SDMX Standards: Section 3A PaRT IV

SDMX-ML:

Schema and Documentation

Data and Metadata

Namespaces

(Version 3.0)

September 2021

Contents

[1 Introduction 1](#_Toc82183453)

[2 Schema Documentation 2](#_Toc82183454)

[2.1 Structure Specific Data Namespace 2](#_Toc82183455)

[2.1.1 Summary 2](#_Toc82183456)

[2.1.2 Complex Types 2](#_Toc82183457)

[2.2 Generic Metadata Namespace 11](#_Toc82183458)

[2.2.1 Summary 11](#_Toc82183459)

[2.2.2 Global Elements 12](#_Toc82183460)

[2.2.3 Complex Types 12](#_Toc82183461)

[3 Mapping to Structure-Specific Schemas 19](#_Toc82183462)

[3.1 General 19](#_Toc82183463)

[3.1.1 Basic Terminology 19](#_Toc82183464)

[3.2 Namespace Rules 21](#_Toc82183465)

[3.3 General Rules 21](#_Toc82183466)

[3.3.1 Component Name Determination 21](#_Toc82183467)

[3.3.2 Representation Determination 21](#_Toc82183468)

[3.3.3 Simple / Primitive Type Determination 21](#_Toc82183469)

[3.3.4 Representation with Codelist Enumeration 22](#_Toc82183470)

[3.3.5 Representation with Value List Enumeration 22](#_Toc82183471)

[3.3.6 Representation with Simple Text Format 22](#_Toc82183472)

[3.3.7 Representation with Complex Text Format 25](#_Toc82183473)

[3.3.8 Type Names 25](#_Toc82183474)

[3.3.9 Type Reuse 26](#_Toc82183475)

[3.4 Data Structure Specific Schema 26](#_Toc82183476)

[3.4.1 DataSetType 26](#_Toc82183477)

[3.4.2 GroupType 27](#_Toc82183478)

[3.4.3 SeriesType 28](#_Toc82183479)

[3.4.4 AttsType 29](#_Toc82183480)

[3.4.5 ObsType 29](#_Toc82183481)

[3.4.6 CompType 30](#_Toc82183482)

[4 Special data functions 30](#_Toc82183483)

[4.1 Updates 30](#_Toc82183484)

[4.2 Deletes 31](#_Toc82183485)

# Introduction

The first change in the data and metadata message is one of terminology. In order to foster consistency in the standard, the names and namespaces of the data and metadata message have been changed. The namespaces now have a uniform format of /data/format and /metadata/format, where format is either generic or structured (i.e. the structure specific formats). This also applies to the message names as well, where the names follow the pattern of FormatData (e.g. StructuredData and GenericMetadata).

The various data messages which existed in the standard have been harmonised into two general formats; generic and structured data. The generic data is much like it has been previously, only slightly modified to conform with the harmonised view on data. The structured data combined the principles of the previous versions compact and cross sectional formats into one more generalised format.

The major shift from the previous version is now that all data can be exchanged as either an un-grouped collection of observations, each specifying a full key or it can be exchanged as data grouped into series with any single dimension placed at the observation level. This in effect, combined the abilities of the time series and cross sectional formats in the a single data format. This format also allows for multiple measure to be expressed when the observation dimension is the measure dimension defined in the data structured. The key differences from the previous versions is that this all is possible in a single data structure definition specific format, as well as being possible in the generic format as well. The goal was to make the structures as homogenous as possible.

Another shift from the previous version in the manner in which the base data structure specific format is defined. In previous versions there was not structure defined, so one had to understand all of the requirements and read the specification in order to understand what could be expected in any structure specific format. In this version, that issue has been addressed. The base structures now impose a strict format on the data structure specific schemas. This is achieved much like the referencing structure in the common namespace through the use of unqualified elements. By the elements not existing in a namespace, the structure specific schemas can place the necessary restrictions of them while still being forced to adhere to prescribed structure. This means that not only are the structures able to perform the validation that is required, but that the messages will be much simpler to process as the format will always use the same element names.

Finally, in order to allow for systems which wish to not process the new more flexible data format, time series only variations of both the generic and structured data sets exist. It is important to note that these structures are derived via restriction from the more generalised format. This means that a data set in the general format with the time dimension at the observation level will have exactly the same content as a time series only data set. The result of this is that there is no additional burden for processing the time series specific format it a system can process the more generalised format. These time series only data sets allow for time series only data messages to exist. These messages make it simple for system which, in the previous version of the standard, only used the time series formats to continue to do so.

These same principles have been applied to the reference data messages as well. In the previous version of the standard there were major differences between the generic and metadata structure-specific formats; some of the differences caused some metadata sets were incompatible between versions. In this version the structures of the generic and metadata structure specific formats have been harmonised to the point where they are nearly identical. Yet, the structured format still provides the strong validation against the metadata structure that is intended. And, as with the data, the base metadata structure specific format now imposes a stricter structure on the generated schemas, making the structure specific instance simpler to process in a generic manner.

# Schema Documentation

## Structure Specific Data Namespace

**http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/data/structurespecific**

### Summary

Referenced Namespaces:

| **Namespace** | **Prefix** |
| --- | --- |
|  |  |
| http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/common | common |
| http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/metadata/generic | metadata |
| http://www.w3.org/2001/XMLSchema | xs |

Contents:

6 Complex Types

### Complex Types

***DataSetType*:** DataSetType is the abstract type which defines the base structure for any data structure definition specific data set. A derived data set type will be created that is specific to a data structure definition and the details of the organisation of the data (i.e. which dimension is the observation dimension). Data is organised into either a collection of series (grouped observations) or a collection of un-grouped observations. The derived data set type will restrict this choice to be either grouped or un-grouped observations. If this dimension is "AllDimensions" then the derived data set type must consist of a collection of un-grouped observations; otherwise the data set will contain a collection of series with the observations in the series disambiguated by the specified dimension at the observation level. This data set is capable of containing data (observed values) and/or documentation (data and metadata attribute values) and can be used for incremental updates and deletions (i.e. only the relevant updates or deletes are exchanged). It is assumed that each series or un-grouped observation will be distinct in its purpose. For example, if series contains both data and documentation, it assumed that each series will have a unique key. If the series contains only data or only documentation, then it is possible that another series with the same key might exist, but with not with the same purpose (i.e. to provide data or documentation) as the first series. This base type is designed such that derived types can be processed in a generic manner; it assures that data structure definition specific data will have a consistent structure. The group, series, obs, and atts elements are unqualified, meaning that they are not qualified with a namespace in an instance. This means that in the derived data set types, the elements will always be the same, regardless of the target namespace of the schemas which defines these derived types. This allows for consistent processing of the structure without regard to what the namespace might be for the data structure definition specific schema.

Derivation:

*AnnotableType* (extension)   
   *DataSetType*

Attributes:

structureRef, setID?, action?, reportingBeginDate?, reportingEndDate?, validFromDate?, validToDate?, publicationYear?, publicationPeriod?

Content:

Annotations?, DataProvider?, (Atts | Group | Series | Obs)\*, Metadata?

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| structureRef | xs:IDREF | The structureRef contains a reference to a structural specification in the header of a data or reference metadata message. The structural specification details which structure the data or reference metadata conforms to, as well as providing additional information such as how the data is structure (e.g. which dimension occurs at the observation level for a data set). |
| setID | IDType | The setID provides an identification of the data or metadata set. |
| action | ActionType | The action attribute indicates whether the file is appending, replacing, or deleting. |
| reportingBeginDate | BasicTimePeriodType | The reportingBeginDate indicates the inclusive start time of the data reported in the data or metadata set. |
| reportingEndDate | BasicTimePeriodType | The reportingEndDate indicates the inclusive end time of the data reported in the data or metadata set. |
| validFromDate | xs:dateTime | The validFromDate indicates the inclusive start time indicating the validity of the information in the data or metadata set. |
| validToDate | xs:dateTime | The validToDate indicates the inclusive end time indicating the validity of the information in the data or metadata set. |
| publicationYear | xs:gYear | The publicationYear holds the ISO 8601 four-digit year. |
| publicationPeriod | ObservationalTimePer iodType | The publicationPeriod specifies the period of publication of the data or metadata in terms of whatever provisioning agreements might be in force (i.e., "Q1 2005" if that is the time of publication for a data set published on a quarterly basis). |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| DataProvider | DataProviderReferenc eType | DataProvider contains a reference to the provider for the data set. |
| Atts | *AttsType* | Atts contains a set of data or metadata attribute values with an attachment level of none (i.e. data set level) or reported against a partial set of dimension values. |
| Group | *GroupType* | Group contains a references to a defined group in the data structure definition along with its key (if necessary) and values for the attributes which are associated with the group. An attribute is associated to a group by either an explicit group relationship or by a group attachment when the attribute has a relationship with a dimension which is a member of this group. |
| Series | *SeriesType* | Series contains a collection of observations that share a common key (set of dimension values). The key of a series is every dimension defined in the data structure definition, save the dimension at the observation level. In addition to the key and observations, the series contains values for data and metadata attributes which have a relationship with any dimension that is part of the series key, so long as the attribute does not specify an attachment group or also has a relationship with the dimension declared to be at the observation level. |
| Obs | *ObsType* | Obs is an un-grouped observation. This observation has a key which is a set of values for all dimensions declared in the data structure definition. In addition to the key, the value of the observation can be provided along with values for all data and metadata attributes which have an association with the observation or any dimension (so long as it does not specify a group attachment). |
| Metadata | MetadataSetType | Allows for attachment of reference metadata against to the data set. |

***GroupType*:** GroupType is the abstract type which defines a structure which is used to communicate attribute values for a group defined in a data structure definition. The group can consist of either a subset of the dimensions defined by the data structure definition, or an association to an attachment constraint, which in turn defines key sets to which attributes can be attached. In the case that the group is based on an attachment constraint, only the identification of group is provided. It is expected that a system which is processing this will relate that identifier to the key sets defined in the constraint and apply the values provided for the attributes appropriately. Data structure definition schemas will drive types based on this for each group defined in the data structure definition. Both the dimension values which make up the key (if applicable) and the attribute values associated with the group will be represented with XML attributes. This is specified in the content model with the declaration of anyAttributes in the "local" namespace. The derived group type will refine this structure so that the attributes are explicit. The XML attributes will be given a name based on the attribute's identifier. These XML attributes will be unqualified (meaning they do not have a namespace associated with them). The dimension XML attributes will be required while the attribute XML attributes will be optional. To allow for generic processing, it is required that the only unqualified XML attributes in the derived group type be for the group dimensions and data or metadata attributes declared in the data structure definition. If additional attributes are required, these should be qualified with a namespace so that a generic application can easily distinguish them as not being meant to represent a data structure definition dimension or attribute.

Derivation:

*AnnotableType* (extension)   
   *GroupType*

Attributes:

type?

Content:

Annotations?, Comp\*, Metadata?

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| type | IDType | The type attribute reference the identifier of the group as defined in the data structure definition. This is optional, but derived group types will provide a fixed value for this so that it always available in the post validation information set. This allows the group to be processed in a generic manner. |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Comp | *CompType* | Comp contains the details of group level attributes that have complex representation and cannot be expressed as XML attributes. |
| Metadata | MetadataSetType | Allows for attachment of reference metadata against to the group. |

***SeriesType*:** SeriesType is the abstract type which defines a structure which is used to group a collection of observations which have a key in common. The key for a series is every dimension defined in the data structure definition, save the dimension declared to be at the observation level for this data set. In addition to observations, values can be provided for data and metadata attributes which are associated with the dimensions which make up this series key (so long as the attributes do not specify a group attachment or also have an relationship with the observation dimension). It is possible for the series to contain only observations or only attribute values, or both. Data structure definition schemas will derive a type based on this that is specific to the data structure definition and the variation of the format being expressed in the schema. Both the dimension values which make up the key and the attribute values associated with the key dimensions will be represented with XML attributes. This is specified in the content model with the declaration of anyAttributes in the "local" namespace. The derived series type will refine this structure so that the attributes are explicit. The XML attributes will be given a name based on the attribute's identifier. These XML attributes will be unqualified (meaning they do not have a namespace associated with them). The dimension XML attributes will be required while the attribute XML attributes will be optional. To allow for generic processing, it is required that the only unqualified XML attributes in the derived group type be for the series dimensions and attributes declared in the data structure definition. If additional attributes are required, these should be qualified with a namespace so that a generic application can easily distinguish them as not being meant to represent a data structure definition dimension or attribute.

Derivation:

*AnnotableType* (extension)   
   *SeriesType*

Attributes:

TIME\_PERIOD?

Content:

Annotations?, Comp\*, Obs\*, Metadata?

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| TIME\_PERIOD | ObservationalTimePer iodType | The TIME\_PERIOD attribute is an explict attribute for the time dimension. This is declared in the base schema since it has a fixed identifier and representation. The derived series type will either require or prohibit this attribute, depending on whether time is the observation dimension. If the time dimension specifies a more specific representation of time the derived type will restrict the type definition to the appropriate type. |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Comp | *CompType* | Comp contains the details of series level attributes that have complex representation and cannot be expressed as XML attributes. |
| Obs | *ObsType* |  |
| Metadata | MetadataSetType | Allows for attachment of reference metadata against to the series. |

***CompType*:** CompType is the abstract base for any component value (e.g. a data or metadata attribute, or a measure) that cannot be represented as an XML attribute. For example, a repeated value, a text value in multiple languages, or a value with structured text (XHTML) cannot be expressed as an XML attribute. This type is meant to be restricted based on the component to restrict the cardinality and type of its Value element to conform to the compoent definition. The type of the value element should be restricted to common:SimpleValueType, common:TextValueType, or common:StructuredValueType. In addition, the id attribute should be restricted to be a fixed value with the component identifier. This restricted type based on the component can then be used on Comp elements by using the xsi:type to state the component being expressed and refine the contents of the element to the values allowed by the component.

Derivation:

*AnnotableType* (extension)   
   *CompType*

Attributes:

id

Content:

Annotations?, Value\*

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| id | NCNameIDType |  |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Value | *ValueType* |  |

***ObsType*:** ObsType is the abstract type which defines the structure of a grouped or un-grouped observation. The observation must be provided a key, which is either a value for the dimension which is declared to be at the observation level if the observation is grouped, or a full set of values for all dimensions in the data structure definition if the observation is un-grouped. This key should disambiguate the observation within the context in which it is defined (e.g. there should not be another observation with the same dimension value in a series). The observation can contain an observed value and/or attribute values. Data structure definition schemas will derive a type or types based on this that is specific to the data structure definition and the variation of the format being expressed in the schema. The dimension value(s) which make up the key and the data and metadata attribute values associated with the key dimension(s) or the primary measure will be represented with XML attributes. This is specified in the content model with the declaration of anyAttributes in the "local" namespace. The derived observation type will refine this structure so that the attributes are explicit. The XML attributes will be given a name based on the attribute's identifier. These XML attributes will be unqualified (meaning they do not have a namespace associated with them). The dimension XML attribute(s) will be required while the attribute XML attributes will be optional. To allow for generic processing, it is required that the only unqualified XML attributes in the derived observation type be for the observation dimension(s) and attributes declared in the data structure definition. If additional attributes are required, these should be qualified with a namespace so that a generic application can easily distinguish them as not being meant to represent a data structure definition dimension or attribute.

Derivation:

*AnnotableType* (extension)   
   *ObsType*

Attributes:

type?, TIME\_PERIOD?

Content:

Annotations?, Comp\*, Metadata?

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| type | IDType | The type attribute is used when the derived format requires that explicit measure be used. In this case, the derived type based on the measure will fix this value to be the identification of the measure concept. This will not be required, but since it is fixed it will be available in the post validation information set which will allow for generic processing of the data. If explicit measures are not used, then the derived type will prohibit the use of this attribute. |
| TIME\_PERIOD | ObservationalTimePer iodType | The TIME\_PERIOD attribute is an explicit attribute for the time dimension. This is declared in the base schema since it has a fixed identifier and representation. The derived series type will either require or prohibit this attribute, depending on whether time is the observation dimension. If the time dimension specifies a more specific representation of time the derived type will restrict the type definition to the appropriate type. |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Comp | *CompType* | Comp contains the details of observation measures or attributes that have complex representation and cannot be expressed as XML attributes. |
| Metadata | MetadataSetType | Allows for attachment of reference metadata against to the observation. |

***AttsType*:** AttsType is the abstract type which defines a structure which is used to group a collection of data or metadata attributes which have a key in common. The key for a attribute collection is a subset of the dimension defined in the data structure definition. This is also used for data set level attributes (i.e. those with an attribute relationship of none). In this case, the subset of dimensions is empty. Data structure definition schemas will derive a type based on this that is specific to the data structure definition. The dimension values which make up the key will be represented with local (non-namespace qualified) XML attributes. The metadata attribute values associated with the key dimensions will be expressed as XML local (non-namespace qualified) attributes if they are simple values (e.g. enumerated, dates, numbers) and are not repeatable. Metadata attributes that are repeatable, or do not have simple values (e.g. text) will be expressed using the Comp element. This dimensions and simple attributes are specified in the content model with the declaration of anyAttributes in the "local" namespace. The derived series type will refine this structure so that the attributes are explicit. The XML attributes will be given a name based on the attribute's identifier. These XML attributes will be unqualified (meaning they do not have a namespace associated with them). The dimension XML attributes will be required while the attribute XML attributes will be optional. To allow for generic processing, it is required that the only unqualified XML attributes in the derived group type be for the series dimensions and attributes declared in the data structure definition. If additional attributes are required, these should be qualified with a namespace so that a generic application can easily distinguish them as not being meant to represent a data structure definition dimension or attribute.

Derivation:

*AnnotableType* (extension)   
   *AttsType*

Attributes:

TIME\_PERIOD?

Content:

Annotations?, Comp\*

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| TIME\_PERIOD | ObservationalTimePer iodType | The TIME\_PERIOD attribute is an explict attribute for the time dimension. This is declared in the base schema since it has a fixed identifier and representation. The derived series type will either require or prohibit this attribute, depending on whether time is the observation dimension. If the time dimension specifies a more specific representation of time the derived type will restrict the type definition to the appropriate type. |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Comp | *CompType* | Comp contains the details of the data or metadata attributes that have complex representation and cannot be expressed as XML attributes. |

## Generic Metadata Namespace

**http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/metadata/generic**

### Summary

Referenced Namespaces:

| **Namespace** | **Prefix** |
| --- | --- |
| http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/common | common |
| http://www.w3.org/2001/XMLSchema | xs |

Contents:

1 Global Element  
3 Complex Types

### Global Elements

**Attribute (AttributeType):** Att elements hold the reported values for a given metadata attribute. These values conform to the definition of the meatadata attribute in the metadata structure definition.

### Complex Types

***MetadataSetBaseType*:** MetadataSetBaseType defines the base refinement of the MetadataSetType. Its purpose is to restrict the urn attribute.

Derivation:

*AnnotableType* (extension)   
   *IdentifiableType* (extension)   
         *NameableType* (extension)   
               *VersionableType* (restriction)   
                     *MaintainableBaseType* (extension)   
                           *MaintainableType* (restriction)   
                                 *MetadataSetBaseType*

Attributes:

id, urn?, uri?, version?, validFrom?, validTo?, agencyID, isExternalReference?, serviceURL?, structureURL?

Content:

Annotations?, Link\*, Name+, Description\*

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| id | IDType | The id is the identifier for the object. |
| urn | MetadataSetUrnType | The urn attribute holds a valid SDMX Registry URN (see SDMX Registry Specification for details). |
| uri | xs:anyURI | The uri attribute holds a URI that contains a link to a resource with additional information about the object, such as a web page. This uri is not a SDMX message. |
| version | VersionType | This version attribute holds a version number (see common:VersionType definition for details). If not supplied, artefact is considered to be un-versioned. |
| validFrom | xs:dateTime | The validFrom attribute provides the inclusive start date for providing supplemental validity information about the version. |
| validTo | xs:dateTime | The validTo attribute provides the inclusive end date for providing supplemental validity information about the version. |
| agencyID | NestedNCNameIDType | The agencyID must be provided, and identifies the maintenance agency of the object. |
| isExternalReference (default: false) | xs:boolean | The isExternalReference attribute, if true, indicates that the actual object is not defined the corresponding element, rather its full details are defined elsewhere - indicated by either the registryURL, the repositoryURL, or the structureURL. The purpose of this is so that each structure message does not have to redefine object that are already defined elsewhere. If the isExternalReference attribute is not set, then it is assumed to be false, and the object should contain the full definition of its contents. If more than one of the registryURL, the repositoryURL, and the structureURL are supplied, then the application processing the object can choose the method it finds best suited to retrieve the details of the object. |
| serviceURL | xs:anyURI | The serviceURL attribute indicates the URL of an SDMX SOAP web service from which the details of the object can be retrieved. Note that this can be a registry or and SDMX structural metadata repository, as they both implement that same web service interface. |
| structureURL | xs:anyURI | The structureURL attribute indicates the URL of a SDMX-ML structure message (in the same version as the source document) in which the externally referenced object is contained. Note that this may be a URL of an SDMX RESTful web service which will return the referenced object. |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Link | LinkType | Allows for the linking of other resources to identifiable objects. For example, if there is reference metadata associated with a structure, a link to the meatadata report can be dynamically inserted in the structure metadata. |
| Name | TextType | Name provides for a human-readable name for the object. This may be provided in multiple, parallel language-equivalent forms. |
| Description | TextType | Description provides for a longer human-readable description of the object. This may be provided in multiple, parallel language-equivalent forms. |

**MetadataSetType:** MetadataSetType describes the structure for a metadata set, which contains a collection of reported metadata against a set of targets. The targets should conform to the restrictions described by the metadata provision or the metadataflow. Note that this is maintainble, and as such must specify in agency. In this case, the agency is the metadata provider. If a metadata provision agreement is referenced, it is assumed that the metadata provider described in the provision will be the same as the agency for this set.

Derivation:

*AnnotableType* (extension)   
   *IdentifiableType* (extension)   
         *NameableType* (extension)   
               *VersionableType* (restriction)   
                     *MaintainableBaseType* (extension)   
                           *MaintainableType* (restriction)   
                                 *MetadataSetBaseType* (extension)   
                                       MetadataSetType

Attributes:

id, urn?, uri?, version?, validFrom?, validTo?, agencyID, isExternalReference?, serviceURL?, structureURL?, action?, reportingBeginDate?, reportingEndDate?, publicationYear?, publicationPeriod?

Content:

Annotations?, Link\*, Name+, Description\*, ( (MetadataProvisionAgreement | Metadataflow), Target+, Attribute+)?

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| id | IDType | The id is the identifier for the object. |
| urn | MetadataSetUrnType | The urn attribute holds a valid SDMX Registry URN (see SDMX Registry Specification for details). |
| uri | xs:anyURI | The uri attribute holds a URI that contains a link to a resource with additional information about the object, such as a web page. This uri is not a SDMX message. |
| version | VersionType | This version attribute holds a version number (see common:VersionType definition for details). If not supplied, artefact is considered to be un-versioned. |
| validFrom | xs:dateTime | The validFrom attribute provides the inclusive start date for providing supplemental validity information about the version. |
| validTo | xs:dateTime | The validTo attribute provides the inclusive end date for providing supplemental validity information about the version. |
| agencyID | NestedNCNameIDType | The agencyID must be provided, and identifies the maintenance agency of the object. |
| isExternalReference (default: false) | xs:boolean | The isExternalReference attribute, if true, indicates that the actual object is not defined the corresponding element, rather its full details are defined elsewhere - indicated by either the registryURL, the repositoryURL, or the structureURL. The purpose of this is so that each structure message does not have to redefine object that are already defined elsewhere. If the isExternalReference attribute is not set, then it is assumed to be false, and the object should contain the full definition of its contents. If more than one of the registryURL, the repositoryURL, and the structureURL are supplied, then the application processing the object can choose the method it finds best suited to retrieve the details of the object. |
| serviceURL | xs:anyURI | The serviceURL attribute indicates the URL of an SDMX SOAP web service from which the details of the object can be retrieved. Note that this can be a registry or and SDMX structural metadata repository, as they both implement that same web service interface. |
| structureURL | xs:anyURI | The structureURL attribute indicates the URL of a SDMX-ML structure message (in the same version as the source document) in which the externally referenced object is contained. Note that this may be a URL of an SDMX RESTful web service which will return the referenced object. |
| action | ActionType | The action attribute indicates whether the file is appending, replacing, or deleting. |
| reportingBeginDate | BasicTimePeriodType | The reportingBeginDate indicates the inclusive start time of the data reported in the data or metadata set. |
| reportingEndDate | BasicTimePeriodType | The reportingEndDate indicates the inclusive end time of the data reported in the data or metadata set. |
| publicationYear | xs:gYear | The publicationYear holds the ISO 8601 four-digit year. |
| publicationPeriod | ObservationalTimePer iodType | The publicationPeriod specifies the period of publication of the data or metadata in terms of whatever provisioning agreements might be in force (i.e., "Q1 2005" if that is the time of publication for a data set published on a quarterly basis). |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Link | LinkType | Allows for the linking of other resources to identifiable objects. For example, if there is reference metadata associated with a structure, a link to the meatadata report can be dynamically inserted in the structure metadata. |
| Name | TextType | Name provides for a human-readable name for the object. This may be provided in multiple, parallel language-equivalent forms. |
| Description | TextType | Description provides for a longer human-readable description of the object. This may be provided in multiple, parallel language-equivalent forms. |
| MetadataProvisionAgr eement | MetadataProvisionAgr eementReferenceType | Metadataflow provides a reference to the. |
| Metadataflow | MetadataflowReferenc eType | Metadataflow provides a reference to the metadataflow the metadata set is being reported against. |
| Target | WildcardUrnType | Target references the target structures for which metadata is being reported. These must conform with the constraints defined by the metadata provision agreement and/or the metadataflow. |
| Attribute | AttributeType | Att elements hold the reported metadata attribute values being reported in the metadata set. These conform to the metadata structure defintion |

**AttributeType:** AttributeType defines the structure for a reported metadata attribute. A value for the attribute can be supplied as either a single value (enumerated or non-enumerated single value), or multi-lingual text values (either structured or unstructured). Optional child attributes ar also available if the metadata attribute definition defines nested metadata attributes.

Derivation:

*AnnotableType* (extension)   
   AttributeType

Attributes:

id

Content:

Annotations?, (Value+ | Text+ | StructuredText+)?, Attribute\*

Attribute Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| id | IDType | The id attribute identifies the metadata attribute that the value is being reported for. |

Element Documentation:

| **Name** | **Type** | **Documentation** |
| --- | --- | --- |
| Annotations | AnnotationsType | Annotations is a reusable element the provides for a collection of annotations. It has been made global so that restrictions of types that extend AnnotatableType may reference it. |
| Value | xs:anySimpleType | Value holds any simple value (enumerated or not) for the metadata attribute. It can be repeated if this metadata attribute allows for multiple values. |
| Text | TextType | Text is used to supply parallel multi-lingual textual values for the reported metadata attribute. This will be used if the text format of the metadata attribute has a type of string and the multi-lingual value is set to true. |
| StructuredText | XHTMLType | StructuredText is used to supply parallel multi-lingual structured (as XHTML) textual values for the reported metadata attribute. This will be used if the text format of the metadata attribute has a type of XHTML and the multi-lingual value is set to true. |
| Attribute | AttributeType | Att contains the reported metadata attribute values for the child metadata attributes. |

# Mapping to Structure-Specific Schemas

## General

Data structure-specific schemas are each based on one single core construct found in the structure-specific namespace;

Data - http://www.SDMX.org/resources/SDMXML/schemas/v3\_0/data/structurespecific

### Basic Terminology

In the subsequent sections, the following namespace prefixes are used:

|  |  |
| --- | --- |
| **Namespace** | **Prefix** |
| http://www.w3.org/2001/XMLSchema | xs |
| http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/common | common |
| http://www.sdmx.org/resources/sdmxml/schemas/ v3\_0/data/structurespecific | dsd |
| http://www.sdmx.org/resources/sdmxml/schemas/v3\_0/metadata/generic | metadata |

It is assumed that in order to use this guide, the reader is familiar with schema terminology. However, for convenience the following is list of the terminology used here:

**Schema:** Refers to the format specific schema in general, and in particular the root xs:schema element of that schema file.

**Global Element:** Refers to an element definition at the top level of the schema (i.e. an xs:element element in the root xs:schema element). It will define a name and type (name and type attributes) and possibly a substitution group (substitutionGroup attribute).

**Local Element:** Refers to an element definition within a complex type (i.e. an xs:element element contained within a xs:sequence element that is contained in a xs:complexType element). A local element must define a name and type (name and type attributes) and may also specify a minimum and maximum occurrence (minOccurs and maxOccurs attribute).

**Qualified/Unqualified Element:** A qualified element is an element that must be referred to by the namespace in which it was defined. An unqualified element does not have a namespace associated with it. The structure specific schemas make use of unqualified elements to that the structure specific schemas can restrict the base content to meet the specific needs of the structure, while maintaining as much of the original document structure as possible.

**Element Reference:** Refers to an element definition within a complex type that is a reference to a global element (i.e. an xs:element element contained within a xs:sequence element that is contained in a xs:complexType element). An element reference must reference a global element (via its ref attribute) and may also specify a minimum and maximum occurrence (minOccurs and maxOccurs attribute).

**Complex Type:** Refers to a complex type definition. In this context, all complex type definitions occur at the top level of the schema (i.e. an xs:complexType element in the root xs:schema element). A complex type must define a name (name attribute) and may be made abstract (via the abstract attribute’s boolean value).

**Simple Type:** Refers to a simple type definition. In this context, all simple type definitions occur at the top level of the schema (i.e. an xs:simpleType element in the root xs:schema element). In this context, a simple type will always be defined via a restriction (an xs:restriction element in the xs:simpleType element). The restriction will reference a base type.

**Anonymous Type:** A complex or simple type definition which occurs within an element definition. The method is sometimes referred to a the "Russian-doll" technique as it creates nested constructs. Anonymous types are not given names and cannot be abstract. The can however, be derived from other types.

**Content Group:** A group which defines a content model for reuse. This is contained in the xs:group element, and is defined at the root of the schema. It allows for a common sequence or choice of elements to be reused across multiple types without having to redefine the sequence or choice in each type.

**Uniqueness Constraint:** A uniqueness constraint is defined within an element and is used to force descendent elements to be unique based on some criteria of it fields (elements or attributes). This is defined in an <xs:unique> element, and has content of an <xs:selector> and multiple <xs:field> elements. The selector designates the descendants that must be unique (with an xpath attribute) and the field specifies which property of the selected element must be unique (also with an xpath attribute)

**Extension:** Refers to the definition of a complex type that is an extension of another complex type. The extension will always make a reference to a base. In the schema, this is defined within the xs:complexType element as a child xs:complexContent element containing an xs:extension element (with a base attribute).

**Restriction:** Refers to the definition of a simple or complex type that is a restriction of another type of the same variety. The restriction will always make a reference to a base. In the schema, this is defined with an xs:restriction element (with a base attribute).

**Sequence:** Refers to a sequence of elements that may be defined as the root of a complex type content model, or as part of the content of a choice or another sequence. This is defined as an xs:sequence element. The sequence may specify a minimum and maximum occurrence (minOccurs and maxOccurs attribute).

**Choice:** Refers to a choice of elements that may be defined as the root of a complex type content model, or as part of the content of a sequence or another choice. This is defined as an xs:choice element. The sequence may specify a minimum and maximum occurrence (minOccurs and maxOccurs attribute).

**Facet:** Refers to a single detail of a simple type restriction. This is represented by elements such as xs:minInclusive, xs:totalDigits, xs:minLength, and is contained in the xs:restriction element of a simple type definition. The value of the facet is contained in a value attribute of the particular element.

**Enumeration:** Refers to an enumerated value of a simple type definition. It is represented by an xs:enumeration element contained within an xs:restriction element of a simple type definition. An enumeration defines a value (in the value attribute) and documentation (in xs:documentation elements contained in an xs:annotation element).

**XML Attribute:** Refers to the definition of an XML attribute for a complex type (i.e. and xs:attribute element in a xs:complexType element). An attribute must define a name and type (name and type attributes) and may also specify a usage (use attribute).

## Namespace Rules

Each format specific schema will specify its namespace in the target namespace of the schema (the targetNamespace attribute of the schema). This document also assumes that the root namespace (that which is defined by the xmlns attribute) of the schema will be the same as the target namespace. Therefore any types or global elements referenced in these descriptions without a namespace prefix are assumed to be in the format specific namespace.

The format specific schemas will incorporate the core format namespace and the common namespace by importing the schemas (via the xs:import element). If necessary, additional namespaces may be imported and referenced.

For the purpose of the descriptions here, the default element form for the schema (as specified in the elementFormDefault attribute of the schema) is “qualified", and the default attribute form (as specified in the attributeFormDefault attribute of the schema) is "unqualified".

## General Rules

The following section details the general rules which apply to all structure specific schema creation.

### Component Name Determination

When required to create an XML element or attribute, the name for a component is always its identifier. However, the identifier may be inherited. Therefore, the general rules is as follows:

1. If the component defines an identifier, the element or attribute name is the value of that identifier
2. Otherwise, the element or attribute name is the identifier of the concept from which it takes its semantic (Note that this is technically the component identifer).

### Representation Determination

Every component has a representation associated with it, whether it is defined as a local representation in the component definition, or it is inherited from the concept from which the component takes it semantic (as defined in the concept identity of the component).

The representation of a component is determined by the following precedence:

1. The local representation defined by the component
2. The core representation defined in the concept from which the component takes its semantic
3. A default representation of an un-faceted text format with a data type of String.

The representation will either define a text format, or an enumeration with an enumeration format.

A text format consists of a data type and an optional collection of facets. It is the combination of these which determine the exact nature of the component representation. An enumeration consists of a reference to a codelist, hierarchy, or value list, for which an enumerated list of possible values can be created.

### Simple / Primitive Type Determination

For any given representation, there exist rules for determining the simple or primitive type which should be used to validate the value. There are no specific requirements to how a simple type is named or if it is referenced or used as an anonymous type. This section simply serves to state the requirements of the type for a component based on its [determined representation](#_Representation_Determination:).

For example, a dimension may inherit its representation for a concept, and the data type of a representation data type may be a String. The simplest solution would be to use the xs:string primitive type. However, an implementer may have chosen to generate simple types for all concepts to avoid having to look up the concept core representation for very component. In this case, the type may be given a name based on the concept and be a simple derivation from the xs:string type that places no further restrictions. The result would be that the type that is actually used for the dimension, although named after the concept, is effectively the required xs:string. These rules are meant to allow such flexibility in how types are created. The only requirement is that the type meet the requirements stated here.

### Representation with Codelist Enumeration

A representation which defines an enumeration from a codelist or hierarchy will result in a simple type that is a restriction of the common:IDType. The simple type will define enumerations for each code in the codelist or hierarchy, accounting for extensions. The value for these enumerations will be identifier of the item. If desired, the names of the item may be placed in the documentation of the enumeration, but this is not required. Example:

<xs:simpleType name="ESTAT.CL\_COUNTRY.1.0">

<xs:restriction base="common:IDType">

<xs:enumeration value="BE">

<xs:annotation>

<xs:documentation xml:lang="en">Belgium</xs:documentation>

</xs:annotation>

</xs:enumeration>

### Representation with Value List Enumeration

A representation which defines an enumeration from a value list will result in a simple type that is a restriction of the xs:string data type. The simple type will define enumerations for each value item in the value list. The value for these enumerations will be identifier of the item. If desired, the names of the item may be placed in the documentation of the enumeration, but this is not required.

### Representation with Simple Text Format

A representation which defines a simple text format will result in a simple type or primitive type. The representation is simple if none of the following conditions are true:

* representation max occurs is greater than 1
* text format data type is XHMTL
* text format is multi-lingual

IF the representation is not simple, see the rules in the following section for complext text formats. If the representation is simple, the first step is to determine the base type from the text format data type:

|  |  |
| --- | --- |
| **SDMX Data Type** | **XML Schema Data Type** |
| String | xs:string |
| AlphaNumeric | common:AlphaNumericType |
| Alpha | common:AlphaType |
| Numeric | common:NumericType |
| BigInteger | xs:integer |
| Integer | xs:int |
| Long | xs:long |
| Short | xs:short |
| Decimal | xs:decimal |
| Float | xs:float |
| Double | xs:double |
| Boolean | xs:boolean |
| URI | xs:anyURI |
| Count | xs:integer |
| InclusiveValueRange | xs:decimal |
| ExclusiveValueRange | xs: decimal |
| Incremental | xs: decimal |
| ObservationalTimePeriod | common:ObservationalTimePeriodType |
| StandardTimePeriod | common:StandardTimePeriodType |
| BasicTimePeriod | common:BasicTimePeriodType |
| GregorianTimePeriod | common:GregorianTimePeriodType |
| GregorianYear | xs:gYear |
| GregorianYearMonth | xs:gYearMonth |
| GregorianDay | xs:date |
| ReportingTimePeriod | common:ReportingTimePeriodType |
| ReportingYear | common:ReportingYearType |
| ReportingSemester | common:ReportingSemesterType |
| ReportingTrimester | common:ReportingTrimesterType |
| ReportingQuarter | common:ReportingQuarterType |
| ReportingMonth | common:ReportingMonthType |
| ReportingWeek | common:ReportingWeekType |
| ReportingDay | common:ReportingDayType |
| DateTime | xs:dateTime |
| TimeRange | common:TimeRangeType |
| Month | xs:gMonth |
| MonthDay | xs:gMonthDay |
| Day | xs:gDay |
| Time | xs:time |
| Duration | xs:duration |
| GeospatialInformation | xs:string |
| XHTML | See the following section for complex representations |

If the text format does not specify any further facets, then the determined type is the listed type or a type which derives from the listed type without placing any addition restrictions on it. However, if one or more facets are specified, then a simple type based on the listed type is necessary. The simple type derives via restriction from the listed type and adds facets according to the following table (the values are mapped as is):

|  |  |
| --- | --- |
| **SDMX Facet** | **XML Schema Facet** |
| minLength | xs:minLength |
| maxLength | xs:maxLength |
| minValue[[1]](#footnote-1) | if ExclusiveValueRange: xs:minExclusives, else: xs:minInclusive |
| maxValue2 | if ExclusiveValueRange: xs:maxExclusives, else: xs:maxInclusive |
| decimals2 | xs:fractionDigits |
| pattern | xs:pattern |

Any other facets are informational only, and will not affect the determined type.

### Representation with Complex Text Format

A representation which defines a complex text format will result in a complex type. The representation is complex if any of the following conditions are true:

* representation max occurs is greater than 1
* text format data type is XHMTL
* text format is multi-lingual

The resulting complex type will be derived via restriction of the common:ValueType. In simple cases, there are pre-defined types that can be used:

* if the text format data type is XHTML, common:StructuredTextValueType can be used; note that this type cannot be further restricted – all other facets are ignored
* if the text format is multi-lingual, common:TextValueType can be used; note that this type cannot be further restricted – all other facets are ignored
* if the text format has no additional facets and the data type is:
  + Boolean, common:BooleanValueType can be used
  + String, common:StringValueType can be used
  + Integer, common:IntValueType can be used
  + Double, common:DoubleValueType can be used
  + ObservationalTimePeriod, common:ObservationalTimePeriodValueType can be used

If a pre-defined type cannot be used, one will have to be created. The complex type must define a simple content restriction of the common:ValueType. The restriction should define an anonymous simple type based on the text format data type and facets as described in the previous section.

### Type Names

These rules will only dictate type names where absolutely necessary. In all other cases, it is the decision of the implementer as to how to name or use the type. It is also the implementer's requirement to ensure that any type name is properly unique within its scope. To assist in this, the following recommendations are offered for naming types such that they are unique.

* It the type is an enumeration from an item scheme, the recommended name is [Item Scheme Class].[Maintenance Agency].[Item Scheme ID].[Item Scheme Version]
* If the type is based on a text format of a concept core representation, the recommended name is Concept.[Maintenance Agency].[Concept Scheme ID].[Concept Scheme Version].[Concept ID]
* If the type is based on a text format of a component local representation, and;
  + The component id is required to be unique for all components within the scope of the structure which defines it (e.g. a dimension), the recommended name is [Component ID]
  + The component id is only required to be unique within the component list and which defines it (e.g. a metadata attribute), the recommend name is [Component List ID].[Parent Component ID]\*.[Component ID]

### Type Reuse

It is possible that organisations that manage a large number of structure specific schemas my wish to take advantage of the reuse of previously defined type in order to simply the structure specific schema creation and lessen the number of schema elements which are created. The structure specific formats are designed in such a way that this would be allowed without any adverse affects.

For example, an organisation my create predefined types for all of codelists and concept schemes which their structures utilize. These could be contained in a common schema with any namespace deemed appropriate. This would allow the structure specific schemas generation process to recognize the reused components and not be concerned with regenerating types. The logical flow for setting the representation of a component might be as follows:

Does the component define a local type?

Yes: Is that type enumerated?

Yes: Type is the qualified type name for the item scheme

No: Generate simple type for text format

No: Type is the qualified name for the concept from which the component takes its semantic.

Only the constructs that will be detailed in the data and metadata structure-specific rules below are required to be in the specified target namespace of the structure-specific schema. So long as any other generated type conforms to the rules specified, it may exist in any namespace.

## Data Structure Specific Schema

Separate schemas will be created for the data structure depending on which dimension occurs at the observation level. The recommended target namespace of the data structured specific schema is: [Data Structure URN]:ObsLevelDim:[Observation Dimensions].

The rules for generating the data structure specific-schema are broken into sections based on the level within the structure (i.e. data set, group, series, attributes, observation). Each section will state the rules for each variation of the structure specific format.

### DataSetType

A complex type named DataSetType must be created. Its content model will be derived via restriction. The base type of the restriction is dsd:DataSetType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. A local element named DataProvider with the type common:DataProviderReferenceType, a form of unqualified and a minimum occurrence of 0
   3. A choice with a minimum occurrence of 0 and a maximum occurrence of unbounded consisting of:
      1. A local element named Atts with a form of unqualified and a type of AttsType (as defined in the AttsType section which follows)
      2. If the data structure defines groups, a local element named Group with a form of unqualified. The type of this element should be the type that is described in the GroupType section which follows.
      3. If the dimension at the observation level is not AllDimensions, a local element named Series with a form of unqualified and a type of SeriesType (as defined in the SeriesType section which follows)
      4. If the dimension at the observation level is AllDimensions, a local element named Obs with a form of unqualified and a type of ObsType (as defined in the ObsType section which follows)
   4. If any metadata attribute usages defined in the data structure that declares an attribute relationship of dataflow, a local element named Metadata with the type metadata:MetadataSetType a form of unqualified, and a minimum occurences of 0

### GroupType

If the data structure definition defines only one group, a complex type with its name taken from the identifier of the lone group must be defined. This type is used for the Group element in the DataSetType. Its content model will be derived via restriction of the dsd:GroupType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. If any attributes defined in the data structure that declares an attribute relationship with the group, a Comp element with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of dsd:CompType
   3. If any metadata attribute usages defined in the data structure that declares an attribute relationship with the group, a local element named Metadata with the type metadata:MetadataSetType a form of unqualified, and a minimum occurences of 0
2. An attribute for each dimension referenced by the group. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is required
3. An attribute for each data attribute with simple representation defined in the data structure that declares an attribute relationship with the group or specifies the group as an attachment group. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is optional
4. An attribute named type with a type of common:IDType, usage of optional, and a fixed value of the identifier of the group

If the data structure definition defines more than one group, an abstract complex type with name GroupType must be created. This type is used for the Group element in the DataSetType. Its content model will be derived via restriction of the dsd:GroupType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. If any attributes defined in the data structure that declares an attribute relationship with a group, a Comp element with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of dsd:CompType
   3. If any metadata attribute usages defined in the data structure that declares an attribute relationship with any group, a local element named Metadata with the type metadata:MetadataSetType a form of unqualified, and a minimum occurences of 0
2. An attribute named type with a type of Group.ID, and a usage of optional
3. An anyAttribute declaration with a namespace of ##local

A simple type named Group.ID must be created. This should restrict the common:IDType. For each group defined by the data structure definition, an enumeration will be created within the restriction with a value of the group identifier.

For each group defined in the data structure definition, a complex type with its name taken from the group identifier is defined. Its content model will be derived via restriction of the previously defined GroupType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. If any attributes with complex representation defined in the data structure declares an attribute relationship with the group, a Comp element with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of dsd:CompType
   3. If any metadata attribute usages defined in the data structure that declares an attribute relationship with the group, a local element named Metadata with the type metadata:MetadataSetType a form of unqualified, and a minimum occurences of 0
2. An attribute for each dimension referenced by the group. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is required
3. An attribute for each data attribute with simple representation defined in the data structure that declares an attribute relationship with the group or specifies the group as an attachment group. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is optional
4. An attribute named type with a type of Group.ID, usage of optional, and a fixed value of the identifier of the group

### SeriesType

If the dimension at the observation is not AllDimensions, a complex type name SeriesType must be created. Its content model will be derived via restriction of dsd:SeriesType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. If any attributes with complex representation defined in the data structure declares an attribute relationship with a dimension that is not at the observation level, a Comp element with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of dsd:CompType
   3. A local element named Obs with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of ObsType (as defined in the ObsType section which follows)
   4. If any metadata attribute usages defined in the data structure that declares an attribute relationship with the series, a local element named Metadata with the type metadata:MetadataSetType a form of unqualified, and a minimum occurences of 0
2. An attribute named TIME\_PERIOD with a type of common:ObservationalTimePeriod. If the dimension at the observation level is the time dimension (TIME\_PERIOD) or there is no time diemsnion defined by the data structure, a usage of prohibited; otherwise a usage of required
3. An attribute for each dimension defined by the data structure definition, except for the dimension at the observation level and the time dimension (TIME\_PERIOD). The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is required
4. An attribute for each data attribute defined with simple representation in the data structure that declares an attribute relationship with any dimension outside of the dimension at the observation level (so long as it does not also declare an attachment group). The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is optional

### AttsType

A a complex type named AttsType must be created. Its content model will be derived via restriction of dsd:AttsType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. If any attributes with complex representation are defined in the data structure, a Comp element with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of dsd:CompType
2. If there is no dimension (TIME\_PERIOD) defined by the data structure, an attribute named TIME\_PERIOD with a type of common:ObservationalTimePeriod, and a usage of prohibited
3. An attribute for all dimension defined by the data structure definition. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is optional
4. An attribute for each data attribute defined with simple representation in the data structure. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is optional

### ObsType

A complex type name ObsType must be created. Its content model will be derived via restriction of the base type dsd:ObsType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. If any measures with complex representations are defined in the data structure, any attributes with complex representation that declare an attribute relationship with the observation, a Comp element with a form of unqualified, a minimum occurrence of 0, a maximum occurrence of unbounded, and a type of dsd:CompType
   3. If any metadata attribute usages defined in the data structure that declares an attribute relationship with the observation, a local element named Metadata with the type metadata:MetadataSetType a form of unqualified, and a minimum occurences of 0
2. An attribute named TIME\_PERIOD with a type of common:TimePeriodType. If the dimension at the observation level is the time dimension (TIME\_PERIOD) or all dimensions and the time dimension is defined by the data structure, a usage of required; otherwise a usage of prohibited
3. If the dimension at the observation level is not all dimensions or the time dimension (TIME\_PERIOD), an attribute for the dimension at the observation level. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is required
4. If the dimension at the observation level is all dimensions, an attribute for each dimension defined by the data structure definition, except for the time dimension (TIME\_PERIOD). The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is required
5. An attribute for each measure with simple representation defined by the data structure definition. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) is defined according to the general rules defined in the previous section, and the usage is optional
6. An attribute for each data attribute with simple representation defined in the data structure that declares an attribute relationship with the obseration. The XML attribute [name](#_Component_Name_Determination) and [type](#_Simple_/_Primitive) are defined according to the general rules defined in the previous section, and the usage is optional

### CompType

For every measure and data attribute with complex representation defined by the data structure definition, a complex type must be derived from the restriction of the dsd:CompType. The complex type content model will be as follows:

1. A sequence consisting of:
   1. An element reference to common:Annotations, with a minimum occurrence of 0
   2. A Value element with a form of unqualified, with a minimum occurrence of 0, a maximum occurrence defined by the representation, and a type based on the [complex representation type](#_Representation_with_Complex) defined according to the general rules defined in the previous section
2. An attribute named id with a type of common: NCNameIDType, usage of required, and a fixed value of the identifier of the measure or attribute

# Special data functions

## Updates

D messages allow for incremental updating of data. This purpose is noted in the action for the data set, which is either inherited from the header of the data message or explicitly stated at the data set level.

A dataset with an action of Append is assumed to be an incremental update. This means that one the information provided explicitly in the message should be altered. Any data attribute or observation value that is to be changed must be provided. However, the absence of an observation value or a data attribute at any level does not imply deletion; instead it is simply implied that the value is to remain unchanged. Therefore, it is valid and acceptable to send a data message with an action of Append which contains only a Series elements with attribute values. In this case, the values for the attributes will be updated. Note that it is not permissible to update data attributes using partial keys (outside of those associated with defined groups). In order to update an attribute, a full key must always be provided even if the message format does not require this.

## Deletes

Data messages allow for incremental deletion of data. This purpose is noted in the action for the data set, which is either inherited from the header of the data message or explicitly stated at the data set level.

A dataset with an action of Delete is assumed to be an incremental deletion. The deletion is assumed to take place of the lowest level of detail provided in the message. For example, if a delete message is sent with only a data set element, the entire data set will be deleted. On the other hand, if that data set contains a data attribute, only that data attribute value will be deleted. This same dynamic continues through the data set hierarchy. A data set containing only a series with no data attributes or observations will result in that entire series (all observations and data attributes) being deleted. If the series contains data attributes, only the supplied data attributes for that series will be deleted. Finally, if a series contains observations, then only the specified observations will be deleted. If an entire observation is to be deleted (value and data attributes), only the observation dimension should be provided. If only the observation value or particular data attributes are to be deleted, then these should be specified for the observation. Note that a group can only be used to delete the data attributes associated with it. Although the format might not require it, a full key must be provided to delete a series or observation. It is not permissible to wild card a key in order to delete more than one series or observation. Finally, to delete a data attribute or observation value it is recommended that the value to be deleted be supplied; however, it is only required that any valid value be provided.

1. Note that these options only apply to numeric representations and should be ignored if the data type is non-numeric [↑](#footnote-ref-1)