the design approach
320 described above. Each module is a single instance of the W3C XML Schema
321 Language’s schema element, associated with its own XML namespace. Where these
322 modules have dependencies on one another, they use the XML Schema importing
323 mechanism to draw on constructs described in another module.
324

- 325 • An SDMX Namespace Module containing the common message constructs,
326 including the common header information (“SDMXMessage.xsd”) - used with
327 all other SDMX-ML namespace modules

- 328 • An SDMX Namespace Module containing the descriptions of structural
329 metadata such as key families, concepts, and codelists
330 (“SDMXStructure.xsd”)

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

- 336 • An SDMX Namespace Module describing the generic (non-key-family-
337 specific) format for formatting data (“SDMXGenericData.xsd”)

- 338 • An SDMX Namespace Module for describing the structure of the generic
339 query message (“SDMXQuery.xsd”) – for web services developers and
340 users, etc.

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

- 344 • A set of namespaced modules created and maintained by those who create
345 key-family-specific “Compact” schemas – not maintained by SDMX



- 346 • An SDMX Namespace Module providing the common framework to be used
347 for all key-family-specific schemas for webmasters and developers using
348 standard XML tools (“SDMXUtilityData.xsd”) –for processing and
349 publication production use
- 350 • A set of namespaced modules created and maintained by those who create
351 key-family-specific “Utility” schemas – not maintained by SDMX
- 352 • An SDMX Namespace Module providing the common framework to be used
353 for all key-family-specific schemas for cross-sectional data
354 (“SDMXCrossSectionalData.xsd”) – for bilateral use and cross-sectional
355 processing of data
- 356 • A set of namespaced modules created and maintained by those who create
357 key-family-specific “Cross-sectional” schemas – not maintained by SDMX
- 358 • An SDMX Namespace Module providing a generic format for reporting of
359 reference metadata, regardless of metadata structure definition
360 (“SDMXRefMetadata.xsd”).
- 361 • An SDMX Namespace Module providing the common framework to be used
362 for all metadata-structure-specific schemas for reference metadata reporting
363 (“SDMXMetadataReport.xsd”).
- 364 • A set of namespaced modules created and maintained by those who create
365 metadata-structure-specific “Metadata Report” schemas – not maintained by
366 SDMX.
- 367 • An SDMX Namespace Module providing standard interfaces for interactions
368 with a set of registry services (“SDMXRegistry.xsd”).

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

375
376 When namespaces are created by the creators and maintainers of the key-family-
377 specific and metadata-structure-definition-specific types described above, the
378 namespaces must be generated according to a specific format which is predictable.
379 This is identical to the construction of registry URNs, as explained in section 5.2 of
380 the SDMX Registry Interfaces specification, with the addition of a single field at the
381 end of the URN:

- 382
- 383 • For Utility schemas: “:utility”
- 384 • For Compact schemas: “:compact”
- 385 • For Cross-Sectional schemas: “:cross”
- 386 • For Metadata Report schemas: “:metadatareport”
- 387
- 388



389 5 GENERIC (NON-STRUCTURE-DEFINITION- 390 SPECIFIC) SCHEMAS

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

393

394 • `SDMXMessage.xsd`, for generically describing the basic message structure
395 common to all SDMX-ML messages

396 • `SDMXStructure.xsd`, for describing key families, metadata structure
397 definitions, dataflows, metadataflows, codelists, concepts, structure sets,
398 processes, hierarchical codelists, and reporting taxonomies

399 • `SDMXGenericData.xsd`, for describing data across key-families for generic
400 processing

401 • `SDMXQuery.xsd`, for marking-up queries against SDMX-conformat
402 databases and web services

403 • `SDMXCommon.xsd`, describing the common constructs used in other schemas

404 • `SDMXGenericMetadata.xsd`, for generically reporting reference metadata

405 • `SDMXRegistry.xsd`, for all interactions with the SDMX Registry Services

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

411

412 5.1 SDMX Message Namespace Module

413

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

415

416 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/structure
417 (`SDMXStructure.xsd`)

418 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/generic
419 (`SDMXGenericData.xsd`)

420 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/utility
421 (`SDMXUtilityData.xsd`)

422 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/compact
423 (`SDMXCompactData.xsd`)

424 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross
425 (`SDMXCrossSectionalData.xsd`)

426 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/query (`SDMXQuery.xsd`)

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

429 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/registry
430 (`SDMXRegistry.xsd`)



431 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/genericmetadata
432 (SDMXGenericMetadata.xsd)
433 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/metadataareport
434 (SDMXMetadataReport.xsd)

435

436 **5.1.1 Global Elements**

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

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

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

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

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

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

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

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

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

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

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

484

485 5.1.2 Complex Types

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

489 *Element Content (Type):*

490
 491 Header (HeaderType)

492 **StructureType:** StructureType defines the contents of a structure message.

493
 494 *Extends:* MessageType

495
 496 *Element Content (Type):*

497
 498 OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
 499 Dataflows (structure:DataflowsType) - min. 0
 500 Metadataflows (structure:MetadataflowsType) - min. 0
 501 CategorySchemes (structure:CategorySchemesType) - min. 0
 502 CodeLists (structure:CodeListsType) - min. 0
 503 HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
 504 Concepts (structure:ConceptsType) - min. 0
 505 MetadataStructureDefinitions
 506 (structure:MetadataStructureDefinitionsType) - min. 0
 507 KeyFamilies (structure:KeyFamiliesType) - min. 0
 508 StructureSets (structure:StructureSetsType) - min. 0



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

511 **GenericDataType:** GenericDataType defines the contents of a GenericData
512 message.

513
514 *Extends:* MessageType
515
516 *Element Content (Type):*

517
518 DataSet (generic:DataSetType)

519 **UtilityDataType:** UtilityDataType defines the contents of a UtilityData
520 message.

521
522 *Extends:* MessageType
523
524 *Element Content (Type):*

525
526 [Reference] (utility:DataSet)

527 **CompactDataType:** CompactDataType defines the contents of a
528 CompactData message.

529
530 *Extends:* MessageType
531
532 *Element Content (Type):*

533
534 [Reference] (compact:DataSet)

535 **CrossSectionalDataType:** CrossSectionalDataType defines the contents of a
536 CrossSectionalData message.

537
538 *Extends:* MessageType
539
540 *Element Content (Type):*

541
542 [Reference] (cross:DataSet)

543 **GenericMetadataType:** GenericMetadataType defines the contents of a
544 Generic Metadata message.

545
546 *Extends:* MessageType



547
548

Element Content (Type):

549
550

[Reference] (genericmetadata:MetadataSet)

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

553
554
555
556

Extends: MessageType

Element Content (Type):

557
558

[Reference] (metadatareport:MetadataSet)

559 **QueryMessageType:** QueryMessageType defines the contents of a
560 QueryMessage.

561
562
563
564

Extends: MessageType

Element Content (Type):

565
566

Query (query:QueryType)

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

569
570

Extends: MessageType

571 **MessageGroupType:** MessageGroupType defines the contents of a
572 MessageGroup message.

573
574

Extends: MessageType

575

Attribute: id(xs:NMTOKEN) - optional

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



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

599 *Element Content (Type):*

600
601 ID (common:IDType)
602 Test (xs:boolean)
603 Truncated (xs:boolean) - min. 0
604 Name (common:TextType) - min. 0 - max. unbounded
605 Prepared (HeaderTimeType)
606 Sender (PartyType) - max. unbounded
607 Receiver (PartyType) - min. 0 - max. unbounded
608 KeyFamilyRef (xs:NMTOKEN) - min. 0
609 KeyFamilyAgency (xs:NMTOKEN) - min. 0
610 DataSetAgency (xs:NMTOKEN) - min. 0
611 DataSetID (xs:NMTOKEN) - min. 0
612 DataSetAction (common:ActionType) - min. 0
613 Extracted (xs:dateTime) - min. 0
614 ReportingBegin (HeaderTimeType) - min. 0
615 ReportingEnd (HeaderTimeType) - min. 0
616 Source (common:TextType) - min. 0 - max. unbounded

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

620 *Element Content (Type):*

621
622 Name (common:TextType) - min. 0 - max. unbounded
623 Contact (ContactType) - min. 0 - max. unbounded

624 *Attribute:* id (xs:NMTOKEN) - required

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

627 *Element Content (Type):*

628
629 Name (common:TextType) - min. 0 - max. unbounded
630 Department (common:TextType) - min. 0 - max. unbounded
631 Role (common:TextType) - min. 0 - max. unbounded
632 Telephone (xs:string) [Choice]
633 Fax (xs:string) [Choice]
634 X400 (xs:string) [Choice]
635 URI (xs:anyURI) [Choice]
636 Email (xs:string) [Choice]

637

638 5.1.3 Simple Types

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

641

642

643 5.2 SDMX Structure Namespace Module

644

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

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

648

649 5.2.1 Complex Types

650 **OrganisationSchemesType:** OrganisationSchemesType contains one or
651 more OrganisationSchemes.

652 *Element Content (Type):*

653

654 OrganisationScheme (OrganisationSchemeType) - max. unbounded

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



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

676 *Element Content (Type):*

677
678 Name (common:TextType) - max. unbounded
679 Description (common:TextType) - min. 0 - max. unbounded
680 Agencies (AgenciesType) - min. 0 - max. unbounded
681 DataProviders (DataProvidersType) - min. 0 - max. unbounded
682 DataConsumers (DataConsumersType) - min. 0 - max. unbounded
683 Annotations (common:AnnotationsType) - min. 0

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

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

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

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

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

689 *Attribute:* agencyID (common:IDType) - required

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

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

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

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

696 *Element Content (Type):*

697
698 DataProvider (OrganisationType) - max. unbounded

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



702 *Element Content (Type):*

703
704 DataConsumer (OrganisationType) - max. unbounded

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

709 *Element Content (Type):*

710
711 Agency (OrganisationType) - max. unbounded

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

739 *Element Content (Type):*

740
741 Name (common:TextType) - max. unbounded
742 Description (common:TextType) - min. 0 - max. unbounded
743 MaintenanceContact (ContactType) - min. 0
744 CollectorContact (ContactType) - min. 0

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

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

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

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

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

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

754 *Attribute:* parentOrganisation (common:IDType) - optional

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

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

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

760 *Element Content (Type):*

761 Name (common:TextType) - min. 0 - max. unbounded
 762 id (common:IDType) - min. 0
 763 Department (common:TextType) - min. 0 - max. unbounded
 764 Role (common:TextType) - min. 0 - max. unbounded
 765 Telephone (xs:string) [Choice]
 766 Fax (xs:string) [Choice]
 767 X400 (xs:string) [Choice]
 768 URI (xs:anyURI) [Choice]
 769 Email (xs:string) [Choice]
 770

771 **DataflowsType:** DataflowsType contains one or more data flows.

772 *Element Content (Type):*

773 Dataflow (DataflowType) - max. unbounded
 774

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

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

792 *Element Content (Type):*

793
794 Name (common:TextType) - max. unbounded
795 Description (common:TextType) - min. 0 - max. unbounded
796 KeyFamilyRef (KeyFamilyRefType) - min. 0
797 CategoryRef (CategoryRefType) - min. 0 - max. unbounded
798 Annotations (common:AnnotationsType) - min. 0

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

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

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

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

803 *Attribute:* agencyID (common:IDType) - required

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

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

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

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

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

814 *Element Content (Type):*



815
816 URN (xs:anyURI) - min. 0
817 KeyFamilyID (common:IDType) - min. 0
818 KeyFamilyAgencyID (common:IDType) - min. 0
819 Version (xs:string) - min. 0

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

824 *Element Content (Type):*

825
826 URN (xs:anyURI) - min. 0
827 CategorySchemeID (common:IDType) - min. 0
828 CategorySchemeAgencyID (common:IDType) - min. 0
829 CategorySchemeVersion (xs:string) - min. 0
830 CategoryID (CategoryIDType) - min. 0

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

839 *Element Content (Type):*

840
841 ID (common:IDType)
842 CategoryVersion (xs:string) - min. 0
843 CategoryID (CategoryIDType) - min. 0

844 **MetadataflowsType:** MetadataflowsType contains one or more metadata
845 flows.

846 *Element Content (Type):*

847
848 Metadataflow (MetadataflowType) - max. unbounded

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



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

868 *Element Content (Type):*

869
870 Name (common:TextType) - max. unbounded
871 Description (common:TextType) - min. 0 - max. unbounded
872 MetadataStructureRef (MetadataStructureRefType) - min. 0
873 CategoryRef (CategoryRefType) - min. 0 - max. unbounded
874 Annotations (common:AnnotationsType) - min. 0

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

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

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

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

879 *Attribute:* agencyID (common:IDType) - required

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

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

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

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

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

888 *Element Content (Type):*

889
890 URN (xs:anyURI) - min. 0
891 MetadataStructureID (common:IDType) - min. 0



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

894 **CategorySchemesType:** CategorySchemesType contains one or more
895 category schemes.

896 *Element Content (Type):*

897
898 CategoryScheme (CategorySchemeType) - max. unbounded

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

918 *Element Content (Type):*

919
920 Name (common:TextType) - max. unbounded
921 Description (common:TextType) - min. 0 - max. unbounded
922 Category (CategoryType) - min. 0 - max. unbounded
923 Annotations (common:AnnotationsType) - min. 0

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

925 *Attribute:* agencyID (common:IDType) - required

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

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

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

929 *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, language-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 omitted, 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



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

971 *Element Content (Type):*

972
973 Name (common:TextType) - max. unbounded
974 Description (common:TextType) - min. 0 - max. unbounded
975 Code (CodeType) - min. 0 - max. unbounded
976 Annotations (common:AnnotationsType) - min. 0

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

978 *Attribute:* agencyID (common:IDType) - required

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

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

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

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

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

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

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

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

994 *Element Content (Type):*

995
996 Description (common:TextType) - max. unbounded
997 Annotations (common:AnnotationsType) - min. 0

998 *Attribute:* value (common:IDType) - required



- 999 *Attribute:* urn (xs:anyURI) - optional
- 1000 *Attribute:* parentCode (common:IDType) - optional
- 1001 **HierarchicalCodelistsType:** HierarchicalCodelistsType contains one or more
1002 sets of structural information about the hierarchies within a codelist
1003 (hierarchical codelists). This corresponds to complex hierarchical codelists
1004 within the SDMX Information Model - very simple hierarchies can be
1005 described within the regular Codelist, using the parentCode attribute.
- 1006 *Element Content (Type):*
- 1007
- 1008 HierarchicalCodelist (HierarchicalCodelistType) - max. unbounded
- 1009 **HierarchicalCodelistType:** A hierarchical codelist references a Codelist, and
1010 supplies the extra structural metadata to assemble the codes into a hierarchy.
1011 A human-readable name must be supplied, and multiple language-specific
1012 variants may be provided. A longer human-readable description may be
1013 provided, and may also be presented as a set of language-specific variants.
1014 The CodelistRef element references a codelist, and may indicate more than
1015 one. Annotations may be provided in the Annotations element. An ID unique
1016 for the agency specified in the agency attribute must be assigned, using the id
1017 attribute. A version may be provided using the version attribute - if no value is
1018 provided, it is assumed to be "1.0". A valid SDMX Registry URN may be
1019 provided in the urn attribute, as specified in the SDMX Registry Specification.
1020 If the isExternalReference attribute has a value of true, the uri attribute must
1021 specify the location of a valid SDMX Structure Message which provides the
1022 full details of the hierarchical codelist; otherwise, all details must be present.
1023 The validFrom and validTo attributes provide inclusive dates for providing
1024 supplemental validity information about the version.
- 1025 *Element Content (Type):*
- 1026
- 1027 Name (common:TextType) - max. unbounded
- 1028 Description (common:TextType) - min. 0 - max. unbounded
- 1029 CodelistRef (CodelistRefType) - min. 0 - max. unbounded
- 1030 Hierarchy (HierarchyType) - min. 0 - max. unbounded
- 1031 Annotations (common:AnnotationsType) - min. 0
- 1032 *Attribute:* id (common:IDType) - required
- 1033 *Attribute:* agencyID (common:IDType) - required
- 1034 *Attribute:* version (xs:string) - optional
- 1035 *Attribute:* urn (xs:anyURI) - optional

1036 *Attribute:* uri (xs:anyURI) - optional

1037 *Attribute:* isExternalReference (xs:boolean) - optional

1038 *Attribute:* isFinal (xs:boolean) - optional

1039 *Attribute:* validFrom (common:TimePeriodType) - optional

1040 *Attribute:* validTo (common:TimePeriodType) - optional

1041 **HierarchyType:** The recursive CodeRef element is used to assemble the
 1042 codes in the codelist(s) referenced by the parent hierarchical codelist into a
 1043 hierarchy. The Level element is used to describe the levels of a levelled
 1044 hierarchy, which may be referenced from each of the CodeRefs in the
 1045 Hierarchy. A human-readable name must be assigned, which may be
 1046 provided in multiple, parallel-language versions. A longer, human-readable
 1047 Description may also be provided, which can also have multiple parallel-
 1048 language versions. Annotations may be provided with the Annotations
 1049 element. The id attribute provides a unique id for the hierarchy. The urn
 1050 attribute can be used to specify the hierarchy with a valid SDMX Registry
 1051 URN (see the SDMX Registry Specification). The version attribute specifies a
 1052 version (understood to be "1.0" if not specified), and isFinal, once given a
 1053 value of true, indicates that nothing may be changed without also changing
 1054 the version number. validFrom and validTo are inclusive dates indicating the
 1055 relevant period of the hierarchy.

1056 *Element Content (Type):*

1057 Name (common:TextType) - max. unbounded
 1058 Description (common:TextType) - min. 0 - max. unbounded
 1059 CodeRef (CodeRefType) - min. 0 - max. unbounded
 1060 Level (LevelType) - min. 0 - max. unbounded
 1061 Annotations (common:AnnotationsType) - min. 0

1063 *Attribute:* id (common:IDType) - required

1064 *Attribute:* urn (xs:anyURI) - optional

1065 *Attribute:* version (xs:string) - optional

1066 *Attribute:* isFinal (xs:boolean) - optional

1067 *Attribute:* validFrom (common:TimePeriodType) - optional

1068 *Attribute:* validTo (common:TimePeriodType) - optional

1069 **LevelType:** LevelType describes a level in a hierarchical codelist. The Order
 1070 element specifies where the level is in a levelled hierarchy, starting with the

1071 value "1" for the top level, and going sequentially from there using whole
 1072 integers. CodingType specifies the text formatting of the codes at that level. A
 1073 human-readable name must be assigned, which may be provided in multiple,
 1074 parallel-language versions. A longer, human-readable Description may also
 1075 be provided, which can also have multiple parallel-language versions.
 1076 Annotations may be provided with the Annotations element. The id attribute
 1077 provides a unique id for the hierarchy. The urn attribute can be used to specify
 1078 the hierarchy with a valid SDMX Registry URN (see the SDMX Registry
 1079 Specification).

1080 *Element Content (Type):*

1081
 1082 Name (common:TextType) - max. unbounded
 1083 Description (common:TextType) - min. 0 - max. unbounded
 1084 Order (xs:integer)
 1085 CodingType (TextFormatType) - min. 0
 1086 Annotations (common:AnnotationsType) - min. 0

1087 *Attribute:* id (common:IDType) - required

1088 *Attribute:* urn (xs:anyURI) - optional

1089 **CodelistRefType:** The CodelistRefType provides the structure for a codelist
 1090 reference. (Note that this is structured differently than a similarly-named type
 1091 in the Registry namespace.) At a minimum, either: AgencyID has the ID of an
 1092 agency as a value; CodelistID takes the ID of a codelist maintained by that
 1093 agency; and Version specifies the version of the codelist; or URN supplies a
 1094 valid SDMX Registry URN (see the SDMX Registry Specification). Alias is
 1095 used to carry the identifier for the referenced codelist, so that codes from that
 1096 list can be easily referenced by the CodeRefs contained in the parent
 1097 Hierarchy, without having to repeat the agency and version for each
 1098 reference. The Alias must be unique within the parent Hierarchical Codelist.

1099 *Element Content (Type):*

1100
 1101 URN (xs:anyURI) - min. 0
 1102 AgencyID (common:IDType) - min. 0
 1103 CodelistID (common:IDType) - min. 0
 1104 Version (xs:string) - min. 0
 1105 Alias (common:IDType) - min. 0

1106 **CodeRefType:** The CodeRefType provides the structure for a codelist
 1107 reference. At a minimum, either a URN value (a valid SDMX Registry URN as
 1108 specified in the SDMX Registry Specification) must be supplied, or a
 1109 CodelistAliasRef and a CodeID must be specified. CodelistAliasRef
 1110 references an alias assigned in a CodelistRef element in the containing
 1111 hierarchical codelist. CodeRef references a code from the codelist identified at
 1112 the level of the parent hierarchical codelist. Codes are arranged in a hierarchy



1113 by reference. Note that it is possible to reference a single code such that it
1114 has multiple parents within the hierarchy. Further, the hierarchy may or may
1115 not be a levelled one. CodeID holds the ID of the code in the codelist
1116 referenced by the hierarchical codelist. CodeRef references a code. LevelRef
1117 holds the id of a Level described in the same parent Hierarchical Codelist.
1118 NodeAliasID allows for an ID to be assigned to the use of the particular code
1119 at that specific point in the hierarchy. This value is unique within the hierarchy
1120 being created, and is used to map the hierarchy against external structures.
1121 Version holds the version number of the referenced code, to support
1122 management of complex hierarchies. Along with this field are the ValidFrom
1123 and ValidTo dates, which are inclusive dates during which the code is valid
1124 within the parent hierarchy.

1125 *Element Content (Type):*

1126
1127 URN (xs:anyURI) - min. 0
1128 CodelistAliasRef (common:IDType) - min. 0
1129 CodeID (common:IDType) - min. 0
1130 CodeRef (CodeRefType) - min. 0 - max. unbounded
1131 LevelRef (common:IDType) - min. 0
1132 NodeAliasID (xs:string) - min. 0
1133 Version (xs:string) - min. 0
1134 ValidFrom (common:TimePeriodType) - min. 0
1135 ValidTo (common:TimePeriodType) - min. 0

1136 **ConceptsType:** The ConceptsType describes an XML type which contains
1137 information about sets of concepts and their relationships, each of which is
1138 described in a ConceptScheme element. This section replaces the section of
1139 the version 1.0 SDMXStructure message which provides details about
1140 concepts. As such, it is backward-compatible, and may be used to contain a
1141 simple list of concepts as per the 1.0 SDMX-ML specification.

1142 *Element Content (Type):*

1143
1144 Concept (ConceptType) - min. 0 - max. unbounded
1145 ConceptScheme (ConceptSchemeType) - min. 0 - max. unbounded
1146 Annotations (common:AnnotationsType) - min. 0

1147 **ConceptType:** ConceptType specifies the information provided for a single
1148 concept. This includes a name, as element content, and an ID. It is possible to
1149 use the uri field to point to the location of an SDMX-ML Structure message
1150 which contains a more detailed version of the concept. (This is termed an
1151 "external reference".) If an external reference is being made, the
1152 isExternalReference attribute must be set to "true". In this case, all details of
1153 the concept are assumed to be found externally, and inline characteristics
1154 provided through child elements and the coreRepresentation and
1155 coreRepresentationAgency attributes are to be ignored. The
1156 coreRepresentation and coreRepresentationAgency attributes can identify a
1157 codelist which is a default representation of the concept. Uncoded default

1158 representations (or information about the textual aspects of coded default
1159 representations) can be provided with the TextFormat child element of the
1160 concept. Semantic relationships between concepts which occur within a single
1161 concept scheme can be captured with the parent and parentAgency attributes
1162 - these identify the concept of which the current concept is a qualification (in
1163 the ISO 11179 sense) or subclass. When used outside of a containing
1164 ConceptScheme, these attributes may be ignored. If a coreRepresentation
1165 and coreRepresentationAgency are not provided, but are provided in the
1166 indicated parent, then the default representation is inherited from the specified
1167 parent concept. Note that all concepts within a concept scheme must be
1168 uniquely identified by their id - each concept scheme has only one agency for
1169 all its concepts. The agency attribute here is provided for backward-
1170 compatibility with version 1.0 of the standards, and it must not be used for
1171 concepts which are child elements of a concept scheme.

1172 *Element Content (Type):*

1173
1174 Name (common:TextType) - max. unbounded
1175 Description (common:TextType) - min. 0 - max. unbounded
1176 TextFormat (TextFormatType) - min. 0
1177 Annotations (common:AnnotationsType) - min. 0

1178 *Attribute:* id (common:IDType) - required

1179 *Attribute:* agencyID (common:IDType) - optional

1180 *Attribute:* version (xs:string) - optional

1181 *Attribute:* uri (xs:anyURI) - optional

1182 *Attribute:* urn (xs:anyURI) - optional

1183 *Attribute:* isExternalReference (xs:boolean) - optional

1184 *Attribute:* coreRepresentation (common:IDType) - optional

1185 *Attribute:* coreRepresentationAgency (common:IDType) -
1186 optional

1187 *Attribute:* parent (common:IDType) - optional

1188 *Attribute:* parentAgency (common:IDType) - optional

1189 **ConceptSchemeType:** ConceptSchemeType describes the structure of a
1190 ConceptScheme element, which is the preferred form (as of version 2.0) of
1191 presenting the concepts used in other SDMX constructs. ConceptSchemes
1192 may be included inline (that is, with all details provided in the instance or may
1193 be referenced externally. It is possible to use the uri field to point to the



1194 location of an SDMX-ML Structure message which contains a more detailed
1195 version of the concept. (This is termed an "external reference".) If an external
1196 reference is being made, the isExternalReference attribute must be set to
1197 "true". A Name may be provided as a child element (in multiple parallel
1198 language versions) and an ID and version and agency information may be
1199 provided. ConceptSchemes represent a collection of concepts which are used
1200 to describe a meaningful set of distinct concepts, to be used in the reporting of
1201 data or metadata. The validFrom and validTo attributes provide inclusive
1202 dates for providing supplemental validity information about the version.

1203 *Element Content (Type):*

1204

1205 Name (common:TextType) - max. unbounded
1206 Description (common:TextType) - min. 0 - max. unbounded
1207 Concept (ConceptType) - min. 0 - max. unbounded
1208 Annotations (common:AnnotationsType) - min. 0

1209 *Attribute:* id (common:IDType) - required

1210 *Attribute:* agencyID (common:IDType) - required

1211 *Attribute:* version (xs:string) - optional

1212 *Attribute:* uri (xs:anyURI) - optional

1213 *Attribute:* urn (xs:anyURI) - optional

1214 *Attribute:* isExternalReference (xs:boolean) - optional

1215 *Attribute:* isFinal (xs:boolean) - optional

1216 *Attribute:* validFrom (common:TimePeriodType) - optional

1217 *Attribute:* validTo (common:TimePeriodType) - optional

1218 **MetadataStructureDefinitionsType:** MetadataStructureDefinitionsType
1219 describes one or more metadata structure definitions.

1220 *Element Content (Type):*

1221

1222 MetadataStructureDefinition (MetadataStructureDefinitionType) - max.
1223 unbounded

1224 **MetadataStructureDefinitionType:** A metadata structure definition performs
1225 several functions: it groups sets of objects into "targets" against which
1226 reference metadata may be reported. Targets define the structure of the
1227 reference metadata "keys" which identify specific types of reported metadata,

1228 and describe the valid values for populating the keys. Also, metadata structure
 1229 definitions provide a presentational organization of concepts for reporting
 1230 purposes. The structure of a reference metadata report is derived from this
 1231 presentational structure. Also, representations - unless defaults from the
 1232 concepts are used - must be indicated as part of this presentational structure.
 1233 Attributes allow the assignment of an ID, an agency, a version, and a uri. It is
 1234 possible to use the uri field to point to the location of an SDMX-ML Structure
 1235 message which contains a more detailed version of the metadata structure
 1236 definition. (This is termed an "external reference".) If an external reference is
 1237 being made, the isExternalReference attribute must be set to "true". When an
 1238 external reference is being made, none of the child elements should be
 1239 included. Otherwise, both TargetIdentifiers and at least one ReportStructure
 1240 must be included. The urn attribute holds a valid SDMX registry URN (see the
 1241 SDMX Registry Specification). The validFrom and validTo attributes provide
 1242 inclusive dates for providing supplemental validity information about the
 1243 version.

1244 *Element Content (Type):*

1245 Name (common:TextType) - max. unbounded
 1246 Description (common:TextType) - min. 0 - max. unbounded
 1247 TargetIdentifiers (TargetIdentifiersType) - min. 0
 1248 ReportStructure (ReportStructureType) - min. 0 - max. unbounded
 1249 Annotations (common:AnnotationsType) - min. 0

1251 *Attribute:* id (common:IDType) - required

1252 *Attribute:* agencyID (common:IDType) - required

1253 *Attribute:* version (xs:string) - optional

1254 *Attribute:* urn (xs:anyURI) - optional

1255 *Attribute:* uri (xs:anyURI) - optional

1256 *Attribute:* isExternalReference (xs:boolean) - optional

1257 *Attribute:* isFinal (xs:boolean) - optional

1258 *Attribute:* validFrom (common:TimePeriodType) - optional

1259 *Attribute:* validTo (common:TimePeriodType) - optional

1260 **TargetIdentifiersType:** TargetIdentifiers are the set of objects against which
 1261 reference metadata is reported (data providers, data flows, data or metadata
 1262 structures, etc.). There are two types of TargetIdentifiers: the "full" target
 1263 identifier, which lists every object used to attach metadata to in the metadata
 1264 structure definition, and the partial target identifiers, which indicate sub-sets of



1265 those concepts against which reference metadata may be reported. It is
1266 sometimes the case that metadata will also be reported against the full target
1267 identifier. An example of this is as follows: we might wish to collect some
1268 metadata concepts such as contact information for some of the objects
1269 described by the SDMX Information Model - for each instance of a metadata
1270 flow or a data provider, for example. Our concepts would be the concepts
1271 associated with contact information: CONTACT_NAME,
1272 CONTACT_PHONE_NUMBER, etc. We would determine how these concepts
1273 are associated with the objects in the model: do we want a contact for each
1274 data provider broken out by data flow? Or do we want only a single contact
1275 reported for each data provider (who might provide several data flows)? Each
1276 object or combination of objects we need to have metadata reported for
1277 becomes a partial target identifier, unless it happens to contain the full target
1278 identifier, in which case we use that instead. Thus, our valid partial target
1279 identifiers here would be "data flow" and "data provider", because "data flow
1280 by data provider" is a full target identifier. For each target identifier, we could
1281 have some set of our concepts reported.

1282 *Element Content (Type):*

1283
1284 FullTargetIdentifier (FullTargetIdentifierType)
1285 PartialTargetIdentifier (PartialTargetIdentifierType) - min. 0 - max. unbounded
1286 Annotations (common:AnnotationsType) - min. 0

1287 **FullTargetIdentifierType:** The full target identifier provides details on all of
1288 the objects against which metadata can be reported. The full target identifier is
1289 made up of a set of identifier components - each getting its own child element
1290 - which are similar to the dimensions of a key family: each one indicates that a
1291 value will be provided by the metadata reporter to identify and describe the
1292 metadata being reported. A human-readable name must be provided, which
1293 may be provided in multiple, parallel-language versions. A longer, human-
1294 readable name may also be provided in multiple, language-parallel versions.
1295 Annotations may be provided.

1296 *Element Content (Type):*

1297
1298 Name (common:TextType) - max. unbounded
1299 Description (common:TextType) - min. 0 - max. unbounded
1300 IdentifierComponent (IdentifierComponentType) - min. 0 - max. unbounded
1301 Annotations (common:AnnotationsType) - min. 0

1302 *Attribute:* id (common:IDType) - required

1303 *Attribute:* urn (xs:anyURI) - optional

1304 *Attribute:* uri (xs:anyURI) - optional



1305 **IdentifierComponentType:** An identifier component describes the use of an
1306 object within the full target identifier set. An identifier component must be one
1307 of the formally-recognized object classes found in the SDMX Information
1308 Model: the sub-element TargetObjectClass provides a way of indicating which
1309 objects will be used in reporting metadata, and will indicate how those objects
1310 are to be identified by the metadata reporters (which value sets will be allowed
1311 for each identification field for each object). The RepresentationScheme child
1312 element is used to indicate the valid range of values for the providing target
1313 identifiers in reported metadata.

1314 *Element Content (Type):*

1315
1316 Name (common:TextType) - max. unbounded
1317 Description (common:TextType) - min. 0 - max. unbounded
1318 TargetObjectClass (ObjectIDType)
1319 RepresentationScheme (RepresentationSchemeType) - min. 0
1320 Annotations (common:AnnotationsType) - min. 0

1321 *Attribute:* id (common:IDType) - required

1322 *Attribute:* urn (xs:anyURI) - optional

1323 *Attribute:* uri (xs:anyURI) - optional

1324 **PartialTargetIdentifierType:** Partial target identifiers are subsets of the full
1325 target identifier. They are themselves given an identifier, so that they can be
1326 referenced by the metadata attributes of a report. A human-readable name
1327 must be provided, which may be provided in multiple, parallel-language
1328 versions. A longer, human-readable name may also be provided in multiple,
1329 language-parallel versions. Annotations may be provided.

1330 *Element Content (Type):*

1331
1332 Name (common:TextType) - max. unbounded
1333 Description (common:TextType) - min. 0 - max. unbounded
1334 IdentifierComponentRef (common:IDType) - min. 0 - max. unbounded
1335 Annotations (common:AnnotationsType) - min. 0

1336 *Attribute:* id (common:IDType) - required

1337 *Attribute:* urn (xs:anyURI) - optional

1338 *Attribute:* uri (xs:anyURI) - optional

1339 **RepresentationSchemeType:** Representation schemes indicated which
1340 values are valid for identifying objects within each class. For any given
1341 representation scheme, two IDs must be provided: the
1342 RepresentationScheme must have an ID as assigned to it by it



1343 representationSchemeAgency, whose ID must also be provided. The type of
1344 the representation scheme is expressed in the representationSchemeType
1345 attribute.

1346 *Attribute:* representationScheme (common:IDType) - required

1347 *Attribute:* representationSchemeAgency (common:IDType) -
1348 required

1349 *Attribute:* representationSchemeType
1350 (RepresentationSchemeTypeType) - required

1351 **ReportStructureType:** The report structure describes the presentation of the
1352 reported concepts, and associates them with target identifiers, full or partial. It
1353 can be given a name and/or annotations. It must be given an ID, using the id
1354 attribute, which must be unique within the MetadataStructureDefinition
1355 element. It contains one or more MetadataAttribute elements, each of which
1356 may either hold a value, or may have subordinate MetadataAttribute
1357 elements. The target attribute holds the ID of a full or partial identifier, which is
1358 the identifier of the target against which the metadata attributes are reported.

1359 *Element Content (Type):*

1360

1361 Name (common:TextType) - max. unbounded

1362 Description (common:TextType) - min. 0 - max. unbounded

1363 MetadataAttribute (MetadataAttributeType) - max. unbounded

1364 Annotations (common:AnnotationsType) - min. 0

1365 *Attribute:* id (common:IDType) - required

1366 *Attribute:* urn (xs:anyURI) - optional

1367 *Attribute:* uri (xs:anyURI) - optional

1368 *Attribute:* target (common:IDType) - required

1369 **MetadataAttributeType:** Metadata attributes are those concepts - whether
1370 taking a coded or uncoded value, or made up of child concepts, or both -
1371 which are reported against a full or partial target identifier. If there are nested
1372 metadata attributes, these concepts are subordinate to the parent metadata
1373 attribute - that is, for the purposes of presentation, the parent concept is made
1374 up of the child concepts. This hierarchy is strictly presentational, for the
1375 purposes of structuring reports. If the metadata attribute can have a coded or
1376 uncoded value, then the characteristics of the value are indicated with the
1377 TextFormat child element. If the value is coded, then the
1378 representationScheme and representationSchemeAgency attributes must
1379 hold values: the representationScheme attribute takes the ID of a
1380 representation scheme, and the representationSchemeAgency takes the ID of



1381 the agency which maintains that scheme. The conceptRef attribute holds the
1382 ID of the metadata attribute's concept. The conceptAgency attribute takes the
1383 agency ID of the concept referenced in conceptRef. The conceptSchemeRef
1384 attribute holds the ID value of the concept scheme from which the concept is
1385 taken, and the conceptSchemeAgency holds the ID of the agency that
1386 maintains the concept scheme referenced in the conceptSchemeRef attribute.
1387 The usageStatus attribute indicates whether provision of the metadata
1388 attribute is conditional or mandatory.

1389 *Element Content (Type):*

1390

1391 MetadataAttribute (MetadataAttributeType) - min. 0 - max. unbounded

1392 TextFormat (TextFormatType) - min. 0

1393 Annotations (common:AnnotationsType) - min. 0

1394 *Attribute:* conceptRef (common:IDType) - required

1395 *Attribute:* conceptVersion (xs:string) - optional

1396 *Attribute:* conceptAgency (common:IDType) - optional

1397 *Attribute:* conceptSchemeRef (common:IDType) - optional

1398 *Attribute:* conceptSchemeAgency (common:IDType) - optional

1399 *Attribute:* representationScheme (common:IDType) - optional

1400 *Attribute:* representationSchemeAgency (common:IDType) -

1401 optional

1402 *Attribute:* usageStatus (UsageStatusType) - required

1403 **TextFormatType:** TextFormatType defines the information for describing a
1404 text format. If the TextType attribute is not specified, any valid characters may
1405 be included in the text field. (It corresponds to the xs:string datatype of W3C
1406 XML Schema.) The textType attribute provides a description of the data type,
1407 and may place restrictions on the values of the other attributes, referred to as
1408 "facets". The isSequence attribute indicates whether the values are intended
1409 to be ordered, and it may work in combination with the interval attribute. The
1410 minLength and maxLength attributes specify the minimum and maximum
1411 lengths of the value in characters. startValue and endValue are used for
1412 inclusive and exclusive ranges, indicating what the bounds of the range are.
1413 The interval attribute specifies the permitted interval between two values. The
1414 timeInterval attribute indicates the permitted duration between two time
1415 expressions. The decimals attribute indicates the number of characters
1416 allowed after the decimal separator. The pattern attribute holds any regular
1417 expression permitted in the simila facet in W3C XML Schema.

1418 *Attribute:* textType (TextTypeType) - optional

1419 *Attribute:* isSequence (xs:boolean) - optional

1420 *Attribute:* minLength (xs:integer) - optional

1421 *Attribute:* maxLength (xs:integer) - optional

1422 *Attribute:* startValue (xs:double) - optional

1423 *Attribute:* endValue (xs:double) - optional

1424 *Attribute:* interval (xs:double) - optional

1425 *Attribute:* timeInterval (xs:duration) - optional

1426 *Attribute:* decimals (xs:integer) - optional

1427 *Attribute:* pattern (xs:string) - optional

1428 **KeyFamiliesType:** KeyFamiliesType defines the structure for describing one
1429 or more key families. It also provides uniqueness constraints for each of the
1430 key family IDs.

1431 *Element Content (Type):*

1432

1433 KeyFamily (KeyFamilyType) - max. unbounded

1434 **KeyFamilyType:** KeyFamilyType defines the structure of a key-family
1435 description. This includes the name and a set of components (attributes and
1436 dimensions) as element content, and an ID, agency, version, and the URL
1437 where located as attributes. The urn attribute holds a valid SDMX Registry
1438 URN, as per the SDMX Registry spec. The isFinal attribute, once set to true,
1439 indicates that no changes may be made without versioning. The validFrom
1440 and validTo attributes provide inclusive dates for providing supplemental
1441 validity information about the version. If the isExternalReference attribute is
1442 true, then the uri attribute must be provided, giving a location where a valid
1443 structure message can be found containing the full details of the key family.

1444 *Element Content (Type):*

1445

1446 Name (common:TextType) - max. unbounded

1447 Description (common:TextType) - min. 0 - max. unbounded

1448 Components (ComponentsType) - min. 0

1449 Annotations (common:AnnotationsType) - min. 0

1450 *Attribute:* id (common:IDType) - required



1451 *Attribute:* agencyID (common:IDType) - required

1452 *Attribute:* version (xs:string) - optional

1453 *Attribute:* uri (xs:anyURI) - optional

1454 *Attribute:* urn (xs:anyURI) - optional

1455 *Attribute:* isFinal (xs:boolean) - optional

1456 *Attribute:* isExternalReference (xs:boolean) - optional

1457 *Attribute:* validFrom (common:TimePeriodType) - optional

1458 *Attribute:* validTo (common:TimePeriodType) - optional

1459 **ComponentsType:** ComponentsType describes the dimensions, groups,
1460 attributes, and measures of the key family. If TimeDimension is included in the
1461 key family - which it must be if time series formats for the data (GenericData,
1462 CompactData, and UtilityData formats) are to be used - then there must also
1463 be a frequency dimension.

1464 *Element Content (Type):*

1465

1466 Dimension (DimensionType) - min. 0 - max. unbounded

1467 TimeDimension (TimeDimensionType) - min. 0

1468 Group (GroupType) - min. 0 - max. unbounded

1469 PrimaryMeasure (PrimaryMeasureType)

1470 CrossSectionalMeasure (CrossSectionalMeasureType) - min. 0 - max.

1471 unbounded

1472 Attribute (AttributeType) - min. 0 - max. unbounded

1473 **DimensionType:** DimensionType describes the structure of non-Time
1474 dimensions. The order of their declaration is significant: it is used to describe
1475 the order in which they will appear in data formats for which key values are
1476 supplied in an ordered fashion (exclusive of the Time dimension, which is not
1477 represented as a member of the ordered key). Some types of non-Time
1478 dimensions have un-coded values: the TextFormat element must be provided,
1479 to indicate what type of values are permissible. The attributes
1480 isFrequencyDimension and isEntityDimension may have a "true" value for any
1481 instance of the Dimension element, indicating that it is a dimension of the
1482 stated type. The attributes isCountDimension,
1483 isNonObservationalTimeDimension, isMeasureDimension, and is
1484 IdentityDimension may occur multiple times, and take a true value to indicate
1485 that the dimension in question is of that type. Note that only one dimension in
1486 the key family may be of the following types: Frequency dimension and Entity
1487 dimension, and only if there is not also an attribute in the key family of the
1488 same type. Any given dimension may only have a true value for one of the set

1489 of attributes isFrequencyDimension, isCountDimension, is
 1490 measureDimension, isEntityDimension, isNonObservationalTimeDimension,
 1491 and is IdentityDimension. The definitions and limits on representation of each
 1492 dimension type are as follows: Frequency dimension describes the period
 1493 between observations, and is coded; Count dimensions are represented by
 1494 values which are sequential, incrementing numbers - representations are
 1495 always of the Increment or Count type; measureType dimensions are always
 1496 coded, and they enumerate the set of possible measures declared for the key
 1497 family; Entity dimensions describe the subject of the data set (ie, a country) -
 1498 they can be coded or represented in any other form; Non-Observational Time
 1499 dimensions must have a representation which contains a time; Identity
 1500 dimensions may be coded or uncoded, but must be represented by a scheme
 1501 which refers to the identifiers of external entites. (Conventionally, it is the first
 1502 dimension in the ordered set of dimensions - the key.) If a key family
 1503 describes cross-sectional data, then for each dimension, the
 1504 crossSectionalAttachDataSet, crossSectionalAttachGroup,
 1505 crossSectionalAttachSection, and crossSectionalAttachObservation attributes
 1506 must be given values. A value of "true" for any of these attributes indicates
 1507 that the dimension may be provided a value at the indicated level within the
 1508 cross-sectional structure. Note that these attributes do not need to be
 1509 provided for any dimension with the isFrequencyDimension set to "true", as
 1510 these dimensions are always attached only at the group level, as is time. A
 1511 key family designed for cross-sectional use must be structured such that any
 1512 observation's complete key can be unambiguously described by taking each
 1513 dimension value from its observation level, section level, group level, and data
 1514 set level, and ordered according to the sequence given in the key family. For
 1515 any dimension, the id of the referenced concept must be unique across the
 1516 entire key family, including all dimensions, attributes and measures.

1517 *Element Content (Type):*

1518

1519 TextFormat (TextFormatType) - min. 0

1520 Annotations (common:AnnotationsType) - min. 0

1521 *Attribute:* conceptRef (common:IDType) - required

1522 *Attribute:* conceptVersion (xs:string) - optional

1523 *Attribute:* conceptAgency (common:IDType) - optional

1524 *Attribute:* conceptSchemeRef (common:IDType) - optional

1525 *Attribute:* conceptSchemeAgency (common:IDType) - optional

1526 *Attribute:* codelist (common:IDType) - optional

1527 *Attribute:* codelistVersion (xs:string) - optional



- 1528 *Attribute:* codelistAgency (common:IDType) - optional
- 1529 *Attribute:* isMeasureDimension (xs:boolean) - default: false
- 1530 *Attribute:* isFrequencyDimension (xs:boolean) - default: false
- 1531 *Attribute:* isEntityDimension (xs:boolean) - default: false
- 1532 *Attribute:* isCountDimension (xs:boolean) - default: false
- 1533 *Attribute:* isNonObservationTimeDimension (xs:boolean) -
1534 default: false
- 1535 *Attribute:* isIdentityDimension (xs:boolean) - default: false
- 1536 *Attribute:* crossSectionalAttachDataSet (xs:boolean) - optional
- 1537 *Attribute:* crossSectionalAttachGroup (xs:boolean) - optional
- 1538 *Attribute:* crossSectionalAttachSection (xs:boolean) - optional
- 1539 *Attribute:* crossSectionalAttachObservation (xs:boolean) -
1540 optional
- 1541 **TimeDimensionType:** TimeDimensionType describes the special Time
1542 dimension. Any key family which will be used for time-series formats
1543 (GenericData, CompactData, and UtilityData) must include the time
1544 dimension. Any key family which uses the time dimension must also declare a
1545 frequency dimension, conventionally the first dimension in the key (the set of
1546 ordered non-time dimensions). A TextFormat element may be included for
1547 indicating the representation of time. The concept attribute must contain the
1548 concept name of the time concept. The codelist attribute may provide the
1549 value of the concept name of a codelist if needed. If a key family describes
1550 cross-sectional data, then for each dimension, the
1551 crossSectionalAttachDataSet, crossSectionalAttachGroup,
1552 crossSectionalAttachSection, and crossSectionalAttachObservation attributes
1553 must be given values. A value of "true" for any of these attributes indicates
1554 that the dimension may be provided a value at the indicated level within the
1555 cross-sectional structure. Note that these attributes do not need to be
1556 provided for any dimension with the isFrequencyDimension set to "true", as
1557 these dimensions are always attached only at the group level, as is time. A
1558 key family designed for cross-sectional use must be structured such that any
1559 observation's complete key can be unambiguously described by taking each
1560 dimension value from its observation level, section level, group level, and data
1561 set level, and ordered according to the sequence given in the key family.
- 1562 *Element Content (Type):*



- 1563
1564 TextFormat (TextFormatType) - min. 0
1565 Annotations (common:AnnotationsType) - min. 0
- 1566 *Attribute:* conceptRef (common:IDType) - required
- 1567 *Attribute:* conceptVersion (xs:string) - optional
- 1568 *Attribute:* conceptAgency (common:IDType) - optional
- 1569 *Attribute:* conceptSchemeRef (common:IDType) - optional
- 1570 *Attribute:* conceptSchemeAgency (common:IDType) - optional
- 1571 *Attribute:* codelist (common:IDType) - optional
- 1572 *Attribute:* codelistVersion (xs:string) - optional
- 1573 *Attribute:* codelistAgency (common:IDType) - optional
- 1574 *Attribute:* crossSectionalAttachDataSet (xs:boolean) - optional
- 1575 *Attribute:* crossSectionalAttachGroup (xs:boolean) - optional
- 1576 *Attribute:* crossSectionalAttachSection (xs:boolean) - optional
- 1577 *Attribute:* crossSectionalAttachObservation (xs:boolean) -
1578 optional
- 1579 **GroupType:** GroupType declares any useful groupings of data, based on a
1580 selection of the declared (non-Time) dimensions (indicated with the
1581 DimensionRef element) which form partial keys to which attributes may be
1582 attached. The value of the DimensionRef element is the concept of the
1583 dimension - that is, the value of the dimension's concept attribute. Thus, if
1584 data is to be presented as a set of time series which vary only according to
1585 their differing frequencies, a "sibling group" would be declared, with all
1586 dimensions except the frequency dimension in it. If data is commonly grouped
1587 as a set of all countries, then a "Country Group" could be declared, with all
1588 dimensions except the country dimension forming part of the partial key. Any
1589 dimension which is not part of a group has a value which varies at the series
1590 level (for time series formats). There is no requirement to have only a single
1591 dimension omitted from a partial key - it can be any subset of the set of
1592 ordered dimensions (that is, all dimensions except the time dimension, which
1593 may never be declared as belonging to a group partial key). All groups
1594 declared in the key family must be unique - that is, you may not have
1595 duplicate partial keys. All groups must also be given unique names (id
1596 attributes). Although it is conventional to declare dimensions in the same
1597 order as they are declared in the ordered key, there is no requirement to do so

1598 - the ordering of the values of the key are taken from the order in which the
1599 dimensions are declared. The Description element provides a human-
1600 readable description (potentially in multiple, parallel languages) of the group.
1601 Note that for cross-sectional formats, the named group mechanism is not
1602 used, but is instead replaced by a generic group which carries time and
1603 frequency values with it, and allows for any available group-level attributes to
1604 be specified if desired.

1605 *Element Content (Type):*

1606
1607 DimensionRef (common:IDType) [Choice] - max. unbounded
1608 AttachmentConstraintRef (common:IDType) [Choice]
1609 Description (common:TextType) - min. 0 - max. unbounded
1610 Annotations (common:AnnotationsType) - min. 0

1611 *Attribute:* id (common:IDType) - required

1612 **AttachmentConstraintRefType:** AttachmentConstraintRefType describes a
1613 reference to an attachment constraint. This includes a reference to a dataflow,
1614 metadataflow, data provider, or provision agreement plus the ID of the
1615 attachment constraint, as assigned within the constraints associated with the
1616 referenced object, in the ConstraintRef element.

1617 *Element Content (Type):*

1618
1619 DataflowRef (DataflowRefType) [Choice]
1620 MetadataflowRef (MetadataflowRefType) [Choice]
1621 DataProviderRef (DataProviderRefType) [Choice]
1622 ProvisionAgreementRef (ProvisionAgreementRefType) [Choice]
1623 ConstraintRef (common:IDType)

1624 **ProvisionAgreementRefType:** ProvisionAgreementRef allows for the
1625 identification of a provision agreement. At a minimum, either the URN element
1626 - holding a valid registry URN - or the set of OragnisationSchemeAgencyID,
1627 OrganisationSchemeID, DataProviderID, DataflowAgencyID, and DataflowID
1628 must be specified. Constraint can be used to express constraints associated
1629 with the provision agreement. This type differs from the similar type in the
1630 Registry namespace package by not providing information about the
1631 datasource or constraints.

1632 *Element Content (Type):*

1633
1634 URN (xs:anyURI) - min. 0
1635 OrganisationSchemeAgencyID (common:IDType) - min. 0
1636 OrganisationSchemeID (common:IDType) - min. 0
1637 DataProviderID (common:IDType) - min. 0
1638 DataProviderVersion (xs:string) - min. 0
1639 DataflowAgencyID (common:IDType) - min. 0
1640 DataflowID (common:IDType) - min. 0



1641 DataflowVersion (xs:string) - min. 0
1642 Constraint (common:ConstraintType) - min. 0

1643 **DataProviderRefType:** The DataProviderRef type structures a reference to a
1644 data provider. This requires that IDs be provided for an organisation scheme,
1645 its maintenance agency, and the data provider as identified in the referenced
1646 organisation scheme. The Version element may be used to specify the
1647 version of the indicated data provider. If absent, the most recent version is
1648 assumed. The URN element is used to provide the registry-specific urn as an
1649 alternate means of identification. At a minimum, either the URN element or
1650 OrganisationSchemeID, OrganisationSchemeAgencyID, DataProviderID,
1651 and (optionally) Version must be supplied. When used in a response
1652 document of any type, the URN must always be provided. Constraints can be
1653 used to specify constraints associated with the data provider. This type differs
1654 from the similar type in the Registry namespace by not describing the
1655 datasource.

1656 *Element Content (Type):*

1657
1658 URN (xs:anyURI) - min. 0
1659 OrganisationSchemeAgencyID (common:IDType)
1660 OrganisationSchemeID (common:IDType)
1661 DataProviderID (common:IDType)
1662 Version (xs:string) - min. 0
1663 Constraint (common:ConstraintType) - min. 0

1664 **AttributeType:** AttributeType describes the structure of attributes declared in
1665 the key family. If the codelist attribute is not used, then the attribute is
1666 uncoded. You may use the TextFormat element to specify constraints on the
1667 value of the uncoded attribute. The concept attribute contains the name of a
1668 concept. The codelist attribute supplies the id value of a codelist. The
1669 attachmentLevel attribute indicates the level to which the attribute is attached
1670 in time-series formats (GenericData, CompactData, and UtilityData formats).
1671 The assignmentStatus attribute indicates whether a value must be provided
1672 for the attribute when sending documentation along with the data. The
1673 AttachmentGroup element is included only when the attribute is attached at
1674 the Group level, to indicate which declared group or groups the attribute may
1675 be attached to. For each such group, an AttachmentGroup element should
1676 appear, with the content of the element being the name of the group. The
1677 AttachmentMeasure element is similar, indicating for cross-sectional formats
1678 which declared measure or measures the attribute attached at the observation
1679 level may be attached to. The isTimeFormat attribute indicates that the
1680 attribute represents the concept of time format (typically a mandatory series-
1681 level attribute with a codelist representation taken from ISO 8601). For key
1682 families not used to structure cross-sectional formats, this element may be
1683 omitted. Each such element contains the name of the declared measure.
1684 The attributes crossSectionalAttachDataSet, crossSectionalAttachGroup,
1685 crossSectionalAttachSection, and crossSectionalAttachObservation indicate

1686 what the attachment level or levels are for cross-sectional data formats, and
 1687 may be omitted if the key family will not be used to structure them. A value
 1688 of "true" indicates that it is permissible to provide a value for the attribute at
 1689 the specified level within the structure. Note that all groups in cross-sectional
 1690 formats are replaced by a generic group which has any values for time and
 1691 frequency, and allows any group-level attributes to be attached to it. An
 1692 attribute which is an Entity attribute has a true value for the isEntityAttribute
 1693 attribute - you may have either one Entity dimension or one Entity Attribute in
 1694 a key family; a non-observational time has a true value for
 1695 isNonObservationalTimeAttribute; and a Count attribute has a true value for
 1696 the isCountAttribute attribute. The attributes isFrequencyAttribute and
 1697 isEntityAttribute are mutually exclusive - that is, only one of them may have a
 1698 "true" value for any instance of the Attribute element. The definitions and limits
 1699 on representation of each attribute type are as follows: Frequency attribute
 1700 describes the period between observations, and is coded; Count attributes are
 1701 represented by values which are sequential, incrementing numbers -
 1702 representations are always of the Increment or Count type; Entity attributes
 1703 describe the subject of the data set - they can be coded or represented in any
 1704 other form; Non-Observational Time attributes must have a representation
 1705 which contains a time; Identity attributes may be coded or uncoded, but must
 1706 be represented by a scheme which refers to the identifiers of external entities.
 1707 Any given instance of an attribute may only have a true value for this set of
 1708 attributes, and if so may not have a true value for the isTimeFormat attribute.

1709 *Element Content (Type):*

1710
1711
1712
1713
1714

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

1715

Attribute: conceptRef (common:IDType) - required

1716

Attribute: conceptVersion (xs:string) - optional

1717

Attribute: conceptAgency (common:IDType) - optional

1718

Attribute: conceptSchemeRef (common:IDType) - optional

1719

Attribute: conceptSchemeAgency (common:IDType) - optional

1720

Attribute: codelist (common:IDType) - optional

1721

Attribute: codelistVersion (xs:string) - optional

1722

Attribute: codelistAgency (common:IDType) - optional



- 1723 *Attribute:* attachmentLevel (structure:AttachmentLevelType) -
1724 required
- 1725 *Attribute:* assignmentStatus (structure:AssignmentStatusType) -
1726 required
- 1727 *Attribute:* isTimeFormat (xs:boolean) - default: false
- 1728 *Attribute:* crossSectionalAttachDataSet (xs:boolean) - optional
- 1729 *Attribute:* crossSectionalAttachGroup (xs:boolean) - optional
- 1730 *Attribute:* crossSectionalAttachSection (xs:boolean) - optional
- 1731 *Attribute:* crossSectionalAttachObservation (xs:boolean) -
1732 optional
- 1733 *Attribute:* isEntityAttribute (xs:boolean) - default: false
- 1734 *Attribute:* isNonObservationalTimeAttribute (xs:boolean) -
1735 default: false
- 1736 *Attribute:* isCountAttribute (xs:boolean) - default: false
- 1737 *Attribute:* isFrequencyAttribute (xs:boolean) - default: false
- 1738 *Attribute:* isIdentityAttribute (xs:boolean) - default: false
- 1739 **PrimaryMeasureType:** PrimaryMeasureType describes the observation
1740 values for all presentations of the data, except those cross-sectional formats
1741 which have multiple measures (for which a set of cross-sectional measures
1742 are used instead). The concept attribute points to the unique concept
1743 represented by the measure. The PrimaryMeasure is conventionally
1744 associated with the OBS-VALUE concept. The TextFormat element allows
1745 description of the contents of the observation value. The codelist attribute
1746 references a codelist if the observation value is coded. If this attribute is used,
1747 then codelistAgencyID must contain the ID of the maintenance agency of the
1748 referenced codelist. The codelistVersion attribute may be specified - if not,
1749 then the version of the referenced codelist is assumed to be "1.0".
- 1750 *Element Content (Type):*
- 1751 TextFormat (TextFormatType) - min. 0
1752 Annotations (common:AnnotationsType) - min. 0
1753
- 1754 *Attribute:* conceptRef (common:IDType) - required

- 1755 *Attribute:* conceptVersion (xs:string) - optional
- 1756 *Attribute:* conceptAgency (common:IDType) - optional
- 1757 *Attribute:* conceptSchemeRef (common:IDType) - optional
- 1758 *Attribute:* conceptSchemeAgency (common:IDType) - optional
- 1759 *Attribute:* codelist (common:IDType) - optional
- 1760 *Attribute:* codelistVersion (xs:string) - optional
- 1761 *Attribute:* codelistAgency (common:IDType) - optional
- 1762 **CrossSectionalMeasureType:** CrossSectionalMeasureType describes the
 1763 observation values for multiple-measure cross-sectional data formats. For
 1764 non-cross sectional key families, it is not necessary to specify any cross-
 1765 sectional measures. The concept attribute points to the unique concept
 1766 represented by the measure. The measureDimension attribute contains the
 1767 concept name of the measure dimension. The code attribute contains the
 1768 value of its corresponding code in the codelist used to represent the measure
 1769 dimension. A CrossSectionalMeasure must be declared for each code in the
 1770 codelist used to represent the measure dimension - these will replace the
 1771 primary measure for multiple-measure cross-sectional data formats. The
 1772 TextFormat element allows description of the contents of the observation
 1773 value. The codelist attribute references a codelist if the observation value is
 1774 coded. If this attribute is used, then codelistAgencyID must contain the ID of
 1775 the maintenance agency of the referenced codelist. The codelistVersion
 1776 attribute may be specified - if not, then the version of the referenced codelist is
 1777 assumed to be "1.0".
- 1778 *Element Content (Type):*
- 1779 TextFormat (TextFormatType) - min. 0
- 1780 Annotations (common:AnnotationsType) - min. 0
- 1781
- 1782 *Attribute:* conceptRef (common:IDType) - required
- 1783 *Attribute:* conceptVersion (xs:string) - optional
- 1784 *Attribute:* conceptAgency (common:IDType) - optional
- 1785 *Attribute:* conceptSchemeRef (common:IDType) - optional
- 1786 *Attribute:* conceptSchemeAgency (common:IDType) - optional
- 1787 *Attribute:* codelist (common:IDType) - optional



- 1788 *Attribute:* codelistVersion (xs:string) - optional
- 1789 *Attribute:* codelistAgency (common:IDType) - optional
- 1790 *Attribute:* measureDimension (common:IDType) - required
- 1791 *Attribute:* code (common:IDType) - required
- 1792 **StructureSetsType:** StructureSetsType contains one or more structure sets.
- 1793 *Element Content (Type):*
- 1794
- 1795 StructureSet (StructureSetType) - max. unbounded
- 1796 **StructureSetType:** StructureSetType describes the relationships between
- 1797 two or more key families and/or metadata structure definitions, including the
- 1798 mapping between category schemes and concept schemes, to provide for the
- 1799 mapping of representations. This can include inheritance and extension of
- 1800 properties, or total or partial equivalencies. It also includes mapping of
- 1801 concepts existing in metadata structure definitions to those used in key
- 1802 families, and vice-versa. A human-readable name is provided in the Name
- 1803 element, which may include several language-specific variants. A longer
- 1804 human-readable description may also be provided, in the Description element,
- 1805 which may also have language-specific variants provided. The Annotations
- 1806 element may be used to provide annotations. The StructureRefs element
- 1807 references all of the key families and/or metadata structure definitions
- 1808 included in the Structure Set - these must be provided if a StructureMap
- 1809 element is used, but is not required if the structure set is only used to provide
- 1810 codelist mappings, concept mappings, or category mappings. The
- 1811 StructureMap element indicates which components in the included data and
- 1812 metadata structures are equivalent; CodelistMap indicates which codes map
- 1813 to other codelists. CategorySchemeMap indicates which categories in one
- 1814 scheme map to those in another scheme. ConceptSchemeMap indicates
- 1815 which concepts in one scheme map to those in another scheme.
- 1816 OrganisationSchemeMap describes how one organisation scheme maps to
- 1817 another. The id attribute takes an id which is unique to all structure sets
- 1818 maintained by the agency specified in the agency attribute. version specifies a
- 1819 version number (by default "1.0"). The uri attribute holds a URL where a valid
- 1820 SDMX Structure message can be found which provides full details of the
- 1821 StructureSet, and it must be used if the isExternalReference attribute has a
- 1822 value of true. The urn attribute holds a valid SDMX Registry URN as
- 1823 described in the SDMX Registry specification. A true value in the isFinal
- 1824 attribute indicates that the contents of the structure set may not be changed
- 1825 without versioning. The validFrom and validTo attributes provide inclusive
- 1826 dates for providing supplemental validity information about the version.
- 1827 *Element Content (Type):*

1828	
1829	Name (common:TextType) - max. unbounded
1830	Description (common:TextType) - min. 0 - max. unbounded
1831	RelatedStructures (RelatedStructuresType) - min. 0
1832	StructureMap (StructureMapType) - min. 0
1833	CodelistMap (CodelistMapType) - min. 0
1834	CategorySchemeMap (CategorySchemeMapType) - min. 0
1835	ConceptSchemeMap (ConceptSchemeMapType) - min. 0
1836	OrganisationSchemeMap (OrganisationSchemeMapType) - min. 0
1837	Annotations (common:AnnotationsType) - min. 0
1838	<i>Attribute:</i> id (common:IDType) - required
1839	<i>Attribute:</i> agencyID (common:IDType) - optional
1840	<i>Attribute:</i> version (xs:string) - optional
1841	<i>Attribute:</i> urn (xs:anyURI) - optional
1842	<i>Attribute:</i> uri (xs:anyURI) - optional
1843	<i>Attribute:</i> isFinal (xs:boolean) - optional
1844	<i>Attribute:</i> isExternalReference (xs:boolean) - optional
1845	<i>Attribute:</i> validFrom (common:TimePeriodType) - optional
1846	<i>Attribute:</i> validTo (common:TimePeriodType) - optional
1847	RelatedStructuresType: RelatedStructuresType includes references to key
1848	families (in the KeyFamilyRef element) and/or metadata structure definitions
1849	(In the MetadataStructureRef element). Any mapped CategorySchemes,
1850	ConceptSchemes, or Organisation Schemes should also be referenced.
1851	HierarchicalCodelistRef allows for HierarchicalCodelists which describe
1852	relationships between pertinent codelists to be referenced and included in the
1853	structure set - this must be used if the CodelistMap in the StructureSet refers
1854	to any hierarchical codelists.
1855	<i>Element Content (Type):</i>
1856	
1857	KeyFamilyRef (KeyFamilyRefType) - min. 0 - max. unbounded
1858	MetadataStructureRef (MetadataStructureRefType) - min. 0 - max.
1859	unbounded
1860	ConceptSchemeRef (ConceptSchemeRefType) - min. 0 - max. unbounded
1861	CategorySchemeRef (CategorySchemeRefType) - min. 0 - max. unbounded
1862	OrganisationSchemeRef (OrganisationSchemeRefType) - min. 0 - max.
1863	unbounded
1864	HierarchicalCodelistRef (HierarchicalCodelistRefType) - min. 0 - max.
1865	unbounded

1866 **CategorySchemeRefType:** CategorySchemeRef allows for references to
1867 specific category schemes. At a minimum, either the URN - which contains a
1868 valid Registry/Repository URN - or the rest of the child elements must be
1869 supplied.

1870 *Element Content (Type):*

1871
1872 URN (xs:anyURI) - min. 0
1873 AgencyID (common:IDType) - min. 0
1874 CategorySchemeID (common:IDType) - min. 0
1875 Version (xs:string) - min. 0

1876 **ConceptSchemeRefType:** ConceptSchemeRef allows for references to
1877 specific concept schemes. At a minimum, either the URN - which contains a
1878 valid Registry/Repository URN - or the rest of the child elements must be
1879 supplied.

1880 *Element Content (Type):*

1881
1882 URN (xs:anyURI) - min. 0
1883 AgencyID (common:IDType) - min. 0
1884 ConceptSchemeID (common:IDType) - min. 0
1885 Version (xs:string) - min. 0

1886 **OrganisationSchemeRefType:** OrganisationSchemeRef allows for
1887 references to specific organisation schemes. At a minimum, either the URN -
1888 which contains a valid Registry/Repository URN - or the rest of the child
1889 elements must be supplied.

1890 *Element Content (Type):*

1891
1892 URN (xs:anyURI) - min. 0
1893 AgencyID (common:IDType) - min. 0
1894 OrganisationSchemeID (common:IDType) - min. 0
1895 Version (xs:string) - min. 0

1896 **HierarchicalCodelistRefType:** HierarchicalCodelistRef allows for references
1897 to specific hierarchical codelists. At a minimum, either the URN - which
1898 contains a valid Registry/Repository URN - or the rest of the child elements
1899 must be supplied.

1900 *Element Content (Type):*

1901
1902 URN (xs:anyURI) - min. 0
1903 AgencyID (common:IDType) - min. 0
1904 HierarchicalCodelistID (common:IDType) - min. 0
1905 Version (xs:string) - min. 0



1906 **StructureMapType:** StructureMapType describes the structure of the
1907 mapping of components between a referenced key family or metadata
1908 structure and a target key family or metadata structure. Components include
1909 any dimension, attribute, or reported concept. The Name element is used to
1910 provide a human-readable name for the component map; the Description
1911 element is used to provide a longer human-readable description. Both of
1912 these elements may be provided in multiple, language-specific variations. The
1913 StructureMapType provides for Annotations with the Annotations element.
1914 Either a KeyFamilyRef or a MetadataStructureRef must be provided; and also
1915 a TargetKeyFamilyRef or a TargetMetadataStructureRef. A series of map
1916 components are then specified using the ComponentMap element, each of
1917 which specifies the equivalence of a concept in the referenced structure
1918 definition to one in the target structure definition. If the isExtension attribute
1919 has a value of true, then the target structure definition inherits all properties of
1920 the referenced structure definition, and may have additional components.
1921 Note that this attribute may only be set to true if the component map has as a
1922 referenced structure definition and a target structure definition either two key
1923 families or two metadata structure definition. You cannot inherit concepts
1924 between the two type of structure definitions using this mechanism. The id
1925 attribute allows for an id to be assigned to the component map - it must be
1926 unique within its StructureSet.

1927 *Element Content (Type):*

1928 Name (common:TextType) - max. unbounded
1929 Description (common:TextType) - min. 0 - max. unbounded
1930 KeyFamilyRef (KeyFamilyRefType) [Choice]
1931 MetadataStructureRef (MetadataStructureRefType) [Choice]
1932 TargetKeyFamilyRef (KeyFamilyRefType) [Choice]
1933 TargetMetadataStructureRef (MetadataStructureRefType) [Choice]
1934 ComponentMap (ComponentMapType) - min. 0 - max. unbounded
1935 Annotations (common:AnnotationsType) - min. 0
1936

1937 *Attribute:* isExtension (xs:boolean) - optional

1938 *Attribute:* id (common:IDType) - required

1939 **CodelistMapType:** CodelistMap allows the description of how the codes in a
1940 codelist are represented in a target codelist or associated hierarchical
1941 codelist. A human-readable Name is provided, and a longer, human-readable
1942 description may be provided as well, in the Name and Description elements
1943 respectively. These may be supplied in multiple, language-specific
1944 versions. CodelistRef references the codelist or hierarchical codelist being
1945 mapped; TargetCodelistRef indicates the codelist to which it will be mapped.
1946 CodeMap is the element which indicates the equivalence of codes in the
1947 referenced codelist to those in the target codelist. Any codes not mapped are
1948 assumed to lack equivalence. The CodelistMap may be annotated using the



1949 Annotations element. The id attribute is used to assign an identifier which is
1950 unique within the StructureSet for all CodelistMaps.

1951 *Element Content (Type):*

1952

1953

Name (common:TextType) - max. unbounded

1954

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

1955

CodelistRef (CodelistRefType) [Choice]

1956

HierarchicalCodelistRef (HierarchicalCodelistRefType) [Choice]

1957

TargetCodelistRef (CodelistRefType) [Choice]

1958

TargetHierarchicalCodelistRef (HierarchicalCodelistRefType) [Choice]

1959

CodeMap (CodeMapType) - max. unbounded

1960

Annotations (common:AnnotationsType) - min. 0

1961

Attribute: id (common:IDType) - required

1962 **CodeMapType:** CodeMap describes the equivalence of the codes in the
1963 referenced codelist or hierarchical codelist indicated in the CodelistRef
1964 element of the containing CodelistMap to those in the referenced
1965 TargetCodelist in the containing CodelistMap. The CodeAlias attribute is used
1966 to assign an alias code to the equivalence for querying the structure set.

1967 *Element Content (Type):*

1968

1969

MapCodeRef (common:IDType)

1970

MapTargetCodeRef (common:IDType)

1971

Attribute: CodeAlias (common:IDType) - optional

1972 **ComponentMapType:** ComponentMapType describes how a component
1973 (that is, dimension, attribute, or reported concept) in a referenced metadata
1974 structure definition or key family maps to a component in a referenced "target"
1975 metadata structure definition or key family. The MapConceptRef contains the
1976 id of the concept in the metadata structure definition or key family referenced
1977 in the KeyFamilyRef or MetadataStructureRef element of the containing
1978 ComponentMap element. The MapTargetConceptRef contains the id of the
1979 concept in the metadata structure definition or key family referenced in the
1980 TargetKeyFamilyRef or TargetMetadataStructureRef element of the
1981 containing ComponentMap element. The RepresentationMapRef element
1982 contains a reference to the CodelistMap which describes how the coded
1983 representation of the referenced component maps to the coded representation
1984 of the target component. If the target component has an uncoded
1985 representation, then ToTextFormat is used to describe the un-coded
1986 representation to which the code of the referenced component should be
1987 transformed. The ToValueType element tells you whether the value, name, or
1988 description of the source value should be used in the resulting text field. The
1989 componentAlias attribute assigns a new ID to the relationship between these
1990 components. Note that of three components from different key families and/or

1991 metadata structure definitions are all equivalent, the two component maps can
 1992 share a single alias. Note also that for metadata concepts which are
 1993 represented not by codelists but rather by other types of item schemes
 1994 (OrganisationSchemes or CategorySchemes), these can also be referenced
 1995 using the RepresentationMapRef element. The preferredLanguage attribute
 1996 specifies the language to use when translating coded values into their names
 1997 or descriptions, if available, in the same form as xml:lang.

1998 *Element Content (Type):*

1999
 2000 MapConceptRef (common:IDType)
 2001 MapTargetConceptRef (common:IDType)
 2002 RepresentationMapRef (RepresentationMapRefType) [Choice]

2003 *Attribute:* componentAlias (common:IDType) - optional

2004 *Attribute:* preferredLanguage (xs:language) - default: en

2005 **RepresentationMapRefType:** RepresentationMapRefType describes the
 2006 structure of a reference to a codelist, category scheme, or organisation
 2007 scheme map. RepresentationMapAgencyID takes the id value of the
 2008 maintenance agency of the codelist, category scheme, or organisation
 2009 scheme map; RepresentationMapID takes the id attribute value of the
 2010 codelist, category scheme, or organisation scheme map.

2011 *Element Content (Type):*

2012
 2013 RepresentationMapAgencyID (common:IDType)
 2014 RepresentationMapID (common:IDType)

2015 *Attribute:* representationType (RepresentationTypeType) -
 2016 default: Codelist

2017 **CategorySchemeMapType:** CategorySchemeMap provides for the mapping
 2018 of categories in one scheme against those in another. It requires a human-
 2019 readable Name, and can have a longer human-readable Description, both of
 2020 which can be supplied in multiple, parallel-language form. It may be annotated
 2021 using Annotations. The id attribute carries a unique ID for
 2022 CategorySchemeMaps within the StructureSet. CategorySchemeRef identifies
 2023 the source CategoryScheme; TargetCategorySchemeRef identifies the target
 2024 CategoryScheme.

2025 *Element Content (Type):*

2026
 2027 Name (common:TextType) - max. unbounded
 2028 Description (common:TextType) - min. 0 - max. unbounded
 2029 CategorySchemeRef (CategorySchemeRefType)



2030 TargetCategorySchemeRef (CategorySchemeRefType)
2031 CategoryMap (CategoryMapType) - max. unbounded
2032 Annotations (common:AnnotationsType) - min. 0

2033 *Attribute:* id (common:IDType) - required

2034 **CategoryMapType:** CategoryMap allows for the mapping of a category in one
2035 scheme against a category in another, target scheme. The categoryAlias
2036 attribute allows for an alias to be assigned to the mapping for searching
2037 across the set of mapped categories. Note that the Category IDs are
2038 recursive, and can therefore describe a path through a category scheme.

2039 *Element Content (Type):*

2040
2041 CategoryID (CategoryIDType)
2042 TargetCategoryID (CategoryIDType)

2043 *Attribute:* categoryAlias (common:IDType) - optional

2044 **ConceptSchemeMapType:** ConceptSchemeMap provides for the mapping of
2045 concepts in one scheme against those in another. It requires a human-
2046 readable Name, and can have a longer human-readable Description, both of
2047 which can be supplied in multiple, parallel-language form. It may be annotated
2048 using Annotations. The id attribute carries a unique ID for
2049 ConceptSchemeMaps within the StructureSet. ConceptSchemeRef identifies
2050 the source ConceptScheme; TargetConceptSchemeRef identifies the target
2051 ConceptScheme.

2052 *Element Content (Type):*

2053
2054 Name (common:TextType) - max. unbounded
2055 Description (common:TextType) - min. 0 - max. unbounded
2056 ConceptSchemeRef (ConceptSchemeRefType)
2057 TargetConceptSchemeRef (ConceptSchemeRefType)
2058 ConceptMap (ConceptMapType) - max. unbounded
2059 Annotations (common:AnnotationsType) - min. 0

2060 *Attribute:* id (common:IDType) - required

2061 **ConceptMapType:** ConceptMap allows for the mapping of a concept in one
2062 scheme against a concept in another, target scheme. The conceptAlias
2063 attribute allows for an alias to be assigned to the mapping for searching
2064 across the set of mapped concepts.

2065 *Element Content (Type):*

2066
2067 ConceptID (common:IDType)
2068 TargetConceptID (common:IDType)

2069 *Attribute:* conceptAlias (common:IDType) - optional

2070 **OrganisationSchemeMapType:** OrganisationSchemeMap provides for the
2071 mapping of Organisations in one scheme against those in another. It requires
2072 a human-readable Name, and can have a longer human-readable Description,
2073 both of which can be supplied in multiple, parallel-language form. It may be
2074 annotated using Annotations. The id attribute carries a unique ID for
2075 OrganisationSchemeMaps within the StructureSet. OrganisationSchemeRef
2076 identifies the source OrganisationScheme; TargetOrganisationSchemeRef
2077 identifies the target OrganisationScheme.

2078 *Element Content (Type):*

2079
2080 Name (common:TextType) - max. unbounded
2081 Description (common:TextType) - min. 0 - max. unbounded
2082 OrganisationSchemeRef (OrganisationSchemeRefType)
2083 TargetOrganisationSchemeRef (OrganisationSchemeRefType)
2084 OrganisationMap (OrganisationMapType) - max. unbounded
2085 Annotations (common:AnnotationsType) - min. 0

2086 *Attribute:* id (common:IDType) - required

2087 **OrganisationMapType:** OrganisationMap allows for the mapping of an
2088 organisation in one scheme against an organisation in another, target
2089 scheme. The organisationAlias attribute allows for an alias to be assigned to
2090 the mapping for searching across the set of mapped organisations.

2091 *Element Content (Type):*

2092
2093 OrganisationID (common:IDType)
2094 TargetOrganisationID (common:IDType)

2095 *Attribute:* organisationAlias (common:IDType) - optional

2096 **ReportingTaxonomiesType:** ReportingTaxonomiesType holds on or more
2097 ReportingTaxonomy elements.

2098 *Element Content (Type):*

2099
2100 ReportingTaxonomy (ReportingTaxonomyType) - max. unbounded

2101 **ReportingTaxonomyType:** ReportingTaxonomyType groups data flows
2102 and/or metadata flows for the purposes of assembling "reports" made up of
2103 data from disparate sources. It is a maintainable object, and thus has a
2104 mandatory human-readable Name and optional Description containing a
2105 longer human-readable description. Annotations may be included. All of these
2106 fields may be provided in multiple, parallel languages. The id attribute



2107 assigns a unique ID to the Reporting Taxonomy, version provides a version
2108 number, uri contains a URL where the SDMX-ML expression of the Reporting
2109 taxonomy can be found, and must be included if the isExternalReference
2110 attribute has a value of true. The urn attribute holds the value of a valid SDMX
2111 Registry URN as per the SDMX Registry specification. The
2112 isExternalReference attribute, if set to true, indicates that the uri attribute
2113 points to an external location for the ReportingTaxonomy, with only the id,
2114 Name element, and version supplied in addition. The agencyID attribute holds
2115 the ID of the Reporting Taxonomies' maintenance agency. Also, if the
2116 Reporting Taxonomy is final, the isFinal attribute must have a value of true -
2117 otherwise, it will be assumed to be non-final. (All production versions must be
2118 made final - that is, unchangeable without versioning.) The sub-element
2119 Category may be used to group dataflows and metadataflows into useful sub-
2120 packages. DataflowRef and MetadataFlowRef are references to the flows
2121 which make up the reporting taxonomy at the top level. The validFrom and
2122 validTo attributes provide inclusive dates for providing supplemental validity
2123 information about the version.

2124 *Element Content (Type):*

2125 Name (common:TextType) - max. unbounded
2126 Description (common:TextType) - min. 0 - max. unbounded
2127 DataflowRef (DataflowRefType) - min. 0 - max. unbounded
2128 MetadataflowRef (MetadataflowRefType) - min. 0 - max. unbounded
2129 Category (CategoryType) - min. 0 - max. unbounded
2130 Annotations (common:AnnotationsType) - min. 0

2132 *Attribute:* id (common:IDType) - required

2133 *Attribute:* version (xs:string) - optional

2134 *Attribute:* uri (xs:anyURI) - optional

2135 *Attribute:* urn (xs:anyURI) - optional

2136 *Attribute:* isExternalReference (xs:boolean) - optional

2137 *Attribute:* agencyID (common:IDType) - required

2138 *Attribute:* isFinal (xs:boolean) - optional

2139 *Attribute:* validFrom (common:TimePeriodType) - optional

2140 *Attribute:* validTo (common:TimePeriodType) - optional

2141 **MetadataflowRefType:** The MetadataflowRef type structures a reference to a
2142 metadataflow definition. This requires that ID are provided for a pre-existing
2143 Agency and Metadataflow Definition in the registry. The Version element may



2144 be used to specify the version of the indicated dataflow. If absent, the most
2145 recent version is assumed. The URN element is used to provide the registry-
2146 specific URN as an alternate means of identification. When used in a
2147 response document of any type, the URN must always be provided. At a
2148 minimum, either the URN element or AgencyID, MetadataflowID, and
2149 (optionally) version must be supplied. Datasource may be used to specify a
2150 datasource. Constraint can be used to provide constraints associated with the
2151 metadataflow. Note that this is similar, but not identical to the
2152 MetadataflowRefType found in the SDMX-ML registry namespace package - it
2153 lacks references to the datasource and the constraints.

2154 *Element Content (Type):*

2155
2156 URN (xs:anyURI) - min. 0
2157 AgencyID (common:IDType) - min. 0
2158 MetadataflowID (common:IDType) - min. 0
2159 Version (xs:string) - min. 0

2160 **DataflowRefType:** The DataflowRef type structures a reference to a dataflow
2161 definition. This requires that ID are provided for a pre-existing Agency and
2162 Dataflow Definition in the registry. The Version element may be used to
2163 specify the version of the indicated dataflow. If absent, the most recent
2164 version is assumed. The URN element is used to provide the registry-specific
2165 URN as an alternate means of identification. At a minimum, either the URN
2166 element or AgencyID, DataflowID, and (optionally) version must be supplied.
2167 When used in a response document of any type, the URN must always be
2168 provided. Datasource may be used to specify a datasource. Constraints can
2169 be used to specify constraints associated with the dataflow. Note that this is
2170 similar, but not identical to the DataflowRefType found in the SDMX-ML
2171 registry namespace package - it lacks references to the datasource and the
2172 constraints.

2173 *Element Content (Type):*

2174
2175 URN (xs:anyURI) - min. 0
2176 AgencyID (common:IDType) - min. 0
2177 DataflowID (common:IDType) - min. 0
2178 Version (xs:string) - min. 0

2179 **ProcessesType:** ProcessesType describes a list of Processes.

2180 *Element Content (Type):*

2181
2182 Process (ProcessType) - max. unbounded

2183 **ProcessType:** ProcessType generically describes a statistical process. In this
2184 version of the SDMX Technical Specifications, it is not meant to support

2185 process automation, but serves as a description of how processes occur. The
 2186 primary use for this structural mechanism is the attachment of reference
 2187 metadata regarding statistical processing. A process has a human-readable
 2188 Name, which may be provided in multiple, parallel-language versions. It also
 2189 has an optional human-readable Description, which also may be provided with
 2190 multiple, parallel-language versions. The Annotations element allows for it to
 2191 be annotated. The id attribute takes a unique id within the set of processes
 2192 maintained by the agency identified in the agencyID attribute. The version
 2193 attribute indicated the version of the process description. The uri value is a
 2194 URL where a complete description of the Process may be found; the urn
 2195 attribute takes the valid registry URN of the Process, as described in the
 2196 SDMX Registry Specification. If isFinal is set to true, the process description
 2197 cannot be changed without versioning. If isExternalReference is true, then
 2198 only the id, agency, Name, and uri (or URN) need be supplied - all other fields
 2199 are assumed to be found in a valid SDMX Structure message found at the uri
 2200 attribute location. The validFrom and validTo attributes provide inclusive dates
 2201 for providing supplemental validity information about the version.

2202 *Element Content (Type):*

2203
 2204 Name (common:TextType) - max. unbounded
 2205 Description (common:TextType) - min. 0 - max. unbounded
 2206 ProcessStep (ProcessStepType) - min. 0 - max. unbounded
 2207 Annotations (common:AnnotationsType) - min. 0

2208 *Attribute:* id (common:IDType) - required

2209 *Attribute:* version (xs:string) - optional

2210 *Attribute:* uri (xs:anyURI) - optional

2211 *Attribute:* urn (xs:anyURI) - optional

2212 *Attribute:* isExternalReference (xs:boolean) - optional

2213 *Attribute:* agencyID (common:IDType) - required

2214 *Attribute:* isFinal (xs:boolean) - optional

2215 *Attribute:* validFrom (common:TimePeriodType) - optional

2216 *Attribute:* validTo (common:TimePeriodType) - optional

2217 **ProcessStepType:** ProcessStepType describes a single step in a statistical
 2218 process. ProcessSteps may be recursive. The Input element specifies the
 2219 type of object(s) which serve as inputs to the process; the Output element
 2220 specifies the type of objects which are the result of the process. Computation
 2221 elements describe the computations involved in the process, in any form

2222 desired by the user (these are informational rather than machine-actionable),
 2223 and so may be supplied in multiple, parallel-language versions. Transitions
 2224 describe the process steps to which a process is connected - that is, which
 2225 processes happen next. The process step must be given a Name, and may
 2226 be given a Description. These are human-readable, and may be supplied in
 2227 multiple, parallel-language versions. Annotations may be supplied. The id
 2228 attribute takes the unique identifier of the process step within the parent
 2229 process.

2230 *Element Content (Type):*

2231
 2232 Name (common:TextType) - max. unbounded
 2233 Description (common:TextType) - min. 0 - max. unbounded
 2234 Input (ObjectIDType) - min. 0 - max. unbounded
 2235 Output (ObjectIDType) - min. 0 - max. unbounded
 2236 Computation (common:TextType) - min. 0 - max. unbounded
 2237 Transition (TransitionType) - min. 0 - max. unbounded
 2238 ProcessStep (ProcessStepType) - min. 0 - max. unbounded
 2239 Annotations (common:AnnotationsType) - min. 0

2240 *Attribute:* id (common:IDType) - required

2241 **TransitionType:** TransitionType describes the Condition and next step in a
 2242 transition. The Condition text is informational, and may be supplied in multiple,
 2243 parallel-language form. The TargetStep holds the id of the next step in the
 2244 process if the condition is met.

2245 *Element Content (Type):*

2246
 2247 TargetStep (common:IDType) - min. 0
 2248 Condition (common:TextType) - min. 0

2249

2250 5.2.2 Simple Types

2251 **ObjectIDType:** The Object ID is used to reference a particular Object within
 2252 the SDMX Information Model's formalization of statistical exchanges.

2253 *Restricts* xs:NMTOKEN

2254 Code: Agency - Agency

2255 Code: ConceptScheme - Concept scheme

2256 Code: Concept - Concept

2257 Code: Codelist - Codelist

2258 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 provider
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: XSDDataSet - Cross-sectional data set
2284	Code: MetadataSet - Metadata set
2285	Code: HierarchicalCodelist - Hierarchical codelist
2286	Code: Hierarchy - Hierarchy



2287	Code: StructureSet - Structure set
2288	Code: StructureMap - Structure map
2289	Code: ComponentMap - Component map
2290	Code: CodelistMap - Codelist map
2291	Code: CodeMap - Code map
2292	Code: CategorySchemeMap - Category scheme map
2293	Code: CategoryMap - Category map
2294	Code: OrganisationSchemeMap - Organisation scheme map
2295	Code: OrganisationRoleMap - Organisation role map
2296	Code: ConceptSchemeMap - Concept scheme map
2297	Code: ConceptMap - Concept map
2298	Code: Process - Process
2299	Code: ProcessStep - Process step
2300	TextTypeType: TextTypeType provides an enumerated list of the types of
2301	characters allowed in a TextFormat field.
2302	<i>Restricts xs:NMTOKEN</i>
2303	Code: String - A string datatype corresponding to W3C XML Schema's xs:string
2304	datatype.
2305	Code: BigInteger - An integer datatype corresponding to W3C XML Schema's
2306	xs:integer datatype.
2307	Code: Integer - An integer datatype corresponding to W3C XML Schema's xs:int
2308	datatype.
2309	Code: Long - A numeric datatype corresponding to W3C XML Schema's xs:long
2310	datatype.
2311	Code: Short - A numeric datatype corresponding to W3C XML Schema's xs:short
2312	datatype.
2313	Code: Decimal - A numeric datatype corresponding to W3C XML Schema's
2314	xs:decimal datatype.
2315	Code: Float - A numeric datatype corresponding to W3C XML Schema's xs:float
2316	datatype.
2317	Code: Double - A numeric datatype corresponding to W3C XML Schema's xs:double
2318	datatype.



2319 2320	Code: Boolean - A datatype corresponding to W3C XML Schema's xs:boolean datatype.
2321 2322	Code: DateTime - A time datatype corresponding to W3C XML Schema's xs:dateTime datatype.
2323 2324	Code: Date - A time datatype corresponding to W3C XML Schema's xs:date datatype.
2325 2326	Code: Time - A time datatype corresponding to W3C XML Schema's xs:time datatype.
2327 2328	Code: Year - A time datatype corresponding to W3C XML Schema's xs:gYear datatype.
2329 2330	Code: Month - A time datatype corresponding to W3C XML Schema's xs:gMonth datatype.
2331 2332	Code: Day - A time datatype corresponding to W3C XML Schema's xs:gDay datatype.
2333 2334	Code: MonthDay - A time datatype corresponding to W3C XML Schema's xs:gMonthDay datatype.
2335 2336	Code: YearMonth - A time datatype corresponding to W3C XML Schema's xs:gYearMonth datatype.
2337 2338	Code: Duration - A time datatype corresponding to W3C XML Schema's xs:duration datatype.
2339	Code: URI - A datatype corresponding to W3C XML Schema's xs:anyURI datatype.
2340 2341 2342	Code: Timespan - A complex datatype specifying a start date (xs:dateTime) and a duration (xs:duration). Note that this is not allowed as the text type representing a dimension.
2343 2344	Code: Count - A simple incrementing Integer type. The isSequence facet must be set to true, and the interval facet must be set to "1".
2345 2346	Code: InclusiveValueRange - This value indicates that the startValue and endValue attributes provide an inclusive numeric range of type xs:double.
2347 2348	Code: ExclusiveValueRange - This value indicates that the startValue and endValue attributes provide an exclusive numeric range, of type xs:double.
2349 2350	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.
2351 2352 2353 2354	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.
2355 2356 2357	UsageStatusType: UsageStatus provides a list of enumerated types for indicating whether reporting a given metadata attribute is mandatory or conditional.



2358 *Restricts xs:NMTOKEN*

2359 Code: Mandatory - Reporting the associated attribute is mandatory - a value must be
2360 supplied.

2361 Code: Conditional - Reporting the associated attribute is not mandatory - a value may
2362 be supplied, but is not required.

2363 **RepresentationSchemeTypeType:** Representation scheme type provides an
2364 enumerated list of valid types of representation schemes.

2365 *Restricts xs:NMTOKEN*

2366 Code: Codelist - Representation scheme is a codelist.

2367 Code: Concept - Representation scheme is a concept scheme.

2368 Code: Category - Representation scheme is a category scheme.

2369 Code: Organisation - Representation scheme is an organisation scheme.

2370 Code: External - Representation scheme is "external" to the known model - that is, it
2371 cannot be enumerated at the time the report is designed. This will only be valid if
2372 some maintained and changing object is to have metadata reported against it: for
2373 example, if the concepts of dimension objects are to be reported against for all of an
2374 agencies' key families, then it is not possible at design time to enumerate all of the
2375 concepts which will be used by that agencies' key families into the future. This value
2376 should not be used unless absolutely necessary, as it reduces the processability of
2377 the metadata report generated.

2378 **AttachmentLevelType:**

2379 *Restricts xs:NMTOKEN*

2380 Code: DataSet - Data set level

2381 Code: Group - Group level

2382 Code: Series - Series level

2383 Code: Observation - Observation level

2384 **AssignmentStatusType:**

2385 *Restricts xs:NMTOKEN*

2386 Code: Mandatory - Providing attribute value is mandatory

2387 Code: Conditional - Providing attribute value is optional

2388 **ToValueTypeType:** ToValueTypeType provides an enumeration of available
2389 text-equivalents for translation of coded values into textual formats.



2390 *Restricts xs:NMTOKEN*

2391 Code: Value - Code or other tokenized value, as provided in the representation
2392 scheme.

2393 Code: Name - The human-readable name of the Value, as provided in the
2394 representation scheme.

2395 Code: Description - The human-readable description of the Value, as provided in the
2396 representation scheme.

2397 **RepresentationTypeType:** RepresentationTypeType provides an
2398 enumeration of representation scheme types useful for the mapping of
2399 reference metadata concepts to one another.

2400 *Restricts xs:NMTOKEN*

2401 Code: Codelist - Codelist

2402 Code: CategoryScheme - CategoryScheme

2403 Code: OrganisationScheme - OrganisationScheme

2404

2405

2406 **5.3 SDMX Generic Data Namespace Module**

2407

2408 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/generic**

2409 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
2410 (SDMXCommon.xsd)

2411

2412 **5.3.1 Global Elements**

2413 **DataSet(DataSetType):** The DataSet element contains one or more groups
2414 that comprise the data set.

2415

2416 **5.3.2 Complex Types**

2417 **DataSetType:** DataSetType defines the structure of a data set. This consists
2418 of a key family reference which contains the ID of the key family, and the
2419 attribute values attached at the data set level. A DataSet may be used to
2420 transmit documentation (that is, only attribute values), data, or a combination
2421 of both. If providing only documentation, you need not send the complete set
2422 of attributes. If transmitting only data, the Group may be omitted if desired.
2423 Uniqueness constraints are defined for the attributes of the data set. If

2424 dataset-level attributes are sent in a delete message, then any valid attribute
2425 value will indicate that the current attribute value should be deleted. The
2426 keyFamilyURI attribute is provided to allow a URI (typically a URL) to be
2427 provided, pointing to an SDMX-ML Structure message describing the key
2428 family. Attributes are provided for describing the contents of a data or
2429 metadata set, which are particularly important for interactions with the SDMX
2430 Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID,
2431 dataflowAgencyID, and dataflowID all take the IDs specified by the attribute
2432 names. The action attribute indicates whether the file is appending, replacing,
2433 or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate,
2434 and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit
2435 year, and publicationPeriod specifies the period of publication of the data in
2436 terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if
2437 that is the time of publication for a data set published on a quarterly basis).

2438 *Element Content (Type):*

2439

2440 KeyFamilyRef (common:IDType)
2441 Attributes (ValueType) - min. 0
2442 Group (GroupType) [Choice] - min. 0 - max. unbounded
2443 Series (SeriesType) [Choice] - min. 0 - max. unbounded
2444 Annotations (common:AnnotationsType) - min. 0

2445 *Attribute:* keyFamilyURI (xs:anyURI) - optional

2446 *Attribute:* datasetID (common:IDType) - optional

2447 *Attribute:* dataProviderSchemeAgencyId (common:IDType) -
2448 optional

2449 *Attribute:* dataProviderSchemeId (common:IDType) - optional

2450 *Attribute:* dataProviderID (common:IDType) - optional

2451 *Attribute:* dataflowAgencyID (common:IDType) - optional

2452 *Attribute:* dataflowID (common:IDType) - optional

2453 *Attribute:* action (common:ActionType) - optional

2454 *Attribute:* reportingBeginDate (common:TimePeriodType) -
2455 optional

2456 *Attribute:* reportingEndDate (common:TimePeriodType) -
2457 optional

2458 *Attribute:* validFromDate (common:TimePeriodType) - optional



2459 *Attribute:* validToDate (common:TimePeriodType) - optional

2460 *Attribute:* publicationYear (xs:gYear) - optional

2461 *Attribute:* publicationPeriod (common:TimePeriodType) -
2462 optional

2463 **GroupType:** The key values at the group level may be stated explicitly, and
2464 all which are not wildcarded listed in GroupKey - they must also all be given a
2465 value at the series level. It is not necessary to specify the group key, however,
2466 as this may be inferred from the values repeated at the series level. If only
2467 documentation (group-level attributes) are being transmitted, however, the
2468 GroupKey cannot be omitted. The type attribute contains the name of the
2469 declared group in the key family. If any group-level attributes are specified in a
2470 delete message, then any valid value supplied for the attribute indicates that
2471 the current attribute value should be deleted for the specified attribute.

2472 *Element Content (Type):*

2473
2474 GroupKey (ValuesType) - min. 0
2475 Attributes (ValuesType) - min. 0
2476 Series (SeriesType) - min. 0 - max. unbounded
2477 Annotations (common:AnnotationsType) - min. 0

2478 *Attribute:* type (xs:NMTOKEN) - required

2479 **SeriesType:** SeriesType specifies the structure of a series. This includes all
2480 of the key values, values for all the attributes, and the set of observations
2481 making up the series content. Messages may transmit only attributes, only
2482 data, or both. Regardless, the series key is always required. Key values
2483 appear at the Series level in an ordered sequence which corresponds to the
2484 key sequence in the key family. A series in a delete message need not supply
2485 more than the key, indicating that the entire series identified by that key
2486 should be deleted. If series attributes are sent in a delete message, any valid
2487 value specified for an attribute indicates that the attribute should be deleted.

2488 *Element Content (Type):*

2489
2490 SeriesKey (SeriesKeyType)
2491 Attributes (ValuesType) - min. 0
2492 Obs (ObsType) - min. 0 - max. unbounded
2493 Annotations (common:AnnotationsType) - min. 0

2494 **SeriesKeyType:** SeriesKeyType defines the contents of a series key. Each
2495 non-time dimension must have a value supplied for it, in the order in which the
2496 dimensions are specified in the key family.

2497 *Element Content (Type):*

2498
2499 Value (ValueType) - max. unbounded

2500 **ObsType:** ObsType defines the structure of an observation. This includes a
2501 time and observation value, as well as values for each of the attributes
2502 assigned at the observation level by the key family. In a delete message, only
2503 the time need be given, indicating that the observation identified by the key
2504 and time should be deleted. For an update message, both time and
2505 observation value are required. If any attributes appear in a delete message,
2506 any valid value supplied for an attribute indicates that the current value should
2507 be deleted.

2508 *Element Content (Type):*

2509
2510 Time (common:TimePeriodType)
2511 ObsValue (ObsValueType) - min. 0
2512 Attributes (ValueType) - min. 0
2513 Annotations (common:AnnotationsType) - min. 0

2514 **ValueType:**

2515 *Element Content (Type):*

2516
2517 Value (ValueType) - max. unbounded

2518 **ValueType:** ValueType is used to assign a single value to a concept, as for
2519 attribute values and key values. It has no element content. The startTime
2520 attribute is only used if the textFormat of the attribute is of the Timespan type
2521 in the key family (in which case the value field takes a duration).

2522 *Attribute:* concept (common:IDType)

2523 *Attribute:* value (xs:string)

2524 *Attribute:* startTime (xs:dateTime) - optional

2525 **ObsValueType:** ObsValueType describes the information set for an
2526 observation value. This is associated with the primary measure concept
2527 declared in the key family. The startTime attribute is only used if the
2528 textFormat of the observation is of the Timespan type in the key family (in
2529 which case the value field takes a duration).

2530 *Attribute:* value (xs:double)

2531 *Attribute:* startTime (xs:dateTime) - optional

2532



2533

2534 **5.4 SDMX Generic Metadata Namespace Module**

2535

2536 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/genericmetadata**2537 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
2538 (SDMXCommon.xsd)

2539

2540 **5.4.1 Global Elements**2541 **MetadataSet(MetadataSetType):**

2542

2543 **5.4.2 Complex Types**

2544 **MetadataSetType:** The Metadata Set is a set of reported metadata against a
2545 set of values for a given full or partial target identifier, as described in a
2546 metadata structure definition. Child elements include identification of the
2547 relevant metadata structure definition using the MetadataStructureRef and
2548 MetadataStructureAgencyRef elements. The ReportRef element includes the
2549 ID of the report structure as described in the metadata structure definition.
2550 AttributeValueSet is a repeatable child element which allows target identifier
2551 keys and their associated metadata attribute values to be reported (this
2552 functions like a series element does for data sets). An optional name and
2553 annotations may also be supplied. The metadataStructureURI allows for a
2554 URI to be provided, pointing to the SDMX-ML Structure Message
2555 representation of the referenced metadata structure definition. Attributes are
2556 provided for describing the contents of a data or metadata set, which are
2557 particularly important for interactions with the SDMX Registry: datasetID,
2558 dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID,
2559 and dataflowID all take the IDs specified by the attribute names. The action
2560 attribute indicates whether the file is appending, replacing, or deleting.
2561 Attributes reportingBeginDate, reportingEndDate, validFromDate, and
2562 validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year,
2563 and publicationPeriod specifies the period of publication of the data in terms of
2564 whatever provisioning agreements might be in force (ie, "Q1 2005" if that is
2565 the time of publication for a data set published on a quarterly basis).

2566 *Element Content (Type):*

2567

2568 Name (common:TextType) - min. 0 - max. unbounded

2569 MetadataStructureRef (common:IDType)

2570 MetadataStructureAgencyRef (common:IDType)

2571 ReportRef (common:IDType)

2572 AttributeValueSet (AttributeValueSetType) - max. unbounded

2573 Annotations (common:AnnotationsType) - min. 0



2574	<i>Attribute:</i> metadataStructureURI (xs:anyURI) - optional
2575	<i>Attribute:</i> datasetID (common:IDType) - optional
2576 2577	<i>Attribute:</i> dataProviderSchemeAgencyId (common:IDType) - optional
2578	<i>Attribute:</i> dataProviderSchemeId (common:IDType) - optional
2579	<i>Attribute:</i> dataProviderID (common:IDType) - optional
2580	<i>Attribute:</i> dataflowAgencyID (common:IDType) - optional
2581	<i>Attribute:</i> dataflowID (common:IDType) - optional
2582	<i>Attribute:</i> action (common:ActionType) - optional
2583 2584	<i>Attribute:</i> reportingBeginDate (common:TimePeriodType) - optional
2585 2586	<i>Attribute:</i> reportingEndDate (common:TimePeriodType) - optional
2587	<i>Attribute:</i> validFromDate (common:TimePeriodType) - optional
2588	<i>Attribute:</i> validToDate (common:TimePeriodType) - optional
2589	<i>Attribute:</i> publicationYear (xs:gYear) - optional
2590 2591	<i>Attribute:</i> publicationPeriod (common:TimePeriodType) - optional
2592	AttributeValueSetType: The attribute value set provides the values for a set
2593	of metadata attributes reported against a target identifier key. The TargetRef
2594	element contains the value of the metadata attribute's target attribute in the
2595	metadata structure definition (that is, the ID of the full or partial target identifier
2596	which is the target of the metadata report). TargetValues is an element
2597	substructure which provides the specific full or partial target identifier
2598	component values, and the ReportedAttribute sub-element allows for values
2599	to be reported against the metadata attributes as described in the referenced
2600	metadata structure definition for the referenced full or partial targets.
2601	<i>Element Content (Type):</i>
2602	
2603	TargetRef (common:IDType)
2604	TargetValues (TargetValuesType)
2605	ReportedAttribute (ReportedAttributeType) - max. unbounded



2606 **TargetValueType:** Target values contains the specific values for each
2607 concept in the full or partial target identifier as described in a metadata
2608 structure definition. These values typically come from codelists or other item
2609 schemes. Each such value should be presented in the order given in the
2610 metadata structure definition, and must use a valid representation for that
2611 concept. Concepts are those referenced by the identifier components of the
2612 target identifiers.

2613 *Element Content (Type):*

2614
2615 ComponentValue (ComponentValueType) - max. unbounded

2616 **ComponentValueType:** Component values have an object attribute with an
2617 object type value as provided in the metadata structure definition, a
2618 component attribute which takes the ID of the identifier component in the
2619 metadata structure definition's full target identifier, and a value, which must be
2620 a valid value for that concept's representation as described in the metadata
2621 structure definition.

2622
2623 [data] (xs:NMTOKEN)

2624 **ReportedAttributeType:** Reported attributes hold the values which are to be
2625 reported against the target specified in the metadata structure definition, and
2626 according to the metadata attributes specified for the target referenced in the
2627 TargetRef element. Each reported attribute may have Value sub-elements
2628 (one per language) if it takes a text or numeric value. The StartTime element
2629 is only used if the attribute being represented is of the Timespan type (as
2630 described in the corresponding TextFormat element in the metadata structure
2631 definition). In this case, the Value takes a duration. Only one such value is
2632 allowed in the ReportedAttribute in this case. The types of these values must
2633 conform to the limitations described in the metadata structure definition. Also -
2634 if permitted by the metadata structure definition - there may be one or more
2635 child ReportedAttribute elements. These must be arranged in the nesting
2636 hierarchy given in the metadata structure definition. The conceptID attribute
2637 provides the id of the concept given in the metadata structure definition to
2638 which the reported attribute corresponds.

2639 *Element Content (Type):*

2640
2641 Value (common:TextType) - min. 0 - max. unbounded
2642 StartTime (xs:dateTime) - min. 0
2643 ReportedAttribute (ReportedAttributeType) - min. 0 - max. unbounded
2644 Annotations (common:AnnotationsType) - min. 0

2645 *Attribute:* conceptID (common:IDType) - required

2646

2647 **5.4.3 Simple Types**

2648 **ObjectIDType:** The Object ID is used to reference a particular Object within
2649 the SDMX Information Model's formalization of statistical exchanges.

2650 *Restricts xs:NMTOKEN*

2651 Code: Agency - Agency

2652 Code: ConceptScheme - Concept scheme

2653 Code: Concept - Concept

2654 Code: Codelist - Codelist

2655 Code: Code - Code

2656 Code: KeyFamily - Key family

2657 Code: Component - Component

2658 Code: KeyDescriptor - Key descriptor

2659 Code: MeasureDescriptor - Measure descriptor

2660 Code: AttributeDescriptor - Attribute descriptor

2661 Code: GroupKeyDescriptor - Group key descriptor

2662 Code: Dimension - Dimension

2663 Code: Measure - Measure

2664 Code: Attribute - Attribute

2665 Code: CategoryScheme - Category scheme

2666 Code: ReportingTaxonomy - Reporting taxonomy

2667 Code: Category - Category

2668 Code: OrganisationScheme - Organisation scheme

2669 Code: DataProvider - Data or metadata provider

2670 Code: MetadataStructure - Metadata structure definition

2671 Code: FullTargetIdentifier - Full target identifier

2672 Code: PartialTargetIdentifier - Partial target identifier

2673 Code: MetadataAttribute - Metadata attribute

2674 Code: DataFlow - Data flow



- 2675 Code: ProvisionAgreement - Data or metadata provision agreement
- 2676 Code: MetadataFlow - Metadata flow
- 2677 Code: ContentConstraint - Content constraint
- 2678 Code: AttachmentConstraint - Attachment constraint
- 2679 Code: DataSet - Data set
- 2680 Code: XSDataSet - Cross-sectional data set
- 2681 Code: MetadataSet - Metadata set
- 2682 Code: HierarchicalCodelist - Hierarchical codelist
- 2683 Code: Hierarchy - Hierarchy
- 2684 Code: StructureSet - Structure set
- 2685 Code: StructureMap - Structure map
- 2686 Code: ComponentMap - Component map
- 2687 Code: CodelistMap - Codelist map
- 2688 Code: CodeMap - Code map
- 2689 Code: CategorySchemeMap - Category scheme map
- 2690 Code: CategoryMap - Category map
- 2691 Code: OrganisationSchemeMap - Organisation scheme map
- 2692 Code: OrganisationRoleMap - Organisation role map
- 2693 Code: ConceptSchemeMap - Concept scheme map
- 2694 Code: ConceptMap - Concept map
- 2695 Code: Process - Process
- 2696 Code: ProcessStep - Process step

2697

2698

2699 **5.5 SDMX Query Namespace Module**

2700 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/query**

2701 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
2702 (SDMXCommon.xsd)

2703

2704 **5.5.1 Global Elements**

2705 **Query(QueryType):** The Query message allows standard querying of SDMX-
2706 compliant databases and web services. It is intended to be used in non-
2707 registry exchanges, and is focused on data sets and metadata sets. It allows
2708 queries to retrieve data, metadata, key families, metadata structure
2709 definitions, codelists, concepts, and other structural metadata. Note that date
2710 and time formats are structured according to the common:TimePeriodType,
2711 rather than being specified in the query. The response documents for this
2712 query message are data formats (for data queries), metadata formats (for
2713 metadata queries), and the SDMX Structure MMessage (for all other queries).

2714

2715 **5.5.2 Complex Types**

2716 **QueryType:** The Query element is a top-level element for this namespace,
2717 which is referenced by the SDMX message envelope, or could be put inside
2718 another envelope, such as SOAP. It contains a query. The defaultLimit
2719 attribute is the suggested maximum response size in kilobytes.

2720 *Element Content (Type):*

2721

2722 DataWhere (DataWhereType) - min. 0 - max. unbounded
2723 MetadataWhere (MetadataWhereType) - min. 0 - max. unbounded
2724 KeyFamilyWhere (KeyFamilyWhereType) - min. 0 - max. unbounded
2725 MetadataStructureWhere (MetadataStructureWhereType) - min. 0 - max.
2726 unbounded
2727 CodelistWhere (CodelistWhereType) - min. 0 - max. unbounded
2728 ConceptWhere (ConceptWhereType) - min. 0 - max. unbounded
2729 AgencyWhere (AgencyWhereType) - min. 0 - max. unbounded
2730 DataProviderWhere (DataProviderWhereType) - min. 0 - max. unbounded
2731 HierarchicalCodelistWhere (HierarchicalCodelistWhereType) - min. 0 - max.
2732 unbounded
2733 ReportingTaxonomyWhere (ReportingTaxonomyWhereType) - min. 0 - max.
2734 unbounded
2735 DataflowWhere (DataflowWhereType) - min. 0 - max. unbounded
2736 MetadataflowWhere (MetadataflowWhereType) - min. 0 - max. unbounded
2737 StructureSetWhere (StructureSetWhereType) - min. 0 - max. unbounded
2738 ProcessWhere (ProcessWhereType) - min. 0 - max. unbounded
2739 OrganisationSchemeWhere (OrganisationSchemeWhereType) - min. 0 -
2740 max. unbounded
2741 ConceptSchemeWhere (ConceptSchemeWhereType) - min. 0 - max.
2742 unbounded
2743 CategorySchemeWhere (CategorySchemeWhereType) - min. 0 - max.
2744 unbounded

2745 *Attribute:* defaultLimit (xs:integer) - optional

2746 **DataWhereType:** The DataWhere element represents a query for data. It
2747 contains all of the clauses in that query, represented by its child elements.
2748 Values are the IDs of the referenced object.



2749 *Element Content (Type):*

2750 (Choice)
2751 DataSet (xs:string) [Choice]
2752 KeyFamily (xs:string) [Choice]
2753 Dimension (DimensionType) [Choice]
2754 Attribute (AttributeType) [Choice]
2755 Codelist (CodelistType) [Choice]
2756 Time (TimeType) [Choice]
2757 Category (CategoryType) [Choice]
2758 Concept (xs:string) [Choice]
2759 DataProvider (xs:string) [Choice]
2760 Dataflow (xs:string) [Choice]
2761 Version (xs:string) [Choice]
2762 Or (OrType) [Choice]
2763 And (AndType) [Choice]

2764 **MetadataWhereType:** The MetadataWhere element represents a query for
2765 metadata. It contains all of the clauses in that query, represented by its child
2766 elements. Values are the IDs of the referenced object.

2767 *Element Content (Type):*

2768 (Choice)
2769 MetadataSet (xs:string) [Choice]
2770 MetadataStructure (xs:string) [Choice]
2771 StructureComponent (StructureComponentType) [Choice]
2772 Attribute (AttributeType) [Choice]
2773 Codelist (CodelistType) [Choice]
2774 Time (TimeType) [Choice]
2775 Category (CategoryType) [Choice]
2776 Concept (xs:string) [Choice]
2777 DataProvider (xs:string) [Choice]
2778 Metadataflow (xs:string) [Choice]
2779 Version (xs:string) [Choice]
2780 Or (OrType) [Choice]
2781 And (AndType) [Choice]

2782 **AndType:** For the And element, each of its immediate child elements
2783 represent clauses all of which represent conditions which must be satisfied. If
2784 children are A, B, and C, then any legitimate response will meet conditions A,
2785 B, and C. Values are the IDs of the referenced object.

2786 *Element Content (Type):*

2787
2788 DataSet (xs:string) - min. 0 - max. unbounded
2789 MetadataSet (xs:string) - min. 0 - max. unbounded
2790 KeyFamily (xs:string) - min. 0 - max. unbounded
2791 MetadataStructure (xs:string) - min. 0 - max. unbounded
2792 Dimension (DimensionType) - min. 0 - max. unbounded
2793 StructureComponent (StructureComponentType) - min. 0 - max. unbounded
2794 Attribute (AttributeType) - min. 0 - max. unbounded
2795 Codelist (CodelistType) - min. 0 - max. unbounded
2796 Time (TimeType) - min. 0 - max. unbounded



2797 Category (CategoryType) - min. 0 - max. unbounded
2798 Concept (xs:string) - min. 0 - max. unbounded
2799 AgencyID (xs:string) - min. 0 - max. unbounded
2800 DataProvider (xs:string) - min. 0 - max. unbounded
2801 Dataflow (xs:string) - min. 0 - max. unbounded
2802 Metadataflow (xs:string) - min. 0 - max. unbounded
2803 Version (xs:string) - min. 0 - max. unbounded
2804 Or (OrType) - min. 0 - max. unbounded
2805 And (AndType) - min. 0 - max. unbounded

2806 **OrType:** The Or element's immediate children represent clauses in the query
2807 any one of which is sufficient to satisfy the query. If these children are A, B,
2808 and C, then any result which meets condition A, or condition B, or condition C
2809 is a match for that query. Values are the IDs of the referenced object.

2810 *Element Content (Type):*

2811
2812 DataSet (xs:string) - min. 0 - max. unbounded
2813 MetadataSet (xs:string) - min. 0 - max. unbounded
2814 KeyFamily (xs:string) - min. 0 - max. unbounded
2815 MetadataStructure (xs:string) - min. 0 - max. unbounded
2816 Dimension (DimensionType) - min. 0 - max. unbounded
2817 StructureComponent (StructureComponentType) - min. 0 - max. unbounded
2818 Attribute (AttributeType) - min. 0 - max. unbounded
2819 Codelist (CodelistType) - min. 0 - max. unbounded
2820 Time (TimeType) - min. 0 - max. unbounded
2821 Category (CategoryType) - min. 0 - max. unbounded
2822 Concept (xs:string) - min. 0 - max. unbounded
2823 AgencyID (xs:string) - min. 0 - max. unbounded
2824 DataProvider (xs:string) - min. 0 - max. unbounded
2825 Dataflow (xs:string) - min. 0 - max. unbounded
2826 Metadataflow (xs:string) - min. 0 - max. unbounded
2827 Version (xs:string) - min. 0 - max. unbounded
2828 Or (OrType) - min. 0 - max. unbounded
2829 And (AndType) - min. 0 - max. unbounded

2830 **DimensionType:** Dimension elements contain the (single) value being
2831 searched on within the key of the data set. The id attribute holds the ID of the
2832 dimension. If the content is empty, then the query is for any dimension with
2833 the given name. If the name attribute is not supplied, then the query is for the
2834 given key value within any dimension.

2835
2836 [data] (xs:string)

2837 **StructureComponentType:** StructureComponent elements contain the
2838 (single) value being searched on within the key of data set, but this value can
2839 be either a code value or the alias assigned to a set of equivalent code
2840 values. The id attribute holds the ID of the dimension, attribute, or alias
2841 assigned to a component in a structure set. If the content is empty, then the
2842 query is for any component with the given name or alias. If the name attribute



2843 is not supplied, then the query is for the given code value or alias within any
2844 component or component alias.

2845
2846 [data] (xs:string)

2847 **AttributeType:** Attribute elements contain the (single) value of an attribute
2848 being queried for. The id attribute contains the id of the attribute. The
2849 attachmentLevel attribute specifies the attachment level of the attribute. If the
2850 content of Attribute is empty, then the search is for the specified attribute (and
2851 attachment level). If the name attribute is not specified, then the search is on
2852 any attribute. If the attachmentLevel attribute is not specified, then the query is
2853 for an attribute at any attachment level, as the value defaults to "Any".

2854
2855 [data] (xs:string)

2856 **CodelistType:** The Codelist element allows queries to specify a (single) value
2857 found within a codelist as the element content, and the agency-qualified name
2858 of the codelist being queried for in the name attribute. If no content is
2859 supplied, then the query is for the named codelist. If the id attribute is left
2860 empty, then the value is searched for in any codelist.

2861
2862 [data] (xs:string)

2863 **CategoryType:** The Category element allows for a search to be made on the
2864 values within a specific category, which is specified (in agency-qualified form)
2865 with the name attribute. If there is no element content, then the search is for
2866 the named Category; if the name is not supplied, then the category value
2867 supplied as content should be sought-for in all available categories.

2868
2869 [data] (xs:string)

2870 **KeyFamilyWhereType:** The KeyFamilyWhere element represents a query
2871 for a key family or key families. It contains all of the clauses in that query,
2872 represented by its child elements. Values are the IDs of the referenced object.

2873 *Element Content (Type):*

2874 (Choice)
2875 KeyFamily (xs:string) [Choice]
2876 Dimension (DimensionType) [Choice]
2877 Attribute (AttributeType) [Choice]
2878 Codelist (CodelistType) [Choice]
2879 Category (CategoryType) [Choice]
2880 Concept (xs:string) [Choice]
2881 AgencyID (xs:string) [Choice]
2882 Version (xs:string) [Choice]

2883 Or (OrType) [Choice]
 2884 And (AndType) [Choice]

2885 **MetadataStructureWhereType:** The MetadataStructureWhere element
 2886 represents a query for a metadata structure or structures. It contains all of
 2887 the clauses in that query, represented by its child elements. Values are the
 2888 IDs of the referenced object.

2889 *Element Content (Type):*

2890 (Choice)
 2891 KeyFamily (xs:string) [Choice]
 2892 MetadataStructure (xs:string) [Choice]
 2893 StructureSet (xs:string) [Choice]
 2894 Dimension (DimensionType) [Choice]
 2895 StructureComponent (StructureComponentType) [Choice]
 2896 Attribute (AttributeType) [Choice]
 2897 Codelist (CodelistType) [Choice]
 2898 Category (CategoryType) [Choice]
 2899 Concept (xs:string) [Choice]
 2900 AgencyID (xs:string) [Choice]
 2901 Version (xs:string) [Choice]
 2902 Or (OrType) [Choice]
 2903 And (AndType) [Choice]

2904 **CodelistWhereType:** The CodelistWhere element represents a query for a
 2905 codelist or codelists. It contains all of the clauses in that query, represented by
 2906 its child elements. Values are the IDs of the referenced object.

2907 *Element Content (Type):*

2908 (Choice)
 2909 Codelist (CodelistType) [Choice]
 2910 AgencyID (xs:string) [Choice]
 2911 Version (xs:string) [Choice]
 2912 Or (OrType) [Choice]
 2913 And (AndType) [Choice]

2914 **ConceptWhereType:** The ConceptWhere element represents a query for a
 2915 concept or concepts. It contains all of the clauses in that query, represented
 2916 by its child elements. Values are the IDs of the referenced object.

2917 *Element Content (Type):*

2918 (Choice)
 2919 Concept (xs:string) [Choice]
 2920 AgencyID (xs:string) [Choice]
 2921 Version (xs:string) [Choice]
 2922 Or (OrType) [Choice]
 2923 And (AndType) [Choice]



2924 **AgencyWhereType:** The AgencyWhere element represents a query for
2925 details for an Agency. It contains all of the clauses in that query, represented
2926 by its child elements. Values are the IDs of the referenced object.

2927 *Element Content (Type):*

2928 (Choice)
2929 KeyFamily (xs:string) [Choice] - min. 0 - max. unbounded
2930 MetadataStructure (xs:string) [Choice] - min. 0 - max. unbounded
2931 StructureSet (xs:string) [Choice] - min. 0 - max. unbounded
2932 Codelist (CodelistType) [Choice] - min. 0 - max. unbounded
2933 Category (CategoryType) [Choice] - min. 0 - max. unbounded
2934 Concept (xs:string) [Choice] - min. 0 - max. unbounded
2935 AgencyID (xs:string) [Choice] - min. 0 - max. unbounded
2936 Or (OrType) [Choice] - min. 0 - max. unbounded
2937 And (AndType) [Choice] - min. 0 - max. unbounded

2938 **DataProviderWhereType:** The DataProviderWhere element represents a
2939 query for details for a provider of data or metadata sets. It contains all of the
2940 clauses in that query, represented by its child elements. Values are the IDs of
2941 the referenced object.

2942 *Element Content (Type):*

2943 (Choice)
2944 DataSet (xs:string) [Choice] - min. 0 - max. unbounded
2945 MetadataSet (xs:string) [Choice] - min. 0 - max. unbounded
2946 KeyFamily (xs:string) [Choice] - min. 0 - max. unbounded
2947 MetadataStructure (xs:string) [Choice] - min. 0 - max. unbounded
2948 StructureSet (xs:string) [Choice] - min. 0 - max. unbounded
2949 Codelist (CodelistType) [Choice] - min. 0 - max. unbounded
2950 Category (CategoryType) [Choice] - min. 0 - max. unbounded
2951 Concept (xs:string) [Choice] - min. 0 - max. unbounded
2952 AgencyID (xs:string) [Choice] - min. 0 - max. unbounded
2953 Or (OrType) [Choice] - min. 0 - max. unbounded
2954 And (AndType) [Choice] - min. 0 - max. unbounded

2955 **TimeType:** TimeType contains the time point or period for which results
2956 should be supplied. When StartTime and EndTime are used, these must be
2957 understood as inclusive.

2958 *Element Content (Type):*

2959 (Choice)
2960 Time (common:TimePeriodType) [Choice]

2961 **StructureSetWhereType:** The StructureSetWhere element represents a
2962 query for a structure set or structure sets. Like other maintainable objects, it
2963 must be queried for using information about its agency, ID, and/or version.
2964 Any field not supplied will be taken as matching all of that type.

2965 *Element Content (Type):*



2966
2967 AgencyID (xs:string) - min. 0
2968 ID (xs:string) - min. 0
2969 Version (xs:string) - min. 0

2970 **HierarchicalCodelistWhereType:** The HierarchicalCodelistWhere element
2971 represents a query for a hierarchical codelist or codelists. Like other
2972 maintainable objects, it must be queried for using information about its
2973 agency, ID, and/or version. Any field not supplied will be taken as matching all
2974 of that type.

2975 *Element Content (Type):*

2976
2977 AgencyID (xs:string) - min. 0
2978 ID (xs:string) - min. 0
2979 Version (xs:string) - min. 0

2980 **ReportingTaxonomyWhereType:** The ReportingTaxonomyWhere element
2981 represents a query for a reporting taxonomy or taxonomies. Like other
2982 maintainable objects, it must be queried for using information about its
2983 agency, ID, and/or version. Any field not supplied will be taken as matching all
2984 of that type.

2985 *Element Content (Type):*

2986
2987 AgencyID (xs:string) - min. 0
2988 ID (xs:string) - min. 0
2989 Version (xs:string) - min. 0

2990 **DataflowWhereType:** The DataflowWhereType element represents a query
2991 for a dataflow or dataflows. Like other maintainable objects, it must be queried
2992 for using information about its agency, ID, and/or version. Any field not
2993 supplied will be taken as matching all of that type.

2994 *Element Content (Type):*

2995
2996 AgencyID (xs:string) - min. 0
2997 ID (xs:string) - min. 0
2998 Version (xs:string) - min. 0

2999 **MetadataflowWhereType:** The MetadataflowWhereType element represents
3000 a query for a metadataflow or metadataflows. Like other maintainable objects,
3001 it must be queried for using information about its agency, ID, and/or version.
3002 Any field not supplied will be taken as matching all of that type.

3003 *Element Content (Type):*

3004
3005 AgencyID (xs:string) - min. 0



3006 ID (xs:string) - min. 0
3007 Version (xs:string) - min. 0

3008 **ProcessWhereType:** The ProcessWhere element represents a query for a
3009 process or processes. Like other maintainable objects, it must be queried for
3010 using information about its agency, ID, and/or version. Any field not supplied
3011 will be taken as matching all of that type.

3012 *Element Content (Type):*

3013
3014 AgencyID (xs:string) - min. 0
3015 ID (xs:string) - min. 0
3016 Version (xs:string) - min. 0

3017 **OrganisationSchemeWhereType:** The OrganisationSchemeWhere element
3018 represents a query for an organisation scheme or schemes. Like other
3019 maintainable objects, it must be queried for using information about its
3020 agency, ID, and/or version. Any field not supplied will be taken as matching all
3021 of that type.

3022 *Element Content (Type):*

3023
3024 AgencyID (xs:string) - min. 0
3025 ID (xs:string) - min. 0
3026 Version (xs:string) - min. 0

3027 **ConceptSchemeWhereType:** The ConceptSchemeWhere element
3028 represents a query for a concept scheme or schemes. Like other maintainable
3029 objects, it must be queried for using information about its agency, ID, and/or
3030 version. Any field not supplied will be taken as matching all of that type.

3031 *Element Content (Type):*

3032
3033 AgencyID (xs:string) - min. 0
3034 ID (xs:string) - min. 0
3035 Version (xs:string) - min. 0

3036 **CategorySchemeWhereType:** The CategorySchemeWhere element
3037 represents a query for a category scheme or schemes. Like other
3038 maintainable objects, it must be queried for using information about its
3039 agency, ID, and/or version. Any field not supplied will be taken as matching all
3040 of that type.

3041 *Element Content (Type):*

3042
3043 AgencyID (xs:string) - min. 0
3044 ID (xs:string) - min. 0
3045 Version (xs:string) - min. 0

3046

3047 **5.5.3 Simple Types**

3048 **AttachmentLevelType:** This type supplies an enumeration of attachment
3049 levels corresponding to those in the SDMX Information Model, plus a value of
3050 "Any" where the search is wildcarded.

3051 *Restricts xs:NMTOKEN*

3052 Code: DataSet - Attached at the Data Set level

3053 Code: Group - Attached at the Group level

3054 Code: Series - Attached at the Series level

3055 Code: Observation - Attached at the Observation level

3056 Code: Any - Attached at any attachment level

3057

3058 **5.6 SDMX Common Namespace Module**

3059

3060 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common**

3061 *Imports:* <http://www.w3.org/XML/1998/namespace> (xml.xsd)

3062

3063 **5.6.1 Complex Types**

3064 **ConstraintType:** Constraint specifies the object to which constraints are
3065 attached. Note that if the constraint is that for a Data Provider, then only
3066 ReleaseCalendar information is relevant, as there is no reliable way of
3067 determining which key family is being used to frame constraints in terms of
3068 cube regions or key sets. ReferencePeriod is used to report start date and
3069 end date constraints. MetadataConceptSet allows for content constraints to
3070 be described for metadata sets.

3071 *Element Content (Type):*

3072

3073 ConstraintID (IDType)

3074 CubeRegion (CubeRegionType) - min. 0 - max. unbounded

3075 MetadataConceptSet (MetadataConceptSetType) - min. 0

3076 KeySet (KeySetType) - min. 0 - max. unbounded

3077 ReleaseCalendar (ReleaseCalendarType) - min. 0

3078 ReferencePeriod (ReferencePeriodType) - min. 0

3079 *Attribute:* ConstraintType (ConstraintTypeType) - required



3080 **CubeRegionType:** CubeRegion describes the portion(s) of the possible
3081 combinations of all components within a key family or metadata structure
3082 definition by providing valid values on a per-component basis. This does not
3083 guarantee that data will be available for all possible combinations, but
3084 describes the portion of the cube in which it is useful to query for data. The
3085 isIncluded attribute, if true, indicates that the described area is the one in
3086 which it is useful to search/expect to find data. If false, this means that the
3087 portions of the cube outside the described region are useful to search/where
3088 you may expect to find data.

3089 *Element Content (Type):*

3090
3091 Member (MemberType) - max. unbounded

3092 *Attribute:* isIncluded (xs:boolean) - required

3093 **MetadataConceptSetType:** The isIncluded attribute, if true, indicates that the
3094 described concepts - of those described as possibilities in the relevant
3095 metadata structure definition - are reported. If the value is false, then the
3096 specified concepts are not reported.

3097 *Element Content (Type):*

3098
3099 Member (MemberType) - max. unbounded

3100 *Attribute:* isIncluded (xs:boolean) - required

3101 **MemberType:** Member describes the constrained component - which can be
3102 a dimension, an attribute, a metadata attribute, or a measure. This must agree
3103 with the metadata structure definition or key family referenced by the
3104 Provision Agreement's Dataflow or Metadataflow. The isIncluded attribute
3105 indicates whether the Member is listing included or excluded values for each
3106 component, as seen against the full valid set described in the key family.
3107 When used to describe reported metadata, the MemberValue may be omitted
3108 in cases where no specification is made regarding the representation of the
3109 concept (as is the case with un-coded metadata attributes). Otherwise,
3110 MemberValue must be included.

3111 *Element Content (Type):*

3112
3113 ComponentRef (IDType)
3114 MemberValue (MemberValueType) - min. 0 - max. unbounded

3115 *Attribute:* isIncluded (xs:boolean) - required



- 3116 **MemberValueType:** MemberValue specifies the value of the specified
3117 component, which must be a valid value as described in the appropriate
3118 structure definition (key family).
- 3119 *Element Content (Type):*
- 3120
3121 Value (xs:string)
- 3122 **KeySetType:** KeySet describes a set of keys. The isIncluded attribute, if true,
3123 indicates that the specified keys are valid keys within the constraint. If false,
3124 the set of keys described are not valid - all other possible keys are the valid
3125 ones.
- 3126 *Element Content (Type):*
- 3127
3128 Key (KeyType)
- 3129 *Attribute:* isIncluded (xs:boolean) - required
- 3130 **KeyType:** Key allows for sets of component references - holding the name of
3131 the component's concept - and a permitted value for that component. This
3132 construct can be repeated as many times as desired, but must describe
3133 complete keys according to the relevant structure definition (key family).
- 3134 *Element Content (Type):*
- 3135
3136 ComponentRef (IDType)
3137 Value (xs:string)
- 3138 **ReleaseCalendarType:** The ReleaseCalendar holds information about the
3139 timing of releases of the constrained data. Periodicity is the period between
3140 releases of the data set. Offset is the interval between January first and the
3141 first release of data within the year. Tolerance is the period after which the
3142 release of data may be deemed late. All of these values use the standard
3143 "P7D"-style format.
- 3144 *Element Content (Type):*
- 3145
3146 Periodicity (xs:string)
3147 Offset (xs:string)
3148 Tolerance (xs:string)
- 3149 **ReferencePeriodType:** Specifies the inclusive start and end times for a
3150 registry query.
- 3151 *Attribute:* startTime (xs:dateTime) - required

3152 *Attribute:* endTime (xs:dateTime) - required

3153 **TextType:** TextType provides for a set of language-specific alternates to be
3154 provided for any human-readable construct in the instance.

3155
3156 [data] (xs:string)

3157 **AnnotationType:** AnnotationType provides for non-documentation notes and
3158 annotations to be embedded in data and structure messages. It provides
3159 optional fields for providing a title, a type description, a URI, and the text of the
3160 annotation.

3161 *Element Content (Type):*

3162
3163 AnnotationTitle (xs:string) - min. 0
3164 AnnotationType (xs:string) - min. 0
3165 AnnotationURL (xs:anyURI) - min. 0
3166 AnnotationText (TextType) - min. 0 - max. unbounded

3167 **AnnotationsType:** AnnotationsType provides for a list of annotations to be
3168 attached to data and structure messages.

3169 *Element Content (Type):*

3170
3171 Annotation (AnnotationType) - max. unbounded

3172

3173 5.6.2 Simple Types

3174 **ConstraintTypeType:** ConstraintType provides an enumeration of values of
3175 the types of constraints.

3176 *Restricts* xs:NMTOKEN

3177 Code: Content - Content constraint.

3178 Code: Attachment - Attachment constraint.

3179 **PeriodType:** PeriodType provides a list of tokens for specifying common
3180 periods: Quarterly: Q1, Q2, Q3, Q4; Weekly: W1 - W52; Triannual: T1, T2, T3;
3181 Biannual: B1, B2. These values appear after a four-digit year indicator,
3182 followed by a dash (ie, 2005-Q1).

3183 *Restricts* xs:string

3184 **TimePeriodType:** TIME_PERIOD is not completely expressible in XML
3185 Schema's date type: instead we use the union of dateTime, date,

3186 gYearMonth, and gYear. The default name for the concept is TIME_PERIOD.
3187 Bi-annual, tri-annual, quarterly, and weekly periods have special formats (see
3188 PeriodType, above), but other periods would be described in terms of their
3189 beginning date or time (eg, a period of a decade is identified with a four-digit
3190 year corresponding to the decades' first year).

3191 **ActionType:** ActionType provides a list of actions, describing the intention of
3192 the data transmission from the sender's side. Each action provided at the
3193 dataset or metadataset level applies to the entire dataset for which it is given.
3194 Note that the actions indicated in the Message Header are optional, and used
3195 to summarize specific actions indicated with this data type for all registry
3196 interactions. The "Informational" value is used when the message contains
3197 information in response to a query, rather than being used to invoke a
3198 maintenance activity.

3199 *Restricts xs:NMTOKEN*

3200 Code: Append - Data or metadata is an incremental update for an existing
3201 data/metadata set or the provision of new data or documentation (attribute values)
3202 formerly absent. If any of the supplied data or metadata is already present, it will not
3203 replace that data or metadata. This corresponds to the "Update" value found in
3204 version 1.0 of the SDMX Technical Standards.

3205 Code: Replace - Data/metadata is to be replaced, and may also include additional
3206 data/metadata to be appended.

3207 Code: Delete - Data/Metadata is to be deleted.

3208 Code: Information - Informational

3209 **IDType:** IDType provides a type which is used for restricting the characters in
3210 codes and IDs throughout all SDMX-ML messages. Valid characters include
3211 A-Z, a-z, @, 0-9, _, -, \$.

3212 *Restricts xs:string*

3213

3214 **5.7 SDMX Registry Interfaces Namespace Module**

3215

3216 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/registry**

3217 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
3218 (SDMXCommon.xsd)

3219 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/structure
3220 (SDMXStructure.xsd)

3221

3222 **5.7.1 Complex Types**

3223 **SubmitSubscriptionRequestType:** The SubmitSubscriptionRequest element
3224 is submitted to the registry to subscribe to registration and change events for
3225 specific registry resources.

3226 *Element Content (Type):*

3227
3228 Subscription (SubscriptionType) - max. unbounded

3229 **SubmitSubscriptionResponseType:** The SubmitSubscriptionResponse
3230 element contains information which describes the success or failure of a
3231 Subscription, providing any error messages in the event of failure. It also
3232 returns the registry URN of the subscription, and the subscriber-assigned ID.

3233 *Element Content (Type):*

3234
3235 SubscriptionURN (xs:anyURI) - min. 0
3236 SubscriberAssignedID (common:IDType) - min. 0
3237 SubscriptionStatus (StatusMessageType)

3238 **NotifyRegistryEventType:** The NotifyRegistryEvent element is sent by the
3239 registry services to subscribers, to notify them of specific registration and
3240 change events. EventTime specifies the time of the triggering event.
3241 ObjectURN provides the URN of the object on which the event occurred.
3242 SubscriptionURN provides the registry/repository URN of the subscription.
3243 EventAction indicates the nature of the event - whether the action was an
3244 addition, a modification, or a deletion.

3245 *Element Content (Type):*

3246
3247 EventTime (xs:dateTime)
3248 ObjectURN (xs:anyURI)
3249 SubscriptionURN (xs:anyURI)
3250 EventAction (common:ActionType)
3251 StructuralEvent (StructuralEventType) [Choice]
3252 ProvisioningEvent (ProvisioningEventType) [Choice]
3253 RegistrationEvent (RegistrationEventType) [Choice]

3254 **SubmitRegistrationRequestType:** SubmitRegistrationRequest is sent to the
3255 registry by an agency or data/metadata provider to request registration for a
3256 data set or metadata set. The resource to be registered must be accessible to
3257 the registry services at an indicated URL, so that it can be processed by those
3258 services. This is the datasource, which may also have been specified for the
3259 data provider or provision agreement, in which case it need not appear here.
3260 Constraints describing the content and release calendar of the registered
3261 dataset (and, for metadata sets, the release calendar) may also be included.



3262 *Element Content (Type):*

3263
3264 Registration (RegistrationType) - max. unbounded

3265 **SubmitRegistrationResponseType:** This document is sent to the agency or
3266 data/metadata provider in response to a registration request. It indicates the
3267 success or failure of the registration request, and contains any error
3268 messages generated by the registration service.

3269 *Element Content (Type):*

3270
3271 RegistrationStatus (RegistrationStatusType) - max. unbounded

3272 **QueryRegistrationRequestType:** The QueryRegistrationRequest is used to
3273 query the contents of a registry for data sets and metadata sets. The
3274 QueryRegistrationRequest specifies whether the result set should include
3275 metadata sets, data sets, or both with the QueryType element. The
3276 constraints which characterize the search - including reference period, are
3277 contained in teh Constraints within the child object references.

3278 *Element Content (Type):*

3279
3280 QueryType (QueryTypeType)
3281 ProvisionAgreementRef (ProvisionAgreementRefType) [Choice]
3282 DataflowRef (DataflowRefType) [Choice]
3283 MetadataflowRef (MetadataflowRefType) [Choice]
3284 DataProviderRef (DataProviderRefType) [Choice]

3285 **QueryRegistrationResponseType:** The QueryRegistrationResponse is sent
3286 as a response document to anyone querying the contents of a registry. The
3287 results set contains a set of links to data and/or metadata. If the result set is
3288 null, or there is some other problem with the query, then appropriate error
3289 messages and statuses will be returned.

3290 *Element Content (Type):*

3291
3292 QueryResult (QueryResultType) - max. unbounded

3293 **SubmitStructureRequestType:** SubmitStructureRequest is used to submit
3294 structure definitions - key families, metadata structures - to the repository. The
3295 structure resources (key families, agencies, concepts and concept schemes,
3296 codelists, etc.) to be submitted must be available as valid SDMX-ML Structure
3297 messages external to the registry, so that they can be retrieved by the
3298 repository submission service. A SubmitStructureResponse will be sent in
3299 response, and will indicate status and contain any relevant error information.
3300 StructureLocation holds the URL of the valid Structure Message. Alternately,
3301 the Structure element can contain the structural descriptions. The

3302 SubmittedStructureType contains a reference to one of the structural
 3303 maintainable artefacts detailed in the Structure Message, which is to be
 3304 submitted to the repository. It does not need to be used when the structures
 3305 being submitted are included in the request message, or when all objects in
 3306 the referenced SDMX-ML Structure message are to be submitted.

3307 *Element Content (Type):*

3308
 3309 StructureLocation (xs:anyURI) [Choice]
 3310 Structure (StructureType) [Choice]
 3311 SubmittedStructure (SubmittedStructureType) - min. 0 - max. unbounded

3312 **SubmitStructureResponseType:** SubmitStructureResponse is returned by
 3313 the registry when a SubmitStructure is received. It indicates the status of the
 3314 submission, and carries any error messages which are generated, if relevant.
 3315 For each submitted structure, a SubmissionResult will be returned.

3316 *Element Content (Type):*

3317
 3318 SubmissionResult (SubmissionResultType) - max. unbounded

3319 **QueryStructureRequestType:** QueryStructureRequest is used to query the
 3320 registry for any maintainable object within the repository. The response is a
 3321 Structure message. In the reference elements to the queryable registry
 3322 objects, a valid registry URN or a complete set of other child elements may be
 3323 used to identify the objects desired in the result set. Any part of an element-
 3324 based (that is, non-URN) identification of an object which is not provided will
 3325 be understood as a wild-card value, referring to "all" possible values. The
 3326 resolveReferences attribute is set to true if all dependent objects should also
 3327 be returned as part of the result set. (For example, if you query for a key
 3328 family and want to also have all codelists, concepts, and agencies, returned
 3329 as well, resolveReferences should be set to true.)

3330 *Element Content (Type):*

3331 (Choice)
 3332 AgencyRef (AgencyRefType) [Choice]
 3333 DataProviderRef (DataProviderRefType) [Choice]
 3334 DataflowRef (DataflowRefType) [Choice]
 3335 MetadataflowRef (MetadataflowRefType) [Choice]
 3336 CodelistRef (CodelistRefType) [Choice]
 3337 CategorySchemeRef (CategorySchemeRefType) [Choice]
 3338 ConceptSchemeRef (ConceptSchemeRefType) [Choice]
 3339 OrganisationSchemeRef (OrganisationSchemeRefType) [Choice]
 3340 KeyFamilyRef (KeyFamilyRefType) [Choice]
 3341 MetadataStructureRef (MetadataStructureRefType) [Choice]
 3342 HierarchicalCodelistRef (HierarchicalCodelistRefType) [Choice]
 3343 StructureSetRef (StructureSetRefType) [Choice]
 3344 ProcessRef (ProcessRefType) [Choice]
 3345 ReportingTaxonomyRef (ReportingTaxonomyRefType) [Choice]



3346 *Attribute:* resolveReferences (xs:boolean) - required

3347 **QueryStructureResponseType:** QueryStructureResponse is sent in
3348 response to a QueryStructureRequest. It carries the status of the response,
3349 with any relevant error messages, and then also carries all information found
3350 in the result set.

3351 *Element Content (Type):*

3352
3353 StatusMessage (StatusMessageType)
3354 OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
3355 Dataflows (structure:DataflowsType) - min. 0
3356 Metadataflows (structure:MetadataflowsType) - min. 0
3357 CategorySchemes (structure:CategorySchemesType) - min. 0
3358 CodeLists (structure:CodeListsType) - min. 0
3359 HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
3360 Concepts (structure:ConceptsType) - min. 0
3361 MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) -
3362 min. 0
3363 KeyFamilies (structure:KeyFamiliesType) - min. 0
3364 StructureSets (structure:StructureSetsType) - min. 0
3365 ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0
3366 Processes (structure:ProcessesType) - min. 0

3367 **SubmitProvisioningRequestType:** This document is sent to the registry
3368 services to submit provisioning information. A provision agreement is typically
3369 sent, which has internal references to existing data providers and
3370 dataflows/metadataflows. These elements are also included as possible
3371 separate submissions, because it may be necessary to provide datasource
3372 and constraint information independent of the establishment of a provision
3373 agreement.

3374 *Element Content (Type):*

3375 *(Choice)*
3376 ProvisionAgreement (ProvisionAgreementType) [Choice]
3377 DataProviderRef (DataProviderRefType) [Choice]
3378 DataflowRef (DataflowRefType) [Choice]
3379 MetadataflowRef (MetadataflowRefType) [Choice]

3380 **SubmitProvisioningResponseType:** The ProvisioningResponse element is
3381 returned by the registry services in response to a provisioning request. It
3382 contains information about the status of the submitted provisioning
3383 information, and any relevant error messages in case of failure.

3384 *Element Content (Type):*

3385
3386 ProvisioningStatus (ProvisioningStatusType) - max. unbounded



3387 **QueryProvisioningRequestType:** QueryProvisioningRequest is used to
3388 query the repository for provisioning metadata. The response is a
3389 QueryProvisioningResponse document, carrying either the result set of the
3390 query or relevant error messages. Note that whatever information is presented
3391 here, regarding provision agreements, data flow, metadataflow, or data
3392 providers, is taken to be the search criteria - the query is for all provision
3393 agreements which match the supplied criteria. If any of provision agreement,
3394 metadataflow, dataflow, or data provider are omitted, the search will apply to
3395 all values for those objects in the repository.

3396 *Element Content (Type):*

3397
3398 ProvisionAgreementRef (ProvisionAgreementRefType) - min. 0
3399 DataflowRef (DataflowRefType) - min. 0
3400 MetadataflowRef (MetadataflowRefType) - min. 0
3401 DataProviderRef (DataProviderRefType) - min. 0

3402 **QueryProvisioningResponseType:** The QueryProvisioningResponse
3403 element is returned in response to queries regarding provisioning information.
3404 It carries either the provisioning information making up the result set, or
3405 relevant status messages containing errors or warnings, or both. The
3406 references to Dataflow, Metadataflow, and Data Provider are included in those
3407 cases where the result set has these objects, but not associated with any
3408 Provisioning Agreement.

3409 *Element Content (Type):*

3410
3411 ProvisionAgreement (ProvisionAgreementType) - min. 0 - max. unbounded
3412 DataflowRef (DataflowRefType) - min. 0 - max. unbounded
3413 MetadataflowRef (MetadataflowRefType) - min. 0 - max. unbounded
3414 DataProviderRef (DataProviderRefType) - min. 0 - max. unbounded
3415 StatusMessage (StatusMessageType)

3416 **SubscriptionType:** Subscriptions submit a subscription for a registry or
3417 repository object.Action indicates what action is being taken by sending the
3418 request. RegistryURN is used to identify the subscription in the case of
3419 deletion or modification. NotificationMailTo holds an e-mail address (the
3420 "mailto:" protocol); NotificationHTTP holds an http address to which
3421 notifications can be addressed as POSTs. SubscriberAssignedID allows the
3422 subscriber to specify an ID which will be returned as part of the notification for
3423 the subscribed events. Validity period sets a start and end date for the
3424 subscription, EventSelector indicates an event or events for the subscription.

3425 *Element Content (Type):*

3426
3427 Action (common:ActionType)
3428 RegistryURN (xs:anyURI) - min. 0
3429 NotificationMailTo (xs:anyURI) - min. 0



3430 NotificationHTTP (xs:anyURI) - min. 0
3431 SubscriberAssignedID (common:IDType) - min. 0
3432 ValidityPeriod (ValidityPeriodType)
3433 EventSelector (EventSelectorType)

3434 **ValidityPeriodType:** Specifies inclusive start and end-dates for the
3435 subscription period.

3436 *Element Content (Type):*

3437
3438 StartDate (xs:date)
3439 EndDate (xs:date)

3440 **EventSelectorType:** Allows subscribers to specify registry and repository
3441 events for which they wish to receive notifications.

3442 *Element Content (Type):*

3443
3444 StructuralRepositoryEvents (StructuralRepositoryEventsType) - min. 0
3445 ProvisioningRepositoryEvents (ProvisioningRepositoryEventsType) - min. 0
3446 DataRegistrationEvents (DataRegistrationEventsType) - min. 0
3447 MetadataRegistrationEvents (MetadataRegistrationEventsType) - min. 0

3448 **StructuralRepositoryEventsType:** Contains details of the subscribed
3449 structural repository events. AgencyID specifies an agency for the object or
3450 objects indicated in the other ID fields. Note that the ID fields (including
3451 AgencyID) may hold a complete ID or Repository URN, but may also insert
3452 the "%" wildcard character, which represents 0 or more characters, in the ID
3453 string. If left empty, all objects will be matched within the other constraints
3454 (agency, object type) provided.

3455 *Element Content (Type):*

3456
3457 AgencyID (common:IDType) - min. 0 - max. unbounded
3458 AllEventsID (xs:string) - min. 0 - max. unbounded
3459 KeyFamilyID (xs:string) - min. 0 - max. unbounded
3460 ConceptSchemeID (xs:string) - min. 0 - max. unbounded
3461 CodeListID (xs:string) - min. 0 - max. unbounded
3462 MetadataStructureID (xs:string) - min. 0 - max. unbounded
3463 CategorySchemeID (xs:string) - min. 0 - max. unbounded
3464 DataflowID (xs:string) - min. 0 - max. unbounded
3465 MetadataflowID (xs:string) - min. 0 - max. unbounded
3466 OrganisationSchemeID (xs:string) - min. 0 - max. unbounded
3467 HierarchicalCodelistID (xs:string) - min. 0 - max. unbounded
3468 StructureSetID (xs:string) - min. 0 - max. unbounded
3469 ReportingTaxonomyID (xs:string) - min. 0 - max. unbounded
3470 ProcessID (xs:string) - min. 0 - max. unbounded

3471 **ProvisioningRepositoryEventsType:** Contains details of the subscribed
3472 provisioning repository events. Note that the ID fields may hold a complete ID



3473 or Repository URN, but may also insert the "%" wildcard character, which
3474 represents 0 or more characters, in the ID string. If left empty, all objects will
3475 be matched within the other constraints (agency, object type) provided.

3476 *Element Content (Type):*

3477

3478 ProvisionAgreementID (common:IDType) - min. 0 - max. unbounded

3479 DataProviderID (xs:string) - min. 0 - max. unbounded

3480 DataflowID (xs:string) - min. 0 - max. unbounded

3481 MetadataflowID (xs:string) - min. 0 - max. unbounded

3482 AllEventsID (xs:string) - min. 0 - max. unbounded

3483 **DataRegistrationEventsType:** Contains details of the subscribed data
3484 registry events. Note that the ID fields may hold a complete ID or Registry
3485 URN, but may also insert the "%" wildcard character, which represents 0 or
3486 more characters, in the ID string. If left empty, all objects will be matched
3487 within the other constraints (agency, object type) provided.

3488 *Element Content (Type):*

3489

3490 AllEventsID (xs:string) - min. 0 - max. unbounded

3491 DataProviderID (xs:string) - min. 0 - max. unbounded

3492 ProvisionAgreementID (xs:string) - min. 0 - max. unbounded

3493 DataflowID (xs:string) - min. 0 - max. unbounded

3494 KeyFamilyID (xs:string) - min. 0 - max. unbounded

3495 CategoryID (xs:string) - min. 0 - max. unbounded

3496 CategorySchemeID (xs:string) - min. 0 - max. unbounded

3497 CategorySchemeAgencyID (xs:string) - min. 0 - max. unbounded

3498 **MetadataRegistrationEventsType:** Contains details of the subscribed
3499 metadadata registry events. Note that the ID fields may hold a complete ID or
3500 Registry URN, but may also insert the "%" wildcard character, which
3501 represents 0 or more characters, in the ID string. If left empty, all objects will
3502 be matched within the other constraints (agency, object type) provided.

3503 *Element Content (Type):*

3504

3505 AllEventsID (xs:string) - min. 0 - max. unbounded

3506 DataProviderID (xs:string) - min. 0 - max. unbounded

3507 ProvisionAgreementID (xs:string) - min. 0 - max. unbounded

3508 MetadataflowID (xs:string) - min. 0 - max. unbounded

3509 MetadatastructureID (xs:string) - min. 0 - max. unbounded

3510 CategoryID (xs:string) - min. 0 - max. unbounded

3511 **StructuralEventType:** This provides the details of a structural repository
3512 event for the purposes of notification.

3513 *Element Content (Type):*

3514
 3515 OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
 3516 Dataflows (structure:DataflowsType) - min. 0
 3517 Metadataflows (structure:MetadataflowsType) - min. 0
 3518 CategorySchemes (structure:CategorySchemesType) - min. 0
 3519 CodeLists (structure:CodeListsType) - min. 0
 3520 HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
 3521 Concepts (structure:ConceptsType) - min. 0
 3522 MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) -
 3523 min. 0
 3524 KeyFamilies (structure:KeyFamiliesType) - min. 0
 3525 StructureSets (structure:StructureSetsType) - min. 0
 3526 Processes (structure:ProcessesType) - min. 0
 3527 ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0

3528 **ProvisioningEventType:** This provides the details of a provisioning event for
 3529 the purposes of notification.

3530 *Element Content (Type):*

3531
 3532 DataProviderRef (DataProviderRefType) - min. 0
 3533 DataflowRef (DataflowRefType) - min. 0
 3534 MetadataflowRef (MetadataflowRefType) - min. 0
 3535 ProvisionAgreement (ProvisionAgreementType) - min. 0

3536 **RegistrationEventType:** This provides the details of a data or metadata
 3537 registration event for the purposes of notification.

3538 *Element Content (Type):*

3539
 3540 Registration (RegistrationType)

3541 **ProvisionAgreementType:** Provision agreements contain a reference to a
 3542 pre-existing data flow or metadata flow definition and a pre-existing data (or
 3543 metadata) provider in the registry. They also must have the action attribute
 3544 set, indicating whether this is an addition, a modification, or a deletion of a
 3545 provision agreement. They may also supply boolean values which describe
 3546 how the registry must behave: if indexTimeSeries is true, then the registry
 3547 must index all time series when a data set is registered against this provision
 3548 agreement; if indexDataSet is true, then the registry must index the range of
 3549 actual (present) values for each dimension of the Dataset (as indicated in the
 3550 dataset's key family); if indexReportingPeriod is true, then the registry must
 3551 index the time period ranges for which data are present in the dataset(s)
 3552 registered against the provision agreement. Note that the values for these
 3553 attributes are not needed when a Delete action is indicated. As for all
 3554 identifiable objects, provision agreements have Name and Description
 3555 elements, which are repeatable to provide language-specific forms. These
 3556 may be omitted if the provision agreement already exists (as is the case for
 3557 modification and deletion); descriptions are always optional. The id attribute
 3558 holds the unique id of the provision agreement as derived (according to teh

3559 logical registry specification.) If specified, the uri attribute points to a location
 3560 (typically a URL) where the provision agreement is described in a valid
 3561 QueryProvisioningResponse message. The urn attribute holds the reserved
 3562 registry URN assigned to the provision agreement - this must be included
 3563 when the ProvisionAgreement structure is used in a response document of
 3564 any type. The action attribute must be specified when the provisionAgreement
 3565 is used in a request document of any type.. Datasource is used to describe
 3566 the data source associated with the provider agreement. Annotations may be
 3567 provided in the Annotations element. The validFrom and validTo attributes
 3568 provide inclusive dates for providing supplemental validity information about
 3569 the version.

3570 *Element Content (Type):*

3571
 3572 Name (common:TextType) - min. 0 - max. unbounded
 3573 Description (common:TextType) - min. 0 - max. unbounded
 3574 DataflowRef (DataflowRefType) [Choice]
 3575 MetadataflowRef (MetadataflowRefType) [Choice]
 3576 DataProviderRef (DataProviderRefType)
 3577 Datasource (QueryableDatasourceType) - min. 0
 3578 Constraint (common:ConstraintType) - min. 0
 3579 Annotations (common:AnnotationsType) - min. 0

3580 *Attribute:* id (common:IDType) - optional

3581 *Attribute:* uri (xs:anyURI) - optional

3582 *Attribute:* urn (xs:anyURI) - optional

3583 *Attribute:* action (common:ActionType) - optional

3584 *Attribute:* indexTimeSeries (xs:boolean) - optional

3585 *Attribute:* indexDataSet (xs:boolean) - optional

3586 *Attribute:* indexReportingPeriod (xs:boolean) - optional

3587 *Attribute:* validFrom (common:TimePeriodType) - optional

3588 *Attribute:* validTo (common:TimePeriodType) - optional

3589 **DatasourceType:** Datasource specifies the properties of a data or metadata
 3590 source. A SimpleDatasource requires only the URL of the data. A
 3591 QueryableDatasource must be able to accept an SDMX-ML Query Message,
 3592 and respond appropriately. Either or both may be specified.

3593 *Element Content (Type):*



3594
3595
3596

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

3597 **QueryableDatasourceType:** If the data provider uses a single, queryable
3598 data source for all provision agreements contained in the ProvisionRequest
3599 element, then this element should be used to describe the queryable
3600 datasource. DataUrl contains the URL of the datasource, with WSDLUrl
3601 optionally providing the location of a WSDL instance on the internet which
3602 describes the queryable datasource. The attributes isRESTDatasource and
3603 isWebServiceDatasource indicate, if true, that the queryable datasource is
3604 accessible via the REST protocol and/or Web Services protocols,
3605 respectively.

3606 *Element Content (Type):*

3607
3608
3609

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

3610 *Attribute:* isRESTDatasource (xs:boolean) - required

3611 *Attribute:* isWebServiceDatasource (xs:boolean) - required

3612 **ProvisioningStatusType:** For each provision agreement, dataflow reference,
3613 metadataflow reference, or data provider reference submitted in a provisioning
3614 request, a provisioning status will be returned, providing a status and any
3615 warnings or errors.

3616 *Element Content (Type):*

3617
3618
3619
3620
3621
3622

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

3623 **RegistrationType:** Registration provides the information needed for data and
3624 reference metadata set registration. LastUpdated can provide a time stamp;
3625 ValidFrom and ValidTo allow for effectivity, so that data visibility from the
3626 registry can be controlled by the registrant. A Datasource must be supplied
3627 here if not already provided in the provision agreement. The data set or
3628 metadata set must be associated with a provision agreement, a metadataflow,
3629 or a dataflow definition. If possible, the provision agreement should be
3630 specified. Only in cases where this is not possible should the dataflow or
3631 metadataflow be used.

3632 *Element Content (Type):*



3633
3634 LastUpdated (xs:dateTime) - min. 0
3635 ValidFrom (xs:dateTime) - min. 0
3636 ValidTo (xs:dateTime) - min. 0
3637 Action (common:ActionType)
3638 Datasource (DatasourceType) - min. 0
3639 DataflowRef (DataflowRefType) [Choice]
3640 MetadataflowRef (MetadataflowRefType) [Choice]
3641 ProvisionAgreementRef (ProvisionAgreementRefType) [Choice]

3642 **RegistrationStatusType:** Each RegistrationStatus reports the status of a
3643 submitted data set or metadata set registration, and carries any error
3644 messages. If successful, the Datasource which has been registered is
3645 returned, and a reference to the provision agreement, dataflow, metadataflow,
3646 or data provider is returned.

3647 *Element Content (Type):*

3648
3649 StatusMessage (StatusMessageType)
3650 Datasource (DatasourceType) - min. 0
3651 DataProviderRef (DataProviderRefType) - min. 0
3652 DataflowRef (DataflowRefType) - min. 0
3653 MetadataflowRef (MetadataflowRefType) - min. 0
3654 ProvisionAgreementRef (ProvisionAgreementRefType) - min. 0

3655 **QueryResultType:** QueryResult contains the results of a specific registry
3656 query for a single datasource. If a successful result is a registered dataset, the
3657 DataResult element is used. If a successful result is a registered metadataset,
3658 the MetadataResult is used. If the query failed, then StatusMessage is
3659 included. The timeSeriesMatch attribute is true when the result is an exact
3660 match with the key found in the registry - that is, when the registered
3661 datasource provides a matching key. It is set to false when the datasource is
3662 registered with cube-region constraints, or in any other circumstance when it
3663 cannot be established that the sought-for keys have been exactly matched.
3664 This is always true when the resulting datasource is the source of a metadata
3665 set.

3666 *Element Content (Type):*

3667 (Choice)
3668 DataResult (ResultType) [Choice]
3669 MetadataResult (ResultType) [Choice]
3670 StatusMessage (StatusMessageType) [Choice]

3671 *Attribute:* timeSeriesMatch (xs:boolean) - required

3672 **ResultType:** Result contains the information about either a data or metadata
3673 source, being returned as part of a QueryResult element. If the

3674 *Element Content (Type):*



3675
3676 Datasource (DatasourceType) - min. 0
3677 ProvisionAgreementRef (ProvisionAgreementRefType) [Choice]
3678 DataflowRef (DataflowRefType) [Choice]
3679 MetadataflowRef (DataflowRefType) [Choice]
3680 DataProviderRef (DataProviderRefType) [Choice]

3681 **StructureType:** Holds the structure information for submission to the
3682 structural repository.

3683 *Element Content (Type):*

3684
3685 OrganisationSchemes (structure:OrganisationSchemesType) - min. 0
3686 Dataflows (structure:DataflowsType) - min. 0
3687 Metadataflows (structure:MetadataflowsType) - min. 0
3688 CategorySchemes (structure:CategorySchemesType) - min. 0
3689 CodeLists (structure:CodeListsType) - min. 0
3690 HierarchicalCodelists (structure:HierarchicalCodelistsType) - min. 0
3691 Concepts (structure:ConceptsType) - min. 0
3692 MetadataStructureDefinitions (structure:MetadataStructureDefinitionsType) -
3693 min. 0
3694 KeyFamilies (structure:KeyFamiliesType) - min. 0
3695 StructureSets (structure:StructureSetsType) - min. 0
3696 Processes (structure:ProcessesType) - min. 0
3697 ReportingTaxonomies (structure:ReportingTaxonomiesType) - min. 0

3698 **SubmittedStructureType:** SubmittedStructure holds a reference to a
3699 structural object to be stored in the repository. The externalDependencies
3700 attribute should be set to true if the repository is expected to use URLs in the
3701 structure Message to retrieve objects on which the stored object has
3702 dependencies. (Thus, if a key family is being submitted to the repository, and
3703 the structure message has URLKs which point to the locations of the codelists
3704 it uses, then the externalDependencies attribute should be set to true.) the
3705 action attribute specifies whether the Structure being submitted is intended to
3706 be added or deleted from the repository. The "modify" action is not applicable
3707 to final structures in the repository, and will produce an error condition, as
3708 these can be versioned but not modified. To submit a later version of a
3709 structure, the structure message should include the incremented version
3710 number. The externalDependencies and action attributes need not be
3711 specified in a response document. The isFinal attribute indicates whether the
3712 structure being submitted to the repository is final or not - this can also be
3713 specified on the structures themselves.

3714 *Element Content (Type):*

3715 (Choice)
3716 DataflowRef (DataflowRefType) [Choice]
3717 MetadataflowRef (MetadataflowRefType) [Choice]
3718 CodelistRef (CodelistRefType) [Choice]
3719 HierarchicalCodelistRef (HierarchicalCodelistRefType) [Choice]
3720 CategorySchemeRef (CategorySchemeRefType) [Choice]
3721 ConceptSchemeRef (ConceptSchemeRefType) [Choice]



- 3722 OrganisationSchemeRef (OrganisationSchemeRefType) [Choice]
3723 KeyFamilyRef (KeyFamilyRefType) [Choice]
3724 MetadataStructureRef (MetadataStructureRefType) [Choice]
3725 ProcessRef (ProcessRefType) [Choice]
3726 StructureSetRef (StructureSetRefType) [Choice]
3727 ReportingTaxonomyRef (ReportingTaxonomyRefType) [Choice]
- 3728 *Attribute:* externalDependencies (xs:boolean) - optional
- 3729 *Attribute:* action (common:ActionType) - optional
- 3730 *Attribute:* isFinal (xs:boolean) - optional
- 3731 **SubmissionResultType:** For each Structure object submitted to the
3732 repository in a SubmitStructure, a SubmissionResult will be returned. It will
3733 identify the object submitted, report back the action requested, and convey the
3734 status and any error messages which are relevant to the submission.
- 3735 *Element Content (Type):*
- 3736 SubmittedStructure (SubmittedStructureType)
3737 StatusMessage (StatusMessageType)
3738
- 3739 **ProvisionAgreementRefType:** ProvisionAgreementRef allows for the
3740 identification of a provision agreement. At a minimum, either the URN element
3741 - holding a valid registry URN - or the set of OrganisationSchemeAgencyID,
3742 OrganisationSchemeID, DataProviderID, DataflowAgencyID, and DataflowID
3743 must be specified. When used in a response document of any type, the URN
3744 must always be provided. Datasource can be used to specify a datasource for
3745 the provision agreement. Constraint can be used to express constraints
3746 associated with the provision agreement.
- 3747 *Element Content (Type):*
- 3748 URN (xs:anyURI) - min. 0
3749 OrganisationSchemeAgencyID (common:IDType) - min. 0
3750 OrganisationSchemeID (common:IDType) - min. 0
3751 DataProviderID (common:IDType) - min. 0
3752 DataProviderVersion (xs:string) - min. 0
3753 DataflowAgencyID (common:IDType) - min. 0
3754 DataflowID (common:IDType) - min. 0
3755 DataflowVersion (xs:string) - min. 0
3756 Datasource (DatasourceType) - min. 0
3757 Constraint (common:ConstraintType) - min. 0
3758
- 3759 **MetadataflowRefType:** The MetadataflowRef type structures a reference to a
3760 metadataflow definition. This requires that ID are provided for a pre-existing
3761 Agency and Metadataflow Definition in the registry. The Version element may
3762 be used to specify the version of the indicated dataflow. If absent, the most



3763 recent version is assumed. The URN element is used to provide the registry-
3764 specific URN as an alternate means of identification. When used in a
3765 response document of any type, the URN must always be provided. At a
3766 minimum, either the URN element or AgencyID, MetadataflowID, and
3767 (optionally) version must be supplied. Datasource may be used to specify a
3768 datasource. Constraint can be used to provide constraints associated with the
3769 metadataflow.

3770 *Element Content (Type):*

3771
3772 URN (xs:anyURI) - min. 0
3773 AgencyID (common:IDType) - min. 0
3774 MetadataflowID (common:IDType) - min. 0
3775 Version (xs:string) - min. 0
3776 Datasource (DatasourceType) - min. 0
3777 Constraint (common:ConstraintType) - min. 0

3778 **DataflowRefType:** The DataflowRef type structures a reference to a dataflow
3779 definition. This requires that ID are provided for a pre-existing Agency and
3780 Dataflow Definition in the registry. The Version element may be used to
3781 specify the version of the indicated dataflow. If absent, the most recent
3782 version is assumed. The URN element is used to provide the registry-specific
3783 URN as an alternate means of identification. At a minimum, either the URN
3784 element or AgencyID, DataflowID, and (optionally) version must be supplied.
3785 When used in a response document of any type, the URN must always be
3786 provided. Datasource may be used to specify a datasource. Constraints can
3787 be used to specify constraints associated with the dataflow.

3788 *Element Content (Type):*

3789
3790 URN (xs:anyURI) - min. 0
3791 AgencyID (common:IDType) - min. 0
3792 DataflowID (common:IDType) - min. 0
3793 Version (xs:string) - min. 0
3794 Datasource (DatasourceType) - min. 0
3795 Constraint (common:ConstraintType) - min. 0

3796 **DataProviderRefType:** The DataProviderRef type structures a reference to a
3797 data provider. This requires that IDs be provided for an organisation scheme,
3798 its maintenance agency, and the data provider as identified in the referenced
3799 organisation scheme. The Version element may be used to specify the
3800 version of the indicated data provider. If absent, the most recent version is
3801 assumed. The URN element is used to provide the registry-specific urn as an
3802 alternate means of identification. At a minimum, either the URN element or
3803 OrganisationSchemeID, OrganisationSchemeAgencyID, DataProviderID,
3804 and (optionally) Version must be supplied. When used in a response
3805 document of any type, the URN must always be provided. Datasource may be
3806 used to specify a datasource. Constraints can be used to specify constraints
3807 associated with the data provider.



3808 *Element Content (Type):*

3809
3810 URN (xs:anyURI) - min. 0
3811 OrganisationSchemeAgencyID (common:IDType)
3812 OrganisationSchemeID (common:IDType)
3813 DataProviderID (common:IDType)
3814 Version (xs:string) - min. 0
3815 Datasource (DatasourceType) - min. 0
3816 Constraint (common:ConstraintType) - min. 0

3817 **AgencyRefType:** The AgencyRef type structures a reference to an Agency.
3818 This requires that IDs be provided for an organisation scheme, its
3819 maintenance agency, and the agency as identified in the referenced
3820 organisation scheme. The Version element may be used to specify the
3821 version of the indicated agency. If absent, the most recent version is
3822 assumed. The URN element is used to provide the registry-specific urn as an
3823 alternate means of identification. At a minimum, either the URN element or
3824 OrganisationSchemeID, OrganisationSchemeAgencyID, AgencyID, and
3825 (optionally) Version must be supplied. When used in a response document of
3826 any type, the URN must always be provided.

3827 *Element Content (Type):*

3828
3829 URN (xs:anyURI) - min. 0
3830 OrganisationSchemeAgencyID (common:IDType)
3831 OrganisationSchemeID (common:IDType)
3832 AgencyID (common:IDType)
3833 Version (xs:string) - min. 0

3834 **CodelistRefType:** KeyFamilyRef allows for references to specific codelists. At
3835 a minimum, either the URN - which contains a valid Registry/Repository URN
3836 - or the rest of the child elements must be supplied. When used in a response
3837 document of any type, the URN must always be provided.

3838 *Element Content (Type):*

3839
3840 URN (xs:anyURI) - min. 0
3841 AgencyID (common:IDType) - min. 0
3842 CodelistID (common:IDType) - min. 0
3843 Version (xs:string) - min. 0

3844 **CategorySchemeRefType:** CategorySchemeRef allows for references to
3845 specific category schemes. At a minimum, either the URN - which contains a
3846 valid Registry/Repository URN - or the rest of the child elements must be
3847 supplied. When used in a response document of any type, the URN must
3848 always be provided.

3849 *Element Content (Type):*



3850
3851
3852
3853
3854

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

3855 **ConceptSchemeRefType:** ConceptSchemeRef allows for references to
3856 specific concept schemes. At a minimum, either the URN - which contains a
3857 valid Registry/Repository URN - or the rest of the child elements must be
3858 supplied. When used in a response document of any type, the URN must
3859 always be provided.

3860 *Element Content (Type):*

3861
3862
3863
3864
3865

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

3866 **OrganisationSchemeRefType:** OrganisationSchemeRef allows for
3867 references to specific organisation schemes. At a minimum, either the URN -
3868 which contains a valid Registry/Repository URN - or the rest of the child
3869 elements must be supplied. When used in a response document of any type,
3870 the URN must always be provided.

3871 *Element Content (Type):*

3872
3873
3874
3875
3876

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

3877 **KeyFamilyRefType:** KeyFamilyRef allows for references to specific key
3878 families (data structure definitions). At a minimum, either the URN - which
3879 contains a valid Registry/Repository URN - or the rest of the child elements
3880 must be supplied. When used in a response document of any type, the URN
3881 must always be provided.

3882 *Element Content (Type):*

3883
3884
3885
3886
3887

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

3888 **MetadataStructureRefType:** MetadataStructureRef allows for references to
3889 specific metadata structure definitions. At a minimum, either the URN - which
3890 contains a valid Registry/Repository URN - or the rest of the child elements



3891 must be supplied. When used in a response document of any type, the URN
3892 must always be provided.

3893 *Element Content (Type):*

3894
3895 URN (xs:anyURI) - min. 0
3896 AgencyID (common:IDType) - min. 0
3897 MetadataStructureID (common:IDType) - min. 0
3898 Version (xs:string) - min. 0

3899 **HierarchicalCodelistRefType:** HierarchicalCodelistRef allows for references
3900 to specific hierarchical codelists. At a minimum, either the URN - which
3901 contains a valid Registry/Repository URN - or the rest of the child elements
3902 must be supplied. When used in a response document of any type, the URN
3903 must always be provided.

3904 *Element Content (Type):*

3905
3906 URN (xs:anyURI) - min. 0
3907 AgencyID (common:IDType) - min. 0
3908 HierarchicalCodelistID (common:IDType) - min. 0
3909 Version (xs:string) - min. 0

3910 **StructureSetRefType:** StructureSetRef allows for references to specific
3911 structure sets. At a minimum, either the URN - which contains a valid
3912 Registry/Repository URN - or the rest of the child elements must be supplied.
3913 When used in a response document of any type, the URN must always be
3914 provided.

3915 *Element Content (Type):*

3916
3917 URN (xs:anyURI) - min. 0
3918 AgencyID (common:IDType) - min. 0
3919 StructureSetID (common:IDType) - min. 0
3920 Version (xs:string) - min. 0

3921 **ProcessRefType:** ProcessRef allows for references to specific processes. At
3922 a minimum, either the URN - which contains a valid Registry/Repository URN
3923 - or the rest of the child elements must be supplied. When used in a response
3924 document of any type, the URN must always be provided.

3925 *Element Content (Type):*

3926
3927 URN (xs:anyURI) - min. 0
3928 AgencyID (common:IDType) - min. 0
3929 ProcessID (common:IDType) - min. 0
3930 Version (xs:string) - min. 0

3931 **ReportingTaxonomyRefType:** ReportingTaxonomyRef allows for references
3932 to specific reporting taxonomies. At a minimum, either the URN - which
3933 contains a valid Registry/Repository URN - or the rest of the child elements
3934 must be supplied. When used in a response document of any type, the URN
3935 must always be provided.

3936 *Element Content (Type):*

3937
3938 URN (xs:anyURI) - min. 0
3939 AgencyID (common:IDType) - min. 0
3940 ReportingTaxonomyID (common:IDType) - min. 0
3941 Version (xs:string) - min. 0

3942 **StatusMessageType:** This carries the text of error messages and/or
3943 warnings in response to queries or requests. The Status attribute carries the
3944 status of the query or request.

3945 *Element Content (Type):*

3946
3947 MessageText (common:TextType) - min. 0 - max. unbounded

3948 *Attribute:* status (StatusType) - required

3949

3950 **5.7.2 Simple Types**

3951 **ConstraintTypeType:** ConstraintType provides an enumeration of values of
3952 the types of constraints.

3953 *Restricts* xs:NMTOKEN

3954 Code: Content - Content constraint.

3955 Code: Attachment - Attachment constraint.

3956 **StatusType:** StatusType provides an enumeration of values that detail the
3957 status of queries or requests.

3958 *Restricts* xs:NMTOKEN

3959 Code: Success - Query or request successful.

3960 Code: Warning - Query or request successful, but with warnings.

3961 Code: Failure - Query or request not successful.

3962 **QueryTypeType:** QueryType provides an enumeration of values which
3963 specify the objects in the result-set for a registry query.

3964 *Restricts xs:NMTOKEN*

3965 Code: DataSets - Only references data sets should be returned.

3966 Code: MetadataSets - Only references to metadata sets should be returned.

3967 Code: AllSets - References to both data sets and metadata sets should be returned.

3968

3969 **5.8 Data Formatting and Character Encoding**

3970 In all SDMX-ML documents – whether key-family-specific or not - the character
3971 encoding must be UTF-8. To simplify the exchange of statistical data and metadata
3972 globally, restrictions also apply to the expression of numeric formats: the decimal
3973 separator is always a period (“.”). There is no character used to separate thousands
3974 in data.

3975 **5.9 Missing Observation Values**

3976 In some of the SDMX-ML documents, an Observation is required (as in the Utility
3977 format) or it is desirable to indicate that a numerical value does not exist. While this
3978 information may be captured in an Observation-level attribute such as
3979 OBS_STATUS, with a code indicating that the value for the observation is missing,
3980 there is also a way to reliably indicate this state in the data itself. For this purpose,
3981 missing observation values – when included in an SDMX-ML data file – should be
3982 indicated using “NaN”. In XML, this indicates “not a number”, but is still valid in
3983 numeric fields. This avoids having to use a number (such as “-9999999” or “0”), along
3984 with a status code of “missing” (or similar construct) to indicate missing numeric
3985 values.
3986

3987 **6 KEY-FAMILY- AND METADATA-STRUCTURE-** 3988 **DEFINITION-SPECIFIC SCHEMAS: CORE** 3989 **STRUCTURES & STANDARD MAPPINGS**

3990 Some schemas are specific to key families and metadata structure definitions, and
3991 therefore there is no single schema for all users. In these cases, standard mappings
3992 are provided so that even though one schema cannot be published, the schemas can
3993 be predicted from an examination of SDMXStructure messages that describe the key
3994 families/metadata structure definitions on which they are based. Automatic creation
3995 of these structure-specific schemas according to these mappings is a natural
3996 consequence of this correspondence, and free tools to enable this creation of
3997 structure-specific schemas is envisioned.
3998

3999 It is important to note that all key-family- and metadata-structure-definition-specific
4000 schemas are based on a core of identical constructs, allowing the smallest possible
4001 number of tags to differ from schema to schema. This section first documents these
4002 “core” structures, each in their own SDMX-maintained namespace module, and then
4003 discusses the mappings from a key family to the key-family-specific schema, and
4004 from a metadata-structure-definition to a metadata-structure-definition-specific
4005 schema.
4006



4007 These schemas are all as similar as possible. They vary according to where in the
4008 common structure key values and attributes may be specified. A less obvious
4009 difference is seen in the Utility and Metadata Report schemas, which are designed to
4010 carry as much structural metadata as possible in order to allow “typical” XML tools
4011 (such as schema-guided editors and parsers) to benefit from the availability of this
4012 data - such tools are generally incapable of consulting the key family or metadata
4013 structure definition for structural metadata.

4014

4015 Note that for all key-family-specific and metadata-structure-definition-specific
4016 schemas, the namespaces must be constructed following the rules for registry URN
4017 identifiers, as described in section 5.2 of the SDMX Registry Interfaces specification,
4018 with the addition of a single field at the end of the URN:

4019

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

4020

4021

4022

4023

4024

4025

6.1 Compact Data Message Core Structure

4026

4027

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

4028

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

4029

4030

4031

6.1.1 Global Elements

4032

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

4033

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

4034

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

4035

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

4036

4037

6.1.2 Complex Types

4038

DataSetType: DataSetType acts as a structural base, which is extended
4039 through the addition of attributes to reflect the particular needs of a specific
4040 key family using the xs:extends element. Attributes are provided for describing
4041 the contents of a data or metadata set, which are particularly important for
4042 interactions with the SDMX Registry: datasetID,
4043 dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID,
4044 and dataflowID all take the IDs specified by the attribute names. The action
4045 attribute indicates whether the file is appending, replacing, or deleting.

4046 Attributes reportingBeginDate, reportingEndDate, validFromDate, and
4047 validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year,
4048 and publicationPeriod specifies the period of publication of the data in terms of
4049 whatever provisioning agreements might be in force (ie, "Q1 2005" if that is
4050 the time of publication for a data set published on a quarterly basis).

4051 *Attribute:* keyFamilyURI (xs:anyURI) - optional

4052 *Attribute:* datasetID (common:IDType) - optional

4053 *Attribute:* dataProviderSchemeAgencyId (common:IDType) -
4054 optional

4055 *Attribute:* dataProviderSchemeId (common:IDType) - optional

4056 *Attribute:* dataProviderID (common:IDType) - optional

4057 *Attribute:* dataflowAgencyID (common:IDType) - optional

4058 *Attribute:* dataflowID (common:IDType) - optional

4059 *Attribute:* action (common:ActionType) - optional

4060 *Attribute:* reportingBeginDate (common:TimePeriodType) -
4061 optional

4062 *Attribute:* reportingEndDate (common:TimePeriodType) -
4063 optional

4064 *Attribute:* validFromDate (common:TimePeriodType) - optional

4065 *Attribute:* validToDate (common:TimePeriodType) - optional

4066 *Attribute:* publicationYear (xs:gYear) - optional

4067 *Attribute:* publicationPeriod (common:TimePeriodType) -
4068 optional

4069 **GroupType:** GroupType acts as a structural base, which is extended through
4070 the addition of attributes to reflect the particular needs of a specific key family
4071 using the xs:extends element.

4072 **SeriesType:** SeriesType acts as a structural base, which is extended through
4073 the addition of attributes to reflect the particular needs of a specific key family
4074 using the xs:extends element.

4075 **ObsType:** ObsType acts as a structural base, which is extended through the
4076 addition of attributes to reflect the particular needs of a specific key family
4077 using the xs:extends element.

4078

4079

4080 **6.2 Utility Data Message Core Structure**

4081

4082 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/utility**

4083 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
4084 (SDMXCommon.xsd)

4085

4086 **6.2.1 Global Elements**

4087 **DataSet(DataSetType):** DataSet exists to act as the head of a substitution
4088 group to which key-family-specific attributes and elements are bound.

4089 **Group(GroupType):** Group exists to act as the head of a substitution group
4090 to which key-family-specific attributes and elements are bound.

4091 **Series(SeriesType):** Series exists to act as the head of a substitution group
4092 to which key-family-specific attributes and elements are bound.

4093 **Key(KeyType):** Key is an element which serves as the head of a substitution
4094 group containing the key-family-specific key values.

4095 **Obs(ObsType):** Obs exists to act as the head of a substitution group to which
4096 key-family-specific attributes and elements are bound.

4097

4098 **6.2.2 Complex Types**

4099 **DataSetType:** DataSetType acts as a structural base, which is extended
4100 through the addition of attributes and elements to reflect the particular needs
4101 of a specific key family using the xs:extends element. Attributes are provided
4102 for describing the contents of a data or metadata set, which are particularly
4103 important for interactions with the SDMX Registry: datasetID,
4104 dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID,
4105 and dataflowID all take the IDs specified by the attribute names. The action
4106 attribute indicates whether the file is appending, replacing, or deleting.
4107 Attributes reportingBeginDate, reportingEndDate, validFromDate, and
4108 validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year,
4109 and publicationPeriod specifies the period of publication of the data in terms of



- 4110 whatever provisioning agreements might be in force (ie, "Q1 2005" if that is
4111 the time of publication for a data set published on a quarterly basis).
- 4112 *Attribute:* keyFamilyURI (xs:anyURI) - optional
- 4113 *Attribute:* datasetID (common:IDType) - optional
- 4114 *Attribute:* dataProviderSchemeAgencyId (common:IDType) -
4115 optional
- 4116 *Attribute:* dataProviderSchemeld (common:IDType) - optional
- 4117 *Attribute:* dataProviderID (common:IDType) - optional
- 4118 *Attribute:* dataflowAgencyID (common:IDType) - optional
- 4119 *Attribute:* dataflowID (common:IDType) - optional
- 4120 *Attribute:* action (common:ActionType) - optional
- 4121 *Attribute:* reportingBeginDate (common:TimePeriodType) -
4122 optional
- 4123 *Attribute:* reportingEndDate (common:TimePeriodType) -
4124 optional
- 4125 *Attribute:* validFromDate (common:TimePeriodType) - optional
- 4126 *Attribute:* validToDate (common:TimePeriodType) - optional
- 4127 *Attribute:* publicationYear (xs:gYear) - optional
- 4128 *Attribute:* publicationPeriod (common:TimePeriodType) -
4129 optional
- 4130 **GroupType:** GroupType acts as a structural base, which is renamed and
4131 extended through the addition of attributes to reflect the particular needs of a
4132 specific key family using the xs:extends element.
- 4133 **SeriesType:** SeriesType acts as a structural base, which is extended through
4134 the addition of attributes to reflect the particular needs of a specific key family
4135 using the xs:extends element.
- 4136 **KeyType:** KeyType describes the abstract type which defines the Key
4137 element.

4138 **ObsType:** ObsType acts as a structural base, which is extended through the
4139 addition of attributes to reflect the particular needs of a specific key family
4140 using the xs:extends element.

4141

4142

4143 **6.3 Cross-Sectional Data Message Core Structure**

4144

4145 **http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross**

4146 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
4147 (SDMXCommon.xsd)

4148

4149 **6.3.1 Global Elements**

4150 **DataSet(DataSetType):** DataSet contains the data set.

4151 **Group(GroupType):** Group contains the group.

4152 **Section(SectionType):** Section contains the data section.

4153 **Obs(ObsType):** Obs contains the observation, with one or more measures.

4154

4155 **6.3.2 Complex Types**

4156 **DataSetType:** DataSetType acts as a structural base, which is extended
4157 through the addition of attributes to reflect the particular needs of a specific
4158 key family using the xs:extends element. Attributes are provided for describing
4159 the contents of a data or metadata set, which are particularly important for
4160 interactions with the SDMX Registry: datasetID,
4161 dataProviderSchemeAgencyID, dataProviderSchemeID, dataflowAgencyID,
4162 and dataflowID all take the IDs specified by the attribute names. The action
4163 attribute indicates whether the file is appending, replacing, or deleting.
4164 Attributes reportingBeginDate, reportingEndDate, validFromDate, and
4165 validToDate are inclusive. publicationYear holds the ISO 8601 four-digit year,
4166 and publicationPeriod specifies the period of publication of the data in terms of
4167 whatever provisioning agreements might be in force (ie, "Q1 2005" if that is
4168 the time of publication for a data set published on a quarterly basis).

4169 *Attribute:* keyFamilyURI (xs:anyURI) - optional

4170 *Attribute:* datasetID (common:IDType) - optional

- 4171 *Attribute:* dataProviderSchemeAgencyId (common:IDType) -
4172 optional
- 4173 *Attribute:* dataProviderSchemeld (common:IDType) - optional
- 4174 *Attribute:* dataProviderID (common:IDType) - optional
- 4175 *Attribute:* dataflowAgencyID (common:IDType) - optional
- 4176 *Attribute:* dataflowID (common:IDType) - optional
- 4177 *Attribute:* action (common:ActionType) - optional
- 4178 *Attribute:* reportingBeginDate (common:TimePeriodType) -
4179 optional
- 4180 *Attribute:* reportingEndDate (common:TimePeriodType) -
4181 optional
- 4182 *Attribute:* validFromDate (common:TimePeriodType) - optional
- 4183 *Attribute:* validToDate (common:TimePeriodType) - optional
- 4184 *Attribute:* publicationYear (xs:gYear) - optional
- 4185 *Attribute:* publicationPeriod (common:TimePeriodType) -
4186 optional
- 4187 **GroupType:** GroupType acts as a structural base, which is extended through
4188 the addition of attributes to reflect the particular needs of a specific key family
4189 using the xs:extends element.
- 4190 **SectionType:** SectionType acts as a structural base, which is extended
4191 through the addition of attributes to reflect the particular needs of a specific
4192 key family using the xs:extends element.
- 4193 **ObsType:** ObsType acts as a structural base, which is extended through the
4194 addition of attributes to reflect the particular needs of a specific key family
4195 using the xs:extends element. It is capable of expressing the value and
4196 attributes of any single available cross-sectional measure (when extended).
-
- 4197
- 4198
- 4199 **6.4 Metadata Report Core Structure**
- 4200
- 4201 http://www.SDMX.org/resources/SDMXML/schemas/v2_0/metadatareport



4202 *Imports:* http://www.SDMX.org/resources/SDMXML/schemas/v2_0/common
4203 (SDMXCommon.xsd)

4204

4205 **6.4.1 Global Elements**

4206 **MetadataSet(MetadataSetType):** The MetadataSet element contains
4207 metadata-structure-specific report(s)described in a single metadata structure
4208 definition. Attributes are provided for describing the contents of a data or
4209 metadata set, which are particularly important for interactions with the SDMX
4210 Registry: datasetID, dataProviderSchemeAgencyID, dataProviderSchemeID,
4211 dataflowAgencyID, and dataflowID all take the IDs specified by the attribute
4212 names. The action attribute indicates whether the file is appending, replacing,
4213 or deleting. Attributes reportingBeginDate, reportingEndDate, validFromDate,
4214 and validToDate are inclusive. publicationYear holds the ISO 8601 four-digit
4215 year, and publicationPeriod specifies the period of publication of the data in
4216 terms of whatever provisioning agreements might be in force (ie, "Q1 2005" if
4217 that is the time of publication for a data set published on a quarterly basis).

4218

4219 **6.4.2 Complex Types**

4220 **MetadataSetType:** MetadataReportType defines the structure of a metadata
4221 structure definition-specific Metadata Report. This consists of a
4222 MetadataStructureRef which holds the ID fo the metadata structure, and
4223 MetadataStructureAgencyRef, which holds the ID of the maintraenance
4224 agency of the metadata structure, and Version, which provides the version
4225 number of the referenced metadata structure definition. If not provided,
4226 version is assumed to be "1.0". This type is designed to be extended to hold
4227 the metadata-structure-sepcific fields needed to validate a specific metadata
4228 report.

4229 *Element Content (Type):*

4230
4231 MetadataStructureRef (common:IDType)
4232 MetadataStructureAgencyRef (common:IDType)
4233 Version (xs:string) - min. 0

4234 *Attribute:* MetadataStructureURI (xs:anyURI) - optional

4235 *Attribute:* datasetID (common:IDType) - optional

4236 *Attribute:* dataProviderSchemeAgencyId (common:IDType) -
4237 optional

4238 *Attribute:* dataProviderSchemeld (common:IDType) - optional

- 4239 *Attribute:* dataProviderID (common:IDType) - optional
- 4240 *Attribute:* dataflowAgencyID (common:IDType) - optional
- 4241 *Attribute:* dataflowID (common:IDType) - optional
- 4242 *Attribute:* action (common:ActionType) - optional
- 4243 *Attribute:* reportingBeginDate (common:TimePeriodType) -
4244 optional
- 4245 *Attribute:* reportingEndDate (common:TimePeriodType) -
4246 optional
- 4247 *Attribute:* validFromDate (common:TimePeriodType) - optional
- 4248 *Attribute:* validToDate (common:TimePeriodType) - optional
- 4249 *Attribute:* publicationYear (xs:gYear) - optional
- 4250 *Attribute:* publicationPeriod (common:TimePeriodType) -
4251 optional

4252

4253

4254 **6.5 Mappings to Key-Family-Specific Data Schemas**

4255 **6.5.1 General Rules:**

4256

4257 For all key-family-specific schemas (Compact, Utility, and Cross-Sectional) SDMX
4258 provides a namespace to be used as the extension base for key-family-specific
4259 schemas of that type. The key-family-specific schema will be created in its own target
4260 name space, owned and maintained by the creating agency. It will use the
4261 targetNamespace attribute of the schema element to identify the namespace which
4262 contains the key-family-specific schema. The namespace module provided by SDMX
4263 for that class of key-family-specific schema will be incorporated using the import
4264 element in the key-family-specific schema. The SDMX Common namespace module
4265 must also be imported into the schema. Other xml:namespace attributes may be
4266 added to the schema element as needed.

4267

4268 The elementFormDefault attribute on the schema element will be given a value of
4269 "qualified", and the attributeFormDefault attribute on the schema element will be
4270 given a value of "unqualified".

4271

4272 All additions to the SDMX module will be made using the extends element from W3C
4273 XML Schema. The term "levels of structure," when referring to the imported SDMX
4274 modules, include the following:

4275

- 4276 • DataSet level
- 4277 • Group level
- 4278 • Series level
- 4279 • Observation level

4280

4281 These levels normally refer to the element provided by the SDMX module to which
 4282 attributes and elements may be assigned. In some cases, specific named constructs
 4283 in the key family will become members of a set of elements corresponding to one of
 4284 the levels named above.

4285

4286 For all of the key-family-specific mappings provided below, SDMX-ML namespace
 4287 modules are identified with the abbreviations used in the standard schemas
 4288 (“compact:” refers to the CompactData module; “common:” to the Common
 4289 namespace module, “utility:” to the UtilityData namespace module; and “cross:” to the
 4290 CrossSectionalData module).

4291

4292 Note that for all of the following mappings the term “concept name” is the value of the
 4293 id attribute of the concept as found in the SDMX-ML message describing the key
 4294 family.

4295

4296 **6.5.2 Representations and Datatypes**

4297 For all key-family-specific schemas, the information about permitted datatypes found
 4298 in the `structure:TextFormat` element for dimensions and attributes (including
 4299 observation values, which are technically an attribute) are represented with a
 4300 standard set of datatypes from W3C XML Schema. The table below shows many of
 4301 these – the first column corresponds to the `structure:TextType` attribute of the
 4302 `structure:TextFormat` element, and the second column shows how this value
 4303 will be represented in the generated schemas.

4304

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

4305

4306

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

4307

4308

4309

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).

4310

4311

4312

4313

4314

4315

4316

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.

4317

4318

4319

4320

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.

4321

4322

4323

4324

4325

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.

4326

4327

4328

4329

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.

4330

4331

4332

4333

4334

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:

4335

4336

4337

4338

4339

minLength: XSD restriction is used to set the minLength value.

4340

maxLength: XSD restriction is used to set the maxLength value.

4341

startValue: XSD restriction is used to set the minInclusive value.

4342

endValue: XSD restriction is used to set the maxInclusive value.

4343

decimals: XSD restriction is used to set the fractionDigits value.

4344

pattern: This is expressed as a pattern facet on the declared simple type.

4345

4346

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

4347



4348 **6.5.3 Use of W3C Schema Extension in XML Mappings**

4349 It is permissible to use schema extension and refinement (that is, xs:extends and
4350 xs:restricts) as well as xs:include to organize a set of SDMX-ML schemas. While this
4351 is not required, it is sometimes useful to organize a large set of similar schemas.

4352

4353 If these techniques are used, however, the resulting XML instance must be identical
4354 to those instances marked-up according to schemas which do not employ them. This
4355 rule includes the use of XML namespaces – that is, they must be identical in all
4356 respects between instances marked-up according to XML schemas which use
4357 xs:extends, xs:restricts, and xs:include, and those which do not.

4358 **6.5.4 Compact Schemas:**

4359 Compact schemas express all attribute and dimension values as XML attributes.
4360 These may be placed at various levels within the imported SDMX "compact"
4361 structure. The key-family-specific schema uses XSD substitution groups to attach
4362 key-family-specific elements and attributes to the structures provided in the
4363 "compact:" namespace.

4364

4365 A global element named "DataSet" will be declared, with an XSD substitutionGroup
4366 attribute which has a value referencing the DataSet element in the "compact:"
4367 namespace. Its type attribute will reference DataSetType in the key-family-specific
4368 namespace.

4369

4370 An XSD complexType will be declared named "DataSetType". It will have XSD
4371 complexContent containing an XSD extension element, with a base attribute of
4372 DataSetType in the "compact:" namespace. The extension will consist of an XSD
4373 choice element, with a minOccurs attribute with a value of "0" and a maxOccurs
4374 value of "unbounded". The choice will contain an XSD element reference for each
4375 named group declared in the key family. They will each have an XSD ref attribute
4376 with a value of the group id provided in the key family. (These elements will take the
4377 names of the group ids declared in the key family.) Additionally, an XSD element will
4378 be declared in the choice with a ref attribute with a value of Series. Further, an
4379 element named Annotations will be added to the choice, with a type of
4380 AnnotationsType from the "common:" namespace.

4381

4382 For each attribute declared in the key family with an attachmentLevel of "DataSet",
4383 an XML attribute will also be declared in the extension. It will have the same name as
4384 the attribute's concept in the key family. It will have a "use" attribute value of
4385 "optional". For coded attributes, the XML attribute will be given a type value which is
4386 the name of the codelist which represents it. In the key-family-specific namespace,
4387 this codelist will be represented by a simpleType declaration which contains a list of
4388 enumerations, equivalent to the values of the codelist, as described in the key family.
4389 These will be extensions of the XSD "string" datatype. The enumerated values will be
4390 the values of the codes. The descriptions of the codes will be placed inside XSD
4391 "documentation" elements, contained in XSD "annotation" elements, which are
4392 themselves contained in the XSD "enumeration" elements as the first instance of the
4393 XSD documentation element. No other text shall occur within this particular instance
4394 of the XSD documentation element, although other XSD documentation elements
4395 may occur within any given XSD enumeration element.

4396

4397 Uncoded attributes will also be represented with XSD simpleType elements declared
4398 in the key-family-specific namespace, with names formed by taking the name of the



4399 attribute in the key family and appending "Type" to them. If unrestricted, these will be
4400 of the W3C XML Schema primitive type "string". Otherwise, bindings will be as
4401 described above in the Representations and Datatypes section.
4402

4403 For each dimension declared in the key family, an XML attribute will be declared, with
4404 a name which is the name of the concept referenced by the dimension. For coded
4405 dimensions, the XML attribute will be given a type value which is the name of the
4406 codelist which represents it. In the key-family-specific namespace, this codelist will
4407 be represented by a simpleType declaration which contains a list of enumerations,
4408 equivalent to the values of the codelist, as described in the key family (and for coded
4409 attributes, above). All data typing with the TextFormat element will be implemented
4410 as provided for attributes, above. The "use" attribute for the dimension XML attribute
4411 declaration will have a value of "optional".
4412

4413 For each named Group in the key family, a global XSD element will be declared,
4414 taking the id of the group. Its XSD type attribute will have a value formed by taking
4415 the name of the element and adding "Type" to the end of it. It will have a
4416 substitutionGroup attribute which references the Group element declared in the
4417 "compact:" namespace.
4418

4419 An XSD complexType will be declared for each named group declared in the key
4420 family, with a name formed by taking the name of the group in the key family and
4421 appending "Type" to it. It will have an XSD complexContent element which contains
4422 an XSD extends with a base attribute value of compact:GroupType. The extends will
4423 contain an XSD sequence element. An element named Annotations will be added to
4424 the end of the sequence, with a type of AnnotationsType from the "common:"
4425 namespace. It will also have a minOccurs value of "0".
4426

4427 For each attribute in the key family with an attachmentLevel of "Group", an XSD
4428 attribute element will be added to the extends element, with a use attribute set to
4429 "optional" and a type attribute defined as for the DataSet level, above. The name will
4430 be the concept name of the attribute in the key family.
4431

4432 For each dimension referenced by DimensionRef element in the named Group
4433 declaration in the key family XML, an XSD attribute element will also be added to the
4434 extends element, with a use attribute set to "required" and a type defined as for
4435 coded attributes for the dataset level, above. The name will be the concept name of
4436 the dimension in the key family.
4437

4438 A XSD global element named Series will be declared in the key-family-specific
4439 namespace, with a type of SeriesType and a substitutionGroup attribute referencing
4440 compact:Series.
4441

4442 An XSD complexType will then be declared with a name of SeriesType. It will have
4443 XSD complexContent, with an XSD extension element that has a base attribute value
4444 of compact:SeriesType. The extends element will contain an XSD sequence
4445 element, which will contain an XSD element with a ref attribute whose value is "Obs".
4446 Its minOccurs attribute will have a value of "0" and a maxOccurs value of
4447 "unbounded". An element named Annotations will be added to the end of the
4448 sequence, with a type of AnnotationsType from the "common:" namespace. It will
4449 also have a minOccurs value of "0".
4450



4451 For each attribute in the key family with an attachmentLevel of “Series”, an XSD
4452 attribute element will be added to the extends element, with a use attribute set to
4453 “optional” and a type attribute defined as for the DataSet level, above. The name will
4454 be the name of the attribute’s concept in the key family. The exception is where an
4455 attribute has an isTimeFormat attribute value of “true” – in this case, it is treated the
4456 same as other series-level attributes except that its use attribute has a value of
4457 “required”.

4458

4459 For each dimension declared in the key family, an XML attribute will be declared, with
4460 a name which is the name of the concept referenced by the dimension. For coded
4461 dimensions, the XML attribute will be given a type value which is the name of the
4462 codelist which represents it. In the key-family-specific namespace, this codelist will
4463 be represented by a simpleType declaration which contains a list of enumerations,
4464 equivalent to the values of the codelist, as described in the key family (and for coded
4465 attributes, above). All data typing with the TextFormat element will be implemented
4466 as provided for attributes, above. The “use” attribute for the dimension attribute
4467 declaration will have a value of “optional”.

4468

4469 An XSD global element will be declared named “Obs”. It will have a
4470 substitutionGroup attribute with a value “compact:Obs”. It will have a type of
4471 “ObsType”.

4472

4473 An XSD complexType element will be declared with a name “ObsType” and an XSD
4474 complexContent. This will contain an XSD extends element with a base attribute of
4475 “compact:ObsType”. It will contain an XSD sequence element. The sequence
4476 element will contain an element named Annotations, with a type of AnnotationsType
4477 from the “common:” namespace. It will have a minOccurs value of “0”.

4478

4479 The extension element will also have an XSD attribute element in it, which will have a
4480 name attribute whose value is the name of the TimeDimension concept from the key
4481 family. It will have a use attribute of “optional” and a type of
4482 “common:TimePeriodType”.

4483

4484 The extension element will also have an XSD attribute element in it, which will have a
4485 name attribute whose value is the concept name of the primary measure from the
4486 key family. It will have a use attribute of “optional” and a type as described for
4487 attributes, above. If the codelist attribute references a codelist, then a simple type
4488 must be declared as indicated above. Otherwise, data typing should be done as for
4489 other constructs using the TextFormat element to describe the data format.

4490

4491 For each attribute declared in the key family with an attachmentLevel of
4492 “Observation”, an XSD attribute will be added to the extends. Each XSD attribute will
4493 take the name of the attribute’s concept declared in the key family, and will have a
4494 use attribute of “optional”. Its type will be defined as for the DataSet-level attributes
4495 described above.

4496

4497 No other declarations or constructs will be added to the schemas created using this
4498 mapping.

4499

4500 **Time Ranges in CompactData:** Unlike any other SDMX-ML data format, the key-
4501 family-specific CompactData format can express a set of observation values without
4502 having to provide a time for each of them. If a Series has a time provided for the first

4503 observation, subsequent observations in the series may omit the time, and only
4504 provide an observation value (a value for the attribute named after the primary
4505 measure), and whatever attributes are needed (see below). The times of the
4506 subsequent observations can be calculated according to the frequency specified by
4507 the relevant time format attribute value (or, failing that, the frequency dimension
4508 value), which can be calculated by the application. Note that support for this
4509 functionality is not mandatory for applications which do not claim this support in their
4510 conformance statements. It is also permissible to supply a time value for the last
4511 observation in the series, to permit double-checking of the calculation, although this
4512 is not mandatory.

4513

4514 **Delete and Update Messages in CompactData:** In the Header element and action
4515 attribute at the DataSet level, the action field specifies whether a message is an
4516 update message (to append or replace) or a delete message for the purposes of
4517 bilateral exchange. If it is an update message, it is used to send new information or
4518 updated information, which may include only data, only documentation (that is,
4519 attribute values as described in the key family), or both. (Agreements regarding the
4520 use of update messages should be specified between counterparties.) For a delete
4521 message, the requirements are that a complete series key always be sent for the
4522 deletion of data, which is identified either as an entire series by the absence of any
4523 specified time periods, or for a specific set of time periods, by the inclusion of those
4524 time periods. Attribute values may be deleted by sending a complete or partial set of
4525 attributes, with any valid value for the attribute (according to the XSD schema) being
4526 taken as an indication that the current attribute value should be deleted.

4527

4528 6.5.5 Cross-Sectional Schemas

4529 Key-family-specific cross-sectional schemas express all non-time-series-based
4530 presentations of the data which are made possible in the key family. They also are
4531 capable of expressing statistical data for which time is not a concept – that is, they
4532 can provide the only SDMX-ML format for data which is inherently only cross-
4533 sectional. As with the CompactData format, key values and attribute values are
4534 attached to a four-level structure as XML attributes. For cross-sectional data,
4535 however, the term “Series” – an abbreviation of “time series” – is replaced by the
4536 equivalent “Section” construct.

4537

4538 Please note that named groups declared in the key family are ignored for the
4539 purposes of the cross-sectional data format. They are replaced by a generic Group
4540 element, leaving it up to the writing or processing application to enforce the validity of
4541 attribute values for groups of Sections. This is true also because a single SDMX-ML
4542 cross-sectional schema may be described in the key family such that it allows for
4543 more than one dimension to be expressed at the observation level, replacing the role
4544 of time in time-series-oriented formats, and therefore allows key values and attribute
4545 values to be attached at more than one level.

4546

4547 A global element named “DataSet” will be declared, with an XSD substitutionGroup
4548 attribute which has a value referencing the DataSet element in the “cross:”
4549 namespace. Its type attribute will reference DataSetType in the key-family-specific
4550 namespace.

4551

4552 An XSD complexType will be declared named “DataSetType”. It will have XSD
4553 complexContent containing an XSD extension element, with a base attribute of



4554 DataSetType in the “cross:” namespace. The extension will consist of an XSD
4555 choice element, with a minOccurs of “0” and a maxOccurs of “unbounded”. The choice
4556 element will contain an XSD element reference with a value of “Group”. Additionally,
4557 an XSD element will be declared in the choice with a ref attribute, whose value is
4558 Section. Further, an element named Annotations will be added to the choice, with a
4559 type of AnnotationsType from the “common:” namespace. It will have a minOccurs
4560 attribute of “0”.

4561

4562 For each attribute or dimension declared in the key family with a
4563 crossSectionalAttachDataSet of “true”, an XML attribute will also be declared in the
4564 extension. It will have the same name as the attribute concept or dimension concept
4565 in the key family. It will have a “use” attribute value of “optional”. For coded attributes
4566 and dimensions, the XML attribute will be given a type value which is the name of the
4567 codelist which represents it. In the key-family-specific namespace, this codelist will
4568 be represented by a simpleType declaration which contains a list of enumerations,
4569 equivalent to the values of the codelist, as described in the key family. These will be
4570 extension of the XSD “string” datatype. The enumerated values will be the values of
4571 the codes. The descriptions of the codes will be placed inside XSD “documentation”
4572 elements, contained in XSD “annotation” elements, which are themselves contained
4573 in the XSD “enumeration” elements as the first instance of the XSD documentation
4574 element. No other text shall occur within this particular instance of the XSD
4575 documentation element, although other XSD documentation elements may occur
4576 within any given XSD enumeration element.

4577

4578 Un-coded dimensions will have XML attributes declared as above, but will have their
4579 data-typing mapped differently. All data typing with the TextFormat element will be
4580 implemented as provided for uncoded attributes, below, with the exception that
4581 Timespan is not permitted as the representation of a dimension.

4582

4583 Uncoded attributes will also be represented with XSD simpleType elements declared
4584 in the key-family-specific namespace, with names formed by taking the name of the
4585 attribute concept in the key family and appending “Type” to them. If unrestricted,
4586 these will be of the W3C XML Schema primitive type “string”; otherwise, mappings
4587 will be as per the Representations and Datatypes section, above. If the textType of
4588 an attribute value is a Timespan, then two attributes will be declared – one as per
4589 usual, which will be of type xs:duration, and the other will have a name value of the
4590 attribute’s concept with “StartTime” appended to it, and it will have a value of
4591 “xs:duration”.

4592

4593 A Global XSD element will be declared named Group. Its XSD type attribute will have
4594 a value of GroupType. It will have a substitutionGroup attribute which references the
4595 Group element declared in the “cross:” namespace.

4596

4597 An XSD complexType named GroupType will be declared. It will have an XSD
4598 complexContent element which contains an XSD extends with a base attribute value
4599 of compact:GroupType. The extends will contain an XSD sequence element, which
4600 will contain an XSD element with a reference to the element Section. Its minOccurs
4601 attribute will have a value of “0” and a maxOccurs value of “unbounded”. An element
4602 named Annotations will be added to the end of the sequence, with a type of
4603 AnnotationsType from the “common:” namespace. It will also have a minOccurs
4604 value of “0”.

4605



4606 For each attribute or dimension in the key family with a crossSectionalAttachGroup
4607 value of “true” or an isFrequencyDimension value of “true”, an XSD attribute element
4608 will be added to the extends element, with a use attribute set to “optional” and a type
4609 attribute defined as for the DataSet level, above. The name will be the name of the
4610 attribute concept or dimension concept in the key family.

4611

4612

4613 A XSD global element named Section will be declared in the key-family-specific
4614 namespace, with a type of SectionType and a substitutionGroup attribute referencing
4615 compact:Section.

4616

4617 An XSD complexType will then be declared with a name of SectionType. It will have
4618 XSD complexContent, with an XSD extension element that has a base attribute value
4619 of cross:SectionType. The extends element will contain an XSD choice element with
4620 a minOccurs of “0” and a maxOccurs of “unbounded”, which will contain an XSD
4621 element for each CrossSectionalMeasure declared in the key family, with a ref
4622 attribute whose value is the name of the measure’s concept. An element named
4623 Annotations will be added to the end of the choice, with a type of AnnotationsType
4624 from the “common:” namespace.

4625

4626 For each attribute or dimension in the key family with a crossSectionalAttachSection
4627 value of “true”, an XSD attribute element will be added to the extends element, with a
4628 use attribute set to “optional” and a type attribute defined as for the DataSet level,
4629 above. The name will be the name of the attribute concept or dimension concept in
4630 the key family.

4631

4632 An XSD global element will be declared for each CrossSectionalMeasure declared in
4633 the key family, with the name of the measure’s concept. It will have a
4634 substitutionGroup attribute with a value “cross:Obs”. It will have a type of “ObsType”.
4635 If no CrossSectionalMeasures have been declared, use the PrimaryMeasure instead.

4636

4637 An XSD complexType element will be declared for each CrossSectionalMeasure
4638 declared in the key family with a name created by appending “Type” to the concept of
4639 the measure. These declarations will contain an XSD complexContent. This will
4640 contain an XSD extends element with a base attribute of “cross:ObsType”. It will
4641 contain an XSD sequence element. The sequence element will contain an element
4642 named Annotations, with a type of AnnotationsType from the “common:” namespace.
4643 It will have a minOccurs value of “0”.

4644

4645 The extension element will also have an XSD attribute element in it for each attribute
4646 or dimension which has a crossSectionalAttachObservation value of “true” and lists
4647 the name of the measure’s concept in an AttachmentMeasure element in its
4648 declaration. The XSD attribute will take its name value from the name of the
4649 attribute’s concept. It will have a use attribute of optional, and a type as described for
4650 the DataSet level, above. Additionally, an attribute will be declared with a name of
4651 “value” and a type created by appending the string “SimpleType” to the name of the
4652 containing Measure. Its use attribute will be “optional”. (Note that the dimension
4653 whose coded representation corresponds to the CrossSectionalMeasures should
4654 never have its crossSectionalAttachObservation attribute set to “true”.) For each of
4655 the “value” attributes, a global XSD simple type will be declared, with a name created
4656 by appending “SimpleType” to the Measure corresponding to the value attribute. The
4657 base of the simple type will be xs:string. The type will be restricted as per the



4658 Representations and datatypes section, above, with the one exception that if the type
4659 of any given attribute is “Timespan,” an additional attribute will be declared as a
4660 sibling to the “value” attribute, with a name of “startTime”, a value of xs:dateTime,
4661 and a use attribute of optional. The type of the generated simple type will in this case
4662 be “xs:duration”.

4663

4664 If no CrossSectionalMeasures were declared in the key family, there will be an XSD
4665 attribute element added to the extension, which will have a name attribute whose
4666 value is the concept name of the PrimaryMeasure concept from the key family. It will
4667 have a use attribute of “optional” and a type value as described for attributes and
4668 dimensions.

4669

4670 In this case, for each attribute declared in the key family with an attachmentLevel of
4671 “Observation”, an XSD attribute will be added to the extends. Each XSD attribute will
4672 take the name of the attribute’s concept declared in the key family, and will have a
4673 use attribute of “optional”. Its type will be defined as for the DataSet-level attributes
4674 described above. Additionally, an attribute will be declared with a name of value and
4675 a type value as described for attributes and dimensions. Its use attribute is “optional”.

4676

4677 No other declarations or constructs will be added to the schemas created using this
4678 mapping.

4679

4680 **Delete and Messages in CrossSectionalData:** In the Header element and in the
4681 action attribute at DataSet level, the action field specifies whether a message is an
4682 update message (Append, Replace) or a delete message for the purposes of bilateral
4683 exchange. If it is an update message, it is used to send new information or updated
4684 information, which may include only data, only documentation (that is, attribute
4685 values as described in the key family), or both. (Agreements regarding the use of
4686 update messages should be specified between counterparties.) For a delete
4687 message, the requirements are that a complete key always be sent for the deletion of
4688 data, which is identified either as an entire series by the absence of any specified
4689 time periods, or for a specific set of time periods, by the inclusion of those time
4690 periods. Attribute values may be deleted by sending a complete or partial set of
4691 attributes, with any valid value for the attribute (according to the XSD schema) being
4692 taken as an indication that the current attribute value should be deleted.

4693

4694 **6.5.6 Utility Schemas**

4695 Utility schemas are different from the Compact and Cross-Sectional schemas
4696 because they differentiate between the expression of the attributes and dimensions
4697 established in the key family. This design serves to preserve the ordering of the keys
4698 - the design provides much of the key-family structural metadata without requiring the
4699 processor to access the XML structure message describing the key family. This
4700 makes the rules inherent in the structure of the key family available to such tools as
4701 schema-guided XML editors, which are part of the primary reason for the Utility
4702 schema format.

4703

4704 The Utility schema employs a technique similar to the Compact and Cross-Sectional
4705 schemas by creating substitution groups which are headed by elements at the
4706 DataSet, Group, Series, and Observation levels. This is done in such a way that the
4707 messages can be more completely validated with a generic XML parser but are
4708 considerably larger in size than the CompactData or CrossSectionalData formats.



4709

4710 A global element named "DataSet" will be declared, with an XSD substitutionGroup
4711 attribute which has a value referencing the DataSet element in the "utility:"
4712 namespace. Its type attribute will reference DataSetType in the key-family-specific
4713 namespace.

4714

4715 An XSD complexType will be declared named "DataSetType". It will have XSD
4716 complexContent containing an XSD extension element, with a base attribute of
4717 DataSetType in the "utility:" namespace. The extension will consist of an XSD
4718 sequence element containing first an XSD choice element, with a maxOccurs value
4719 of "unbounded". The choice will contain an XSD element reference for each named
4720 group declared in the key family. They will each have an XSD ref attribute with a
4721 value of the group name provided in the key family. (These elements will take the
4722 names of the groups declared in the key family.) If there are no named groups
4723 declared in the key family, an XSD element will be declared in the choice with a ref
4724 attribute with a value of Series. An element named Annotations will be added to the
4725 end of the sequence, with a type of AnnotationsType from the "common:" namespace
4726 and a minOccurs attribute of "0".

4727

4728 For each attribute declared in the key family with an attachmentLevel of "DataSet",
4729 an XML attribute will be declared in the extension. It will have the same name as the
4730 attribute's concept in the key family. It will have a use attribute with a value of
4731 "required" if the attribute declared in the key family has an assignmentStatus of
4732 "Mandatory", and a use attribute with a value of optional if its assignmentStatus in the
4733 key family is "Conditional". For coded attributes, the XML attribute will be given a type
4734 value which is the id of the codelist which represents it. In the key-family-specific
4735 namespace, this codelist will be represented by a simpleType declaration which
4736 contains a list of enumerations, equivalent to the values of the codelist, as described
4737 in the key family. These will be extension of the XSD "string" datatype. The
4738 enumerated values will be the values of the codes. The descriptions of the codes will
4739 be placed inside XSD "documentation" elements, contained in XSD "annotation"
4740 elements, which are themselves contained in the XSD "enumeration" elements as
4741 the first instance of the XSD documentation element. No other text shall occur within
4742 this particular instance of the XSD documentation element, although other XSD
4743 documentation elements may occur within any given XSD enumeration element.

4744

4745 Uncoded attributes will also be represented with XSD simpleType elements declared
4746 in the key-family-specific namespace, with names formed by taking the name of the
4747 attribute's concept in the key family and appending "Type" to them. If unrestricted,
4748 these will be of the W3C XML Schema primitive type "string"; any restrictions as
4749 described in a TextFormat element will be implemented as per the Representations
4750 and Datatypes section, above. If any attribute is described in the TextFormat element
4751 as having a textType of "Timespan", then an additional attribute will be added to the
4752 extension with a name formed by taking the concept name of the attribute and
4753 appending "StartTime" to it. This attribute will have a type of "xs:dateTime"; the
4754 primary attribute will be given a type of "xs:duration".

4755

4756 For each named Group in the key family, a global XSD element will be declared,
4757 taking the name of the group. Its XSD type attribute will have a value formed by
4758 taking the name of the element and adding "Type" to the end of it. It will have a
4759 substitutionGroup attribute which references the Group element declared in the
4760 "utility:" namespace.

4761

4762 An XSD complexType will be declared for each named group declared in the key
4763 family, with a name formed by taking the name of the group in the key family and
4764 appending “Type” to it. It will have an XSD complexContent element which contains
4765 an XSD extends with a base attribute value of utility:GroupType. The extends will
4766 contain an XSD sequence element, which will contain an XSD element with a
4767 reference to the element Series. Its maxOccurs attribute will have a value of
4768 “unbounded”. An element named Annotations will be added to the end of the
4769 sequence, with a type of AnnotationsType from the “common:” namespace. It will
4770 also have a minOccurs value of “0”.

4771

4772 For each attribute in the key family with an attachmentLevel of “Group”, an XSD
4773 attribute element may be added to the extends element for any given group. To
4774 determine if a declared Group-level attribute in the key family is to be added to a
4775 particular named group XSD type, look at the AttachmentGroup elements in the XML
4776 of the key family. If the group element in the key-family-specific schema that is being
4777 declared appears in an AttachmentGroup element in the key family XML, then the
4778 attribute should be included in the utility schema being created. If added, this
4779 attribute should be declared as defined for the DataSet level, above. The name will
4780 be the name of the attribute’s concept in the key family.

4781

4782 A XSD global element named Series will be declared in the key-family-specific
4783 namespace, with a type of SeriesType and a substitutionGroup attribute referencing
4784 utility:Series.

4785

4786 An XSD complexType will then be declared with a name of SeriesType. It will have
4787 XSD complexContent, with an XSD extension element that has a base attribute value
4788 of utility:SeriesType. The extends element will contain an XSD sequence element,
4789 which will contain first an XSD element whose ref value is “Key”. This is followed by
4790 an XSD element with a ref attribute whose value is “Obs”. Its maxOccurs attribute
4791 will have a value of “unbounded”. An element named Annotations will be added to
4792 the end of the sequence, with a type of AnnotationsType from the “common:”
4793 namespace. It will also have a minOccurs value of “0”.

4794

4795 For each attribute in the key family with an attachmentLevel of “Series”, an XSD
4796 attribute element will be added to the extends element, with name, use, and type
4797 attributes defined as for the DataSet level, above.

4798

4799 A global XSD element named Key will be declared. It will have a type of KeyType,
4800 and a substitutionGroup attribute with a value of utility:Key.

4801

4802 An XSD complexType will be declared, with a name of KeyType. It will have an XSD
4803 complexContent element with an XSD extends element inside it, whose base
4804 attribute will have a value of “utility:KeyType”. The extends element will contain a
4805 XSD sequence of elements, one for each non-time dimension declared in the key
4806 family, in the order in which they appear in the XML for the key family. These
4807 elements will have names that are the same as the dimension’s concepts in the key
4808 family which they represent. Their type attributes will be the names of simpleTypes
4809 created exactly as for attributes at the DataSet level, above, with some additional
4810 mapping rules. For Time dimensions and non-observational time dimensions, the
4811 type will be set to “common:TimePeriodType”. For count dimensions, the type will be
4812 set to “xs:integer”. For entity dimensions, the type will be set to “xs:string”. All data

4813 typing with the TextFormat element will be implemented as provided for uncoded
4814 attributes, below, and in the general rules regarding this mapping.

4815
4816

4817 An XSD global element will be declared named “Obs”. It will have a
4818 substitutionGroup attribute with a value “utility:Obs”. It will have a type of “ObsType”.

4819

4820 An XSD complexType element will be declared with a name “ObsType” and an XSD
4821 complexContent. This will contain an XSD extends element with a base attribute of
4822 “compact:ObsType”. It will contain an XSD sequence element. The sequence
4823 element will contain an element whose name is the name of the TimeDimension
4824 concept from the key family, with a type of common:TimePeriodType. It will be
4825 followed by an element whose name is the name of the PrimaryMeasure declared in
4826 the key family, with a type created as for other attribute and dimension values. If the
4827 Primary Measure was described in a TextFormat element as being of textType
4828 “Timespan”, another element will be declared with a name of “ObsStartTime”, and it
4829 will have a declared type of “xs:dateTime”. The declared type of the primary measure
4830 element will be “xs:duration”. Last is an element named Annotations, with a type of
4831 AnnotationsType from the “common:” namespace. It will have a minOccurs value of
4832 “0”.

4833

4834 For each attribute declared in the key family with an attachmentLevel of
4835 “Observation”, an XSD attribute will be added to the extends. Each XSD attribute will
4836 take the name of the attribute’s concept declared in the key family, and will have a
4837 use attribute, name, and type created as defined as for the DataSet-level attributes
4838 described above.

4839

4840 No other declarations or constructs will be added to the schemas created using this
4841 mapping.

4842

4843 **Note:** The UtilityData key-family-specific schema does not have any mechanism for
4844 expressing time ranges across a set of observation values. The only permissible
4845 message for this schema type is an “update” message containing a complete set of
4846 attributes and observation values for the transmitted series. There is no concept of a
4847 “delete” message, and the action field in the message Header element is ignored if
4848 specified.

4849 **6.6 Mappings to Metadata Structure Definition-Specific Metadata** 4850 **Schemas**

4851 **6.6.1 General Rules**

4852 For all metadata-structure-specific schemas SDMX provides a namespace to be
4853 used as the extension base: `SDMXMetadatReport.xsd` The metadata-structure-
4854 specific schema will be created in its own target name space, owned and maintained
4855 by the creating agency. It will use the targetNamespace attribute of the schema
4856 element to identify the namespace which contains the metadata-structure-specific
4857 schema. The `SDMXMetadatReport.xsd` namespace module provided by SDMX
4858 will be incorporated using the import element in the key-family-specific schema. The
4859 `SDMXCommon.xsd` namespace module must also be imported into the schema.
4860 Other `xml:namespace` attributes may be added to the schema element as needed.

4861



4862 The elementFormDefault attribute on the schema element will be given a value of
4863 "qualified", and the attributeFormDefault attribute on the schema element will be
4864 given a value of "unqualified".
4865

4866 **6.6.2 Use of W3C Schema Extension in XML Mappings**

4867 These rules for Metadata Schemas are identical to those given for Data Schemas
4868 above.

4869 **6.6.3 Attribute and Observation Values**

4870
4871 In many places, the TextFormat element is used in the SDMX Structure message to
4872 describe a data type in the schema. This is identical to the Representations and
4873 Datatypes section above.

4874 **6.6.4 Metadata Report**

4875
4876 In the MetadataReport namespace, a global element will be declared with the name
4877 MetadataSet. This element declaration will have a substitutiongroup attribute with the
4878 value "metadatareport:MetadataSet", and will have a type of "MetadataSetType".
4879

4880 A complex type will be declared with the name "MetadataSetType", and it will contain
4881 a complexContent element. Inside of this will be an extension element with a base
4882 attribute value of "metadatareport:MetadataSetType". Inside of this will be a
4883 sequence element. For each ReportStructure element in the
4884 MetadataStructureDefinition, there will be an element declared which has the name
4885 of the id attribute of each report structure. These elements will have type values
4886 created by appending the string "Type" to the end of these id values. These elements
4887 will have a minOccurs attribute with a value of "0" and a maxOccurs attribute with a
4888 value of "unbounded".
4889

4890 For each ReportStructure element, a complex type is declared with a name value
4891 created by appending the string "Type" to the end of the value of its id attribute. Each
4892 of these types will contain a sequence element. Inside this sequence element, an
4893 XSD element is declared with a name of the id attribute with "Target" appended to it,
4894 with a type value named by taking the id value and appending "TargetType" to it. It
4895 has no minOccurs or maxOccurs attributes.
4896

4897 For each top-level MetadataAttribute element in the metadata structure definition,
4898 there will be an element declaration after the "Target" element declaration. Each
4899 report-structure type only has element declarations for the top-level
4900 MetadataAttributes which it contains. These elements which correspond to the top-
4901 level MetadataAttribute elements will be named after the values of the conceptRef
4902 attributes of each one. If the usageStatus attribute has a value of "Conditional," then
4903 the element declaration has a minOccurs attribute with a value of "0". Each element
4904 will have a type value which has a value created by appending the string "Type" to
4905 the value of the conceptRef attribute.
4906

4907 For each type created by appending "Type" to the conceptRef attribute value, for
4908 each of its child MetadataAttributes an element and type will be declared, following
4909 the pattern for the top-level element, recursively. There will be no target types
4910 declared, however.



4911

4912 If the representationScheme attribute for any MetadataAttribute is used, then the
4913 declaration of that MetadataAttribute's type is changed: the type value will be set to a
4914 value created by appending the string "CodeType" to the value of the
4915 MetadataAttribute's conceptRef field, and a simpleType declaration which has that
4916 name will also be declared. This will contain a restriction element with a base
4917 attribute with a value of "xs:NMTOKEN", and the values of the codelist referenced by
4918 the MetadataAttributes representationScheme and representationSchemeAgency
4919 attributes will each be represented by an enumeration element. The value attribute of
4920 each enumeration element will contain the code value, and the code description will
4921 be contained in a documentation element inside an annotation element, which will
4922 form the contents of the enumeration element.

4923

4924 If a MetadataAttribute contains a TextFormat element, then a simple type is declared
4925 as above, but instead of having an enumeration, it is mapped to the schema as per
4926 the TextFormat bindings provided above for Key-Family-Specific schemas. If neither
4927 the representationScheme attribute nor a TextFormat child element is present, then
4928 the default representation of the referenced concept should be used, as provided in
4929 the ConceptScheme.

4930

4931 For each ReportStructure element, a complex type will be declared which has the
4932 name of the ReportStructure id attribute with "TargetType" appended to it. This
4933 complex type will contain a sequence element. For each IdentifierComponent or
4934 IdentifierComponentRef element present in the FullTargetIdentifier or
4935 PartialTargetIdentifier referenced by the ReportStructure in its target attribute, an
4936 element will be declared, which will have a name composed of the contents of the
4937 corresponding id attribute of the IdentifierComponent (as referenced by the
4938 IdentifierComponentRef in the case of PartialTargetIdentifiers) with "Target"
4939 appended to it. The type of each such element will have a value which is the name
4940 value plus the string "Type". For each of these, a simple type will be declared, with
4941 the type name as formulated, which has a value derived from the representation of
4942 the concept as provided for MetadataAttributes. Duplicate type declarations are to be
4943 avoided. When name collisions occur, they should be resolved by pre-pending the
4944 relevant agency code to the name. For those attributes which are described in a
4945 TextFormat element as being of textType "Timespan", a second element will be
4946 made available with the same cardinality as the first: this will be named by appending
4947 "StartTime" to the attribute's concept name. It will be of type "xs:dateTime", and the
4948 original value element will be of type "xs:duration".

4949

4950 **7 APPENDIX: SAMPLE SDMX-ML DATA MESSAGES**

4951 This appendix is presented to provide example layouts for some of the simpler
4952 SDMX-ML sample data files, allowing them to be more easily understood. For each
4953 sample data file, one or more tables are offered, to show how the data itself might be
4954 formatted. Please note that all data is fictitious, and used for demonstration purposes
4955 only. (Numbers are not consistent across samples, but are randomly generated.)

4956

4957 **7.1 CompactSample.xml**

4958 **ID:** Message JD014 (Untruncated Test Message)

4959 **Name:** Trans46305



4960 **Prepared:** 2001-03-11T09:30:47-05:00

4961 **Sent by:** GB Smith from the BIS, +000.000.0000

4962 **To:** B.S. Featherstone, Statistics Division, ECB, +000.000.0001

4963

4964 This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

4965

4966 **External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks**
4967 **in Millions of US Dollars, Monthly at the beginning of period. (Free data)**

4968

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

4969

4970 **External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks**
4971 **in Millions of US Dollars, Annually at the beginning of period. (Free data)**

4972

Time	Data
2000-01	3.14

4973

4974 **External Debt, All Maturities, Debt Securities Issued Abroad for Mexico,**
4975 **expressed as Stocks in Millions of US Dollars, Monthly at the beginning of**
4976 **period. (Free data)**

4977

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

4978

4979 **External Debt, All Maturities, Debt Securities Issued Abroad for Mexico,**
4980 **expressed as Stocks in Millions of US Dollars, Annually at the beginning**
4981 **of period. (Free data)**

4982

Time	Data
2000-1	4.14

4983

4984 **7.2 UtilitySample.xml**4985 **ID:** Message JD01678594 (Untruncated Test Message)4986 **Name:** Trans463044987 **Prepared:** 2001-03-11T09:30:47-05:004988 **Sent by:** GB Smith from the BIS, +000.000.00004989 **To:** B.S. Featherstone, Statistics Division, ECB, +000.000.0001

4990

4991 This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

4992

4993 **External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks**
4994 **in Millions of US Dollars, Monthly at the beginning of period. (Free data)**

4995

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

4996

4997 **7.3 GenericSample.xml**4998 **ID:** Message JD014 (Untruncated Test Message)4999 **Name:** Trans463025000 **Prepared:** 2001-03-11T09:30:47-05:005001 **Sent by:** GB Smith from the BIS, +000.000.00005002 **To:** B.S. Featherstone, Statistics Division, ECB, +000.000.0001

5003

5004 This message contains new data, and was created at 2001-03-11T09:30:47-
5005 05:00.

5006

5007 **External Debt, All Maturities, Bank Loans for Mexico, expressed as Stocks**
5008 **in Millions of US Dollars, Monthly at the beginning of period. (Free data)**

5009

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



5010
5011

5012 **7.4 CrossSectionalSample.xml**

5013 **ID:** Message BIS947586 (Untruncated Test Message)

5014 **Name:** Trans46305

5015 **Prepared:** 2001-03-11T09:30:47-05:00

5016 **Sent by:** GB Smith from the BIS, +000.000.0000

5017 **To:** B.S. Featherstone, Statistics Division, ECB, +000.000.0001

5018

5019 This message contains new data, and was created at 2001-03-11T09:30:47-05:00.

5020

5021 **External Debt for Mexico, in Millions of US Dollars, at the beginning of**
5022 **period for 2000. (Free data)**

5023

5024	Topic	Stocks	Flows
5025	All Maturities, Bank Loans	3.14	1.00
5026	All Maturities, Debt Securities Issued Abroad	6.39	2.27
5027	All Maturities, Brady Bonds	2.34	-1.00
5028	All Maturities, Non-Bank Trade Credits	3.19	-1.06