Clinical Data Interchange Standards Consortium

Specification for the Operational Data Model (ODM)

Version 1.3.2 Production
Source File: ODM1-3-2.htm
Last Update: 2013-12-01

An official copy of this document is available on the ODM page of the CDISC website http://www.cdisc.org/odm.

Please see Section 7.0 for Review Period; Licensing Obligations, Representations and Warranties; Limitations of Liability, and Disclaimers.

This document is the specification for ODM version 1.3.2. A list of additions and changes since ODM version 1.3.0 is provided in Section 2.5 Changes from Previous Versions. All changes are backward compatible -- ODM XML files conforming to the ODM version 1.2, 1.2.1, 1.3.0 or 1.3.1 schema will also conform to the ODM version 1.3.2 schema.

Table of Contents

1 Introduction (non-normative)

The Operational Data Model (ODM) is a vendor neutral, platform independent format for interchange and archive of clinical study data. The model includes the clinical data along with its associated metadata, administrative data, reference data and audit information. All of the information that needs to be shared among different software systems during the setup, operation, analysis, submission or for long term retention as part of an archive is included in the model.

This is version 1.3.2 of the ODM. Section 2.5 Changes from Previous Versions provides a summary of the new features. Detailed descriptions of the new features have been incorporated into the text of this document. ODM Version 1.3.2 has been designed to provide backward compatibility with previous versions of the model in most cases. In cases where backward compatibility has been impacted, it is due to clarifying the specification to more completely match the original intent of the standard.

Earlier versions of the model included a DTD. For version 1.2, both an XML Schema and a DTD were provided. Beginning with version 1.3.0 and going forward, only an XML Schema is available.

Clinical data management systems vary significantly in the information they store and the rules they enforce. The ODM model has been designed to represent a wide range of study information so as to be compatible with most existing clinical data management systems. Systems that do not have all of the features represented by the ODM model may still be ODM compatible as long as they comply with the conformity rules provided in the section on System Conformity.

The ODM has been designed to be compliant with guidance and regulations published by the FDA for computer systems used in clinical studies. This document is intended to be both the formal specification of the ODM and a user guide for those involved in transferring or archiving of clinical data using the model.

2 General Issues

2.1 Content of the Standard

This document contains the specification for the CDISC ODM standard. Most of the text in this standard expresses requirements on systems that read or write ODM files. Some, however, is commentary, usage notes or examples. All such "unofficial" text is segregated into paragraphs or sections prominently marked as **non-normative**, a **note**, or an **example**.

2.2 File Conformity

An XML file conforms to the ODM 1.3.2 standard only if it satisfies all the criteria detailed in this standard. These criteria include both syntactic constraints and semantic ones.

The syntactic constraints are

1. The ODM file must be a well-formed XML file. See the XML standard for details.
2. The ODM file must conform to the XML Namespace standard. See the XML Namespace standard for details.
3. The ODM file must contain only elements and attributes defined in the ODM standard schema or in a valid vendor extension schema, and must satisfy the rules about element nesting and the formats of attribute values and element bodies.
4. The ODM file must contain a prolog and a single (top-level) ODM element.
5. The ODM file must use the ODM 1.3 namespace, which is "http://www.cdisc.org/ns/odm/v1.3"
6. The ODMVersion attribute on the top level ODM element must be set to "1.3.2"

The schema for version 1.3.2 of the ODM can be found at http://www.cdisc.org/schemas/odm/v1.3. This final version of this specification will be posted at http://www.cdisc.org/odm.

Example of prolog and top level ODM element:

```
<?xml version="1.0" encoding="UTF-8"?>
<ODM xmlns="http://www.cdisc.org/ns/odm/v1.3"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://www.cdisc.org/ns/odm/v1.3 ODM1-3-2.xsd"

    ODMVersion="1.3.2"
    FileOID="000-00-0000"
    FileType="Transactional"
    Description="Sample ItemData Extension"

    AsOfDateTime="2013-02-04T07:57:00"
    CreationDateTime="2013-02-06T10:30:00">
```

The semantic constraints are expressed throughout the rest of this document.

The ODM version 1.3 XML Schema file can be used with a validating XML parser to test that an ODM file is syntactically valid. To ensure semantic correctness the use of an ODM-specific validation application is recommended.

2.3 System Conformity

The value of the ODM standard, like any information standard, is enhanced when systems can be developed with the assumption that ODM files have a high level of conformity. However, the ODM also provides value in that it provides both a standard and a technology for the interchange of clinical data where none has existed in the past. The conformity statements below represent an attempt at balancing the need for quality control with respect to the standard with the need for flexibility when encouraging adoption and innovation:

A computing system that processes information in ODM format can claim conformance to this standard only if it obeys the following rules.

1. Generated ODM files must satisfy all the correctness rules in this standard, both syntactic and semantic.
2. A receiving system must be able to read any ODM file that satisfies all the correctness rules in this standard, both syntactic and semantic.
3. ODM files must validate against the ODM schema for the ODM version indicated in the ODM root element.
4. Information included in generated ODM files must be accurate according to the rules of this standard as defined in this specification.
5. A receiving system must interpret information read from an ODM file accurately according to the rules of this standard as defined in this specification.
6. Generated ODM files need not include information that is not normally handled or stored by the generating system.
7. A receiving system may selectively ignore information read from an ODM file if that information is not normally handled or stored by the receiving system.
8. A receiving system may constrain the range of data values, keys, names, and so on, that it is capable of handling or storing.
9. Systems that receive ODM clinical data files but do not normally support one or more of the datatypes specified in section 2.14, should accept clinical data of the unsupported types as text.
10. All system limitations (rules 6 through 9) must be documented.
11. If conformity is dependent on certain modes or settings, this must also be documented.

2.4 Vendor Extensions

The ODM has been designed to provide a standard format for interchange and archive of clinical study data and metadata. Although the model has proven to be usable with a wide variety of clinical data applications, it is sometimes the case that individual clinical data systems store information that cannot be expressed conveniently in the ODM model. To encourage these systems to adopt the use of the ODM standard for interchange and archive and to improve interoperability of clinical research applications, the use of the ODM vendor extensions is recommended.

Requirements for vendor extensions to the ODM model are:

1. The vendor must supply an XML Schema fully describing their extended ODM format.
2. Extended ODM files should reference the proper extension Schema.
3. The extension may add new XML elements and attributes, but may not render any standard ODM elements or attributes obsolete. Vendor extensions cannot be used for information that is normally expressed using other ODM elements.
4. All new elements and attributes must use distinct XML namespaces to insure that there are no naming conflicts with other vendor extensions.
5. Removing all vendor extensions from an extended ODM file must result in a meaningful and accurate standard ODM file.
6. Vendors should be able to produce ODM files free of any vendor extensions upon request.

Applications that use extended ODM files must also accept standard ODM files.

Note: Vendor extensions that carry generally useful information should be brought to CDISC's attention for possible future standardization.

Note: It is possible to write a filter that removes all non-standard attributes and elements from an extended ODM file, either as a stand-alone program or as part of normal input processing. Receiving systems are encouraged to be forgiving when encountering extended ODM files.

2.5 Changes from Previous Versions (non-normative)

ODM 1.3.2 includes all of the elements and attributes defined in ODM 1.3.1. New elements and/or attributes have been added to provide new capabilities. Errors have been fixed, clarifications have been made, and minor edits have been made for consistency and clarity in various sections. No elements or attributes have been deprecated in this version of ODM.

You can view this specification with changes made in Version 1.3.2 highlighted by editing a copy of this file and removing the comment tags surrounding the ODM 1.3.2 style definition near the top of the file.

2.5.1 OrderNumber on CodeListItems and EnumeratedItems

OrderNumbers have been added to CodeListItem and EnumeratedItem.

2.6 Entities and Elements

The ODM model assumes that a study's clinical data will consist of several kinds of entities. These include subjects, study events, forms, item groups, items, and annotations.

An *item* is an individual clinical data item, such as a single systolic blood pressure reading. Items are collected together into item groups.

An *item group* is a closely related set of items that are generally analyzed together. (Item groups are sometimes referred to as "records" and are associated with "panels" or "tables".) Item groups are aggregated into forms.

A *form* is analogous to a page in a paper CRF book or electronic CRF screen. A form generally collects a set of logically and temporally related information. A series of forms is collected as part of a study event.

A *study event* is a reusable package of forms usually corresponding to a study data-collection event.

A study protocol design may include zero or more planned subject visits at which data will be collected. Each planned visit may correspond to one or more ODM study events and ODM study events may be used for more than one subject visit.

A *subject* is a patient participating in the study.

An *annotation* is a comment applied to a subject, study event, form, item group, or item. Annotations can also be applied to pairs of entities.

The metadata of a study describes the types of study events, forms, item groups, and items that are allowed in the study. The metadata entities include StudyEventDefs, FormDefs, ItemGroupDefs, and ItemDefs.

A *StudyEventDef* describes a particular type of study event (mostly by listing the types of forms it can contain).

A *FormDef* describes a particular type of form.

An *ItemGroupDef* describes a particular type of item group.

An *ItemDef* describes a particular type of item.

The clinical data of a study will typically have many actual study events corresponding to each StudyEventDef, many actual forms corresponding to each FormDef, and so on.

An ODM file (like any XML file) consists of a tree of *elements*. The clinical data elements in an ODM file represent either the state of a clinical entity or a change to the state of that entity. These elements include the *SubjectData*, *StudyEventData*, *FormData*, *ItemGroupData*, *ItemData*, and *Annotation* elements.

Each such element contains key attributes that identify the data entity that it provides information for. Frequently, many data elements will correspond to a single data entity. This can occur when a series of updates are being applied to a single entity, or when an audit trail is being represented.

Similarly, there are XML elements corresponding to metadata entities.

Note: Not all clinical studies organize their data into visits or forms. For such a study, it is appropriate to have a single common study event per subject, and/or a single common form per study event.

2.7 Clinical Data Keys

There are two kinds of keys in the ODM: *internal* and *external*. Internal keys are used to designate entities within the model, and to allow cross-references between entities within (and between) ODM files. Internal keys are not for human use, and are assumed to be unchanging. Internal keys include subject keys, element OIDs, and repeat keys.

To fully identify a clinical data entity, the following internal keys are needed:

Kind of Entity	Identifying Keys
study	StudyOID
subject	above plus SubjectKey
study event	above plus StudyEventOID and StudyEventRepeatKey
form	above plus FormOID and FormRepeatKey
item group	above plus ItemGroupOID and ItemGroupRepeatKey
item	above plus ItemOID
annotation	keys for the annotated entity plus SeqNum

A StudyOID uniquely identifies a study. A SubjectKey uniquely identifies a subject within a study. SubjectKeys cannot be used to identify a subject across studies.

A StudyEventOID uniquely identifies a StudyEventDef within a study. However there may be several study events of a particular type for a given subject. Thus, to fully identify a particular study event, we need the StudyOID, the SubjectKey, the StudyEventOID, and a

StudyEventRepeatKey.

Each form belongs to a study event, and can be identified (within that study event) by a FormOID (which gives its type) and a FormRepeatKey.

Each item group in clinical data belongs to a form, and can be identified (within that form) by an ItemGroupOID (which gives its type) and an ItemGroupRepeatKey.

Other items groups are reference data, and can be identified solely by an ItemGroupOID and an ItemGroupRepeatKey.

Each item belongs to an item group. However an item may be referenced only once within an item group. Thus, only the ItemOID is needed to uniquely identify an item within its item group.

External keys are any keys used by clinical personnel. These include subject randomization codes, site codes, and so on. In the ODM, external keys are represented as if they were clinical data - i.e. using one of the currently defined ItemData types. Thus external keys can be changed as needed, and are subject to normal auditing processes.

2.8 Single Files and Collections

An ODM document (file) can be viewed as a set of instructions telling how to modify a target database so that it can be brought into alignment with a source database. A single ODM document could contain all the information about the source database -- all the subjects, all the metadata, all the clinical data, and all the changes to each item of clinical data. However, it's not always convenient to create such a large document. So we allow a series of ODM files to incrementally provide this information.

Each ODM document has a FileOID attribute (see the ODM element). The FileOID should uniquely identify the document.

An ODM document can also have a PriorFileOID attribute. This attribute gives the FileOID of the immediately preceding file in a series. Thus, a collection of files linked by PriorFileOID can be used to incrementally describe a source database. (Of course, the first document in the series has no PriorFileOID attribute.)

Data in one file of a series is allowed to depend on definitions contained in an earlier file in the series. If a file contains either ReferenceData or ClinicalData then that file must either contain the necessary metadata definitions to interpret that data directly, or there must be a previous file in the series that contains these definitions. Similarly, if a file contains ClinicalData, the file must either contain the administrative data referenced in the audit and signature records, or there must be a previous file in the series that contains that data.

A tree-structured collection of ODM files (where more than one file references the same prior file) is also allowed.

An ODM document (or collection) can contain information on more than one study. To simplify the discussion, we will occasionally write as if only one study was present. If a file (or collection of files) contains multiple studies, the rules described below should be applied to each of the studies independently.

Note: The FileOID identifies the document content. It does not change just because the document is copied.

Note: FileOIDs should be universally unique if at all possible. One way to ensure this is to prefix every FileOID with an Internet domain name owned by the creator of the ODM file or database (followed by a /). For example, FileOID="BestPharmaceuticals.com/Study5894/1" might be a good way to denote the first file in a series for study 5894 from Best Pharmaceuticals.

Note: Similarly, StudyOIDs should be universally unique if at all possible. For example, StudyOID="BestPharmaceuticals.com/Study5894".

Note: Applications may need to store ODM documents that they receive and later retrieve them based on their FileOID. In particular, interpreting a new file in a series will require locating the prior file by its FileOID. One easy way to do this is to use the (possibly transformed) FileOID as part of the file name itself. For example, the ODM document mentioned above could be stored in ...\BestPharmaceuticals.com\Study5894\1.xml.

Other File Attributes

As implied above, most ODM documents will contain only part of the total information (current and historical) held in the source database.

The information that is sent in a given document can vary along several dimensions. Some examples of the contents of a document are:

- just metadata,
- just clinical data,
- just domain data (SDTM),
- a series of updates of a single value for a single subject,
- the current state of the entire study,
- the current state of a subset of a study (particular subjects, particular forms, etc),
- the changes to the study since the last communication, or
- a full history of all changes ever made to the study.

Because of this variability, it is also important that each document describe itself, so that the consumer of the document knows what to expect of it. To address these needs, the ODM element has several attributes for describing the current document.

The CreationDateTime attribute tells when the ODM document was created. In contrast, the AsOfDateTime attribute tells when the document content was accurate. This is of particular importance when a series of files is used to give an evolving view of a changing database.

The FileType and Granularity attributes allow the document sender to define the scope, across time and data, that a particular document spans. The Archive attribute allows the sender to assert that the contents of the document meet a specific set of criteria that qualifies it as an electronic record defined in the FDA 21 CFR 11 regulation. Finally, the Description attribute provides the sender a text string in which to give details, as elaborately as necessary, to supplement the other attributes in describing the document. This section details the values these attributes can take, and discusses the use of the values in a single document and in a series of documents.

FileType

An ODM document's FileType must be either Snapshot or Transactional. A Snapshot document shows how to re-create the current state of the source database with respect to the included data, but does not show how it got to be in that state over time. A Transactional document shows, for each included entity, both the latest state of the source database, and (optionally) some prior states of that entity in the source database. The TransactionType attribute need not be present in a Snapshot document. When processing a Transactional document, the rules for ordering a set of transactions for a single data point are those described in the Element Ordering Section.

Granularity

An ODM document's Granularity is intended to give the sender a shorthand way to describe the breadth of the information in the document, for certain common types of documents.

Here are the intended meanings of these categories:

Category	Document contains
All	Any and all types of data and metadata
Metadata	Metadata only
AdminData	Admin data only
ReferenceData	Reference data only
AllClinicalData	Clinical data only
SingleSite	Clinical data for a single site only
SingleSubject	Clinical data for a single subject only

If these shorthand categories are not sufficient, use the document's Description attribute to give details.

Archival

The Archival attribute is optional.

Archival=Yes states that this file (or collection of files) is intended to meet the requirements of an electronic record as defined in 21 CFR 11. More specifically, the file (or set of files) must clearly establish a complete and non-redundant set of insertions, updates, and deletions of data values with full auditing and signature information (if available).

Archival=Yes requires

- The current file must be Transactional.
- If the current file is one of a collection, all other files in the collection must be both Archival and Transactional.
- The set of transactions must be both complete and non-redundant within the file or collection of files. In particular each file must contain:
  - no transactions that have an audit date of prior to the AsOfDateTime of its PriorFile (if any),
  - *all* transactions up to its AsOfDateTime. (For Granularity=All, this means all transactions in the study. See Granularity for the meaning of "all" for other Granularity settings),
  - all AuditRecord and Signature information available in the source database, and
  - no Upsert transactions.

2.9 Transactions

Each data element has an optional TransactionType attribute. This attribute can be one of *Insert*, *Update*, *Remove*, *Upsert*, or *Context*.

A TransactionType of Insert means that the data entity is new (does not currently exist in the study) and must be added to the study data along with all the properties provided. It is an error if the data entity already exists, or if the data entity's parent does not exist.

A TransactionType of Update means that the data entity already exists and must be modified to have the new properties provided. Properties not mentioned are not modified. It is an error if the data entity does not exist.

A TransactionType of Remove means that the data entity already exists and must be deleted along with all its properties and children. It is an error if the data entity does not exist.

Note: The use of the word "delete" in this section does not imply that all records of the data entity are expunged, just that the next transaction on that entity (if any) must be an Insert rather than Update or Remove.

A TransactionType of Upsert is the same as Update if the data entity exists, and the same as Insert otherwise.

Note: Upsert allows the sender to ignore whether the data entity exists or not. (A capability that may be needed by certain data collection systems.) **This capability is potentially dangerous and may be removed in future versions of this standard.**

A TransactionType of Context means that the data is explicitly being resent for context purposes only. An example of this would be sending an ItemGroup containing demographic data every time (with a TransactionType of Context) to permit matching and checking of Patient IDs between the sender and receiver systems.

A child element that does not specify its own TransactionType inherits the TransactionType of its parent. The TransactionType of a top level data element must be explicit.

Transaction processing proceeds in the order in which data elements appear in the ODM document. If two transactions for an entity are included in an ODM document, they are applied in the order in which they appear.

An entity having a TransactionType of Remove may include descendents. Each such descendent must also specify a TransactionType of Remove, or else not specify a TransactionType attribute (in which case the descendent inherits the TransactionType of the parent). A program that processes ODM files should look ahead far enough to determine if an element breaks this rule before processing the element. Note that data corresponding to such descendents may have been deleted via a cascading effect resulting from the deletion of the parent. In such cases, the processor may or may not have generated an audit trail entry for each descendent data entity deleted.

If a Snapshot document includes a TransactionType, only TransactionType=Insert is permitted.

2.10 Element Ordering

The elements in the data portion of an ODM file that apply to a single entity must occur in temporal order. That is, the transactions on each entity are applied in the order in which that entity's elements occur in the file, and Signatures and AuditRecords must occur in increasing value of their DateTimeStamps.

In general, the elements for distinct entities do not have to be ordered relative to one another. However, when an entity is signed, all transactions on that entity that occurred before the time of the signature must be present earlier in the same ODM file or in a prior ODM file in a linked series.

All the DateTimeStamps in a file must precede the CreationDateTime of that file.

In a linked series of ODM files, the AsOfDateTimes of those files must occur in temporal order, and the DateTimeStamps in each file must be later than the AsOfDateTimes of the prior file.

A data entity (like a form) can be represented as a series of data elements. These elements can contain data for disjoint parts of the entity (properties and children), sequential changes to the same parts, or a mixture of the two.

In a study, the initial or default value of any clinical data is assumed to be NULL. Thus it is not necessary to include NULL valued items in a Snapshot transfer. Neither is it necessary to include a NULL value as the first in a series of insertions and updates of an item.

Systems that generate ODM files should attempt to minimize the number of elements in such files. Thus, if two elements for the same entity can be merged into one without changing the semantics of the events described (and without violating any of the ordering rules), the merged version is recommended. Similarly, unnecessary elements should be omitted.

2.11 Element Identifiers and References

In the ODM, there are many instances where one element needs to reference another -- both within the same file and across files within a series of ODM documents. To accomplish this, the target element is given a unique "name" (its OID). All elements that need to reference that target element just use its OID.

Example: Say we have an item definition (ItemDef) called "Systolic Blood Pressure". We set the OID attribute of this ItemDef to "SBP". Then any specific instance of that data item (ItemData) can link back to the definition by setting its own ItemOID attribute to "SBP".

The OIDs used have to be unique, but only within certain contexts. The uniqueness rules are:

- OIDs for each element type inside a MetaDataVersion (e.g., FormDef) must be unique within the scope of that MetaDataVersion.
  - It is best practice to make OIDs unique within the entire MetaDataVersion, however it is allowable to use the same OID for different element types if necessary, e.g., a FormDef may have the same OID as an ItemGroupDef. This is not recommended.
  - Elements in a MetaDataVersion may have the same OID as ones in an Included MetaDataVersion. An element in a MetaDataVersion with the same OID as the same element type in an Included MetaDataVersion is interpreted as replacing the earlier definition. See 3.1.1.3.1 Include.
- MetaDataVersion OIDs must be unique within the containing Study.
- ArchiveLayout OIDs must be unique within a single FormDef. ArchiveLayout OIDs may be reused in different FormDefs.
- Study OIDs must be unique within the containing ODM.
- Measurement Unit OIDs must be unique within the containing Study.

- User, Location and Signature OIDs must be unique within the containing AdminData.

Whenever an element uses an OID to reference another element, the referenced element must occur within the same document as the reference *or* within a prior document in a linked series. Thus, the referenced element can always be found by tracing back through the PriorFileOID links.

Note: The fact that OID references need to cross files prevents the use of standard XML ID and IDREF values (which are references solely within a single file). This is highlighted in our element definitions by using the types *oid* and *oidref* instead of *ID* and *IDREF*.

Note: The above reference rules imply the following.

- You cannot send a reference to an OID value unless that OID value has been defined, either in the current document, or in a prior document within the series.
- If a definition identified by an OID has changed, the change in its definition must be transmitted before or in the document in which the changed definition is referenced.
- A document that initiates a series must contain all definitions that it references.
- A Transactional document must include any supporting definitions that are new or changed relative to its prior document. For example, suppose that document B3207 includes references to item definition X, and that a codelist constraint has been added to X since B3207's predecessor document. In that case, B3207 must also include a MetaDataVersion element (with an new OID) containing an updated definition of item X.

2.12 Syntax Notation

The syntax of each XML element is specified as follows:

Element-Name

Body:
content-specification

Attributes:
attribute-name data-format optionality semantic-details
...

The content specification can be "EMPTY", a data format name, or an element group. EMPTY means that this element has no body (nested content). A data format name means that the body can be any text string matching the named format. An element group means that the named elements can be directly nested within this element.

An element group consists of one or more element names (or element groups) enclosed in parentheses, and separated with commas or vertical bars. Commas indicate that the elements (or element groups) must occur in the XML sequentially in the order listed in the group. Vertical bars indicate that exactly one of the elements (or element groups) must occur. An element or element group can be followed by a ? (meaning optional), a * (meaning zero or more occurrences), or a + (meaning one or more occurrences).

Attributes are listed in tabular form, one attribute per row, including the attribute's name, its data format, its optionality, and possibly some semantic details. All attributes are required unless the optionality column contains "*(optional)*". The ODM foundation schema implements the specification using XML Schema.

2.13 Data Formats

All XML attribute values and some element bodies are text strings. These strings are defined by data format. Each data format specifies the allowable form of the string, the corresponding XML Schema datatype, and the intended use of the value. The set of ODM data formats follows:

Format Name	**Schema Datatype**	**Allowed String Pattern**
integer	xs:integer	**-**?digit+
positiveInteger	xs:positiveInteger	+?digit+ (and representing an integer number > 0)
nonNegativeInteger	xs:nonNegativeInteger	+?digit+ (and representing an integer number >= 0)
float	xs:decimal	**-**?digit+(**.**digit+)?
date	xs:date	YYYY**-**MM**-**DD
time	xs:time	hh**:**mm**:**ss(**.**n+)? (((**+**|**-**)hh**:**mm)|Z)?
datetime	xs:dateTime	YYYY**-**MM**-**DD**T**hh**:**mm**:**ss(**.**n+)? (((**+**|**-**)hh**:**mm)|Z)?
text	xs:string	*any sequence of characters*
oid	xs:string	*any sequence of characters* (minLength="1")
oidref	xs:string	*any sequence of characters* (minLength="1")

ID	xs:ID	*any sequence of characters* (minLength="1")
IDREF	xs:IDREF	*any sequence of characters* (minLength="1")
subjectKey	xs:string	*any sequence of characters* (minLength="1")
repeatKey	xs:string	*any sequence of characters* (minLength="1")
name	xs:string	*any sequence of characters* (minLength="1")
sasName	xs:string	(letter | _)(letter | digit | _)* (maxLength="8")
sasFormat	xs:string	(letter | _ | **$**)(letter | digit | _ | **.**)* (maxLength="8")
fileName	xs:anyURI	*any sequence of characters*
languageTag	xs:language	LL (**-**CC)* *(see below)*
string	xs:string	Semantically equivalent to **text** but directly supported as XML Schema datatype
boolean	xs:boolean	(true | false | 1 | 0)
double	xs:string	(((\+|-)?[0-9]+(\.[0-9]+)?((D|d|E|e)(\+|-)[0-9]+)?)|(-?INF)|(NaN))
hexBinary	xs:hexBinary	hex-encoded binary stream data
base64Binary	xs:base64Binary	binary stream encoded using Base64 Alphabet
hexFloat	xs:hexBinary	up to 16 characters
base64Float	xs:base64Binary	up to 12 characters
partialDate	xs:date	**[**YYYY**[-**MM**[-**DD]]]
partialTime	xs:time	**[**hh**[:**mm**[:**ss(**.**n+)? (((**+|-**)hh**:**mm)|Z)?]]]
partialDatetime	xs:dateTime	**[**YYYY**[-**MM**[-**DD**[T** hh**[:**mm**[:**ss(**.**n+)? ((**+|-**)hh**:**mm)?]]]]]
intervalDatetime	xs:string	partialDatetime/partialDatetime)| (durationDatetime/partialDatetime)| (partialDatetime/durationDatetime)
durationDatetime	xs:duration	(((+|-)?P((((n(n+)?)Y)?((nn+)?)M)? ((nn+)?)D)?)(T(((n(n+)?)H)? ((n(n+)?)M)?((n(n+)?) ((\.n+)?)S)?)?)?|(((n(n+)?)W))))
incompleteDatetime	xs:string	[YYYY|-]-[MM |-]-[DD|-]]**T**[hh|-]:[mm|-]:[ss.s|-][? (+|-)nn:nn|Z]
incompleteDate	xs:string	[YYYY|-]-[MM |-]-[DD|-]
incompleteTime	xs:string	**T**[hh|-]:[mm|-]:[ss.s|-][? (+|-)nn:nn|Z]
URI	xs:anyURI	

Numbers are represented in base 10. Floats are allowed to have fractional parts.

Note: Certain integer attributes (KeySequence, OrderNumber, and SeqNum) are used to define an order among related entities. For these attributes, we recommend using consecutive integer values starting at 1.

A *text* value is any character or sequence of characters valid within the XML document encoding. When embedded in ODM files, text strings must be represented using the standard XML quoting and character escaping rules.

Dates are represented in the Gregorian calendar. The year (YYYY) ranges from 0001 to 9999. The month (MM) ranges from 01 to 12. The day (DD) ranges from 01 to 31. ODM dates are equivalent to the XML Schema datatype **date**.

Times are clock readings within a 24 hour period. The hour (hh) ranges from 00 to 23. The minutes (mm) and the seconds (ss) range from 00 to 59. The optional fractional part (.nnn) expresses fractional seconds. The timezone can be either expressed using the ±hh:mm format or using the "Z" format (UTC - Coordinated Universal Time)

Datetimes combine a date, a time and (optionally) a timezone.

Dates, *times*, and *datetimes* are to be interpreted as local clock readings at the place the data was collected. In a datetime, the ±hh:mm is the offset in hours and minutes to add to (or subtract from) Universal Time to get the local clock reading at the time the data was collected. A missing timezone specification means that the relationship between the local clock and Universal Time is not known.

Example: 3:14 pm on 3 January 2001 in Chicago (6 timezones west of Greenwich, standard time) would be represented as "2001-01-03T15:14:00-06:00". 3.5 seconds after midnight on the morning of July 20th 2001 in Chicago (daylight time) would be represented as "2001-07-20T00:00:03.500-05:00".

Note: The above formats for *dates*, *times*, and *datetimes* are compatible with ISO 8601 (refer to "Complete Representations"). The ODM date, time, and datetime data formats must contain all element components.

partialDate, *partialTime* and *partialDatetime* formats are compatible with ISO8601(refer to "Representations other than complete"). They are provided to support transmission of dates in which one or more less significant components are unknown at the time of data capture. For example, for a partialDate value the day of the month may be unknown.

The *intervalDatetime* format is represented as a pair of partialDatetime values representing the beginning and end of the time interval, a beginning partialDatetime and a durationDatetime, or a durationDatetime and an ending partialDatetime separated by a slash (refer to "Representation of time-interval identified by its start and its end", "Representation of time-interval identified by its start and its duration", and "Representation of time-interval identified by its duration and its end" respectively).

The *durationDatetime* data format is represented as an ISO 8601 duration (refer to "Representation of time-interval by duration only") . For example, 4 hours and 35 minutes is represented as PT4H35M. In the ODM implementation, carry-over of individual component values and negative durations are supported.

The *incompleteDatetime* format enables transmission of datetime values where one or more of the components -- not necessarily of lower significance -- is missing. The full set of delimiters is used as for a complete datetime, the missing component(s) are represented by a single dash (refer to "Truncated Representations"). In the ODM implementation, most meaningful partialDatetime forms are also supported. For example, 5 minutes after an unknown hour on the 15th day of an unknown month in 2004 is represented as 2004---15T-:05.

The *incompleteDate* format enables transmission of date values where one or more of the components -- not necessarily of lower significance -- is missing. The full set of delimiters is used as for a complete date, the missing component(s) are represented by a single dash (refer to "Truncated Representations").
For example, 2001---30 means the 30th of an unknown month in the year 2001, and ----30 means the 30th of an unknown month and unknown year.

The *incompleteTime* format enables transmission of time values where one or more of the components -- not necessarily of lower significance -- is missing. The full set of delimiters is used as for a complete time, the missing component(s) are represented by a single dash (refer to "Truncated Representations").
For example, -:55:30 means 30 seconds after the 55th minute of an unknown hour, and -:-:30 means 30 seconds after an unknown minute of an unknown hour.

The hexFloat and base64Float data formats are binary types intended for precise exchange of floating-point data in a machine/platform independent manner. The types are based on the IBM Mainframe format used in SAS V5 Transport format -- described in the SAS Technical Support document "TS-140 The Record Layout of a Data set in SAS Transport (XPORT) Format" with the appropriate direct translation of the converted value to either hexBinary format or the base64Binary (MIME) representation. This secondary conversion ensures XML encoding compatibility.

Note: Data values transmitted as hexFloat or Base64Float will not be humanly readable in an XML markup or by a standard XML authoring tool. For this reason, the decision to use these types implies that an application will be needed to process data received in this format.

Values for hexFloat and hexBinary data are encoded as character tuples consisting of two hexadecimal digits ([0-9A-F]).

Values for base64Binary data are encoded using the 65 characters of the Base64Alphabet. These include a-z, A-Z, 0-9, +, /, = and the whitespace characters defined in [XML 1.0 (Second Edition)].

The hexBinary and base64Binary datatypes may also be used for exchange of floating point values, but may be subject to platform or application dependencies. Implementers should note that this could result in an inability to determine whether the receiving application has interpreted a value as a string rather than as a float. Encoded values using these datatypes may be exchanged reliably only across similar hardware platforms.

Note: ODM 1.1 allowed a timezone specification of -99:99 to designate "no timezone information." This format is no longer supported.

An *oid* (OID) is a string that uniquely identifies an element (see Element Identifiers and References for details). An *oidref* is the use of an OID to reference an element (as opposed to the defining occurrence of that OID). OIDs and oidrefs are non-empty strings.

A *subjectKey* or *repeatKey* helps designate a clinical data entity (see Clinical Data Keys). SubjectKeys are non-empty strings.

A *name* is intended to be a human readable name for some entity. Names are non-empty strings.

A *sasName* (or *sasFormat*) is any valid SAS Version 5 Transport Format name (or format). These are limited to 8 characters in length.

A *fileName* designates a file. File names are system dependent, and expressed relative to the directory that contains the ODM file being processed. FileNames are non-empty strings.

A *languageTag* represents a natural language or a country-specific variant of a natural language. The allowed values are specified in IETF RFC 3066: Tags for the Identification of Languages. Note that a *languageTag* cannot be null.

Example: "fr-CA" denotes the French language, Canadian variant. See TranslatedText for a discussion of how languageTags are used.

In general for data formats that allow NULLs, you should use an empty string (e.g., attribute-name="") to represent a NULL attribute value. To represent a NULL for data transmitted in an element body, send the element as empty (e.g. <element-name/>). There is a special IsNull indicator for clinical data, to differentiate between the case where there is no known value, and the case where you want to replace a value with NULL. See the IsNull attribute of the ItemData element.

See the ODM Foundation Schema for definitions of the enumeration types and the ODM Example XML file for the further examples of legal and illegal values for each of the ODM clinical data formats.

2.14 ItemData*[TYPE]*

Starting in ODM 1.3.0, there are two forms of the ItemData element -- the element used by the ODM for transmitting clinical data item values. These are the untyped and typed forms. The ItemData element, present in all previous ODM versions, is the form used for untyped data transmissions. The ItemData*[TYPE]* elements are the form used for typed data transmission.

The ODM schema for Version 1.3.2 includes the following ItemData*[TYPE]* elements:

ItemData**Any**
ItemData**String**
ItemData**Integer**
ItemData**Float**
ItemData**Date**
ItemData**Time**
ItemData**Datetime**
ItemData**Boolean**
ItemData**Double**
ItemData**HexBinary**
ItemData**Base64Binary**
ItemData**HexFloat**
ItemData**Base64Float**
ItemData**PartialDate**
ItemData**PartialTime**
ItemData**PartialDatetime**
ItemData**DurationDatetime**
ItemData**IntervalDatetime**
ItemData**IncompleteDatetime**
ItemData**IncompleteDate**
ItemData**IncompleteTime**
ItemData**URI**

All typed data value elements carry their data value as *the element's PCDATA content*. For example, an ItemDataInteger element would appear as:

```
<ItemDataInteger ItemOID="ID.114">100</ItemDataInteger>
```

Use of the ItemData*[TYPE]* element form permits the application receiving the ODM XML file to validate the ClinicalData when parsing the ODM file.

The ItemData***Any*** element is provided as an escape mechanism for transmitting clinical data values that do not have the correct DataType for the corresponding ItemDef. Receiving applications are not required to load the data values transmitted using ItemDataAny elements into the corresponding database field.

Typed and untyped data must not both be used within a single ODM file.

Note: in untyped data transmission, special characters such as ' , " , < , > , & have to be escaped. In the typed data transmission form, these can be transmitted either within a CDATA block, or escaped.

```
<ItemDataString ItemOID="ID.115">
  <![CDATA[
      The five special characters that must normally be escaped are ' , " , < , > , &
      ]]>
</ItemDataString>
```

Systems that generate or receive ODM files are encouraged to provide typed data transmission.

Systems that receive ODM clinical data files but do not normally support one or more of the datatypes listed in this section should accept clinical data of the unsupported types as text. In cases where these systems transmit clinical data by generating a new ODM file with the unsupported type, the generated file must include the original datatype information.

For example, if a system that does not normally support hexBinary or hexFloat datatypes receives an ODM XML file containing data for an Item defined as having datatype 'hexFloat', the clinical data for the Item may be stored as Text. If that system generates an ODM file that includes data for that item, the ItemDef must still define the datatype as hexFloat.

3 General Elements

3.1 ODM

Body:

(**Study***, **AdminData***, **ReferenceData***, **ClinicalData***, **Association***, **ds:Signature***)

Attributes:

Description	text	*(optional)*	The sender should use the Description attribute to record any information that will help the receiver interpret the document correctly.
FileType	(Snapshot | Transactional)		Snapshot means that the document contains only the current state of the data and metadata it describes, and no transactional history. A Snapshot document may include only one instruction per data point. For clinical data, TransactionType in a Snapshot file must either not be present or be Insert. Transactional means that the document may contain more than one instruction per data point. Use a Transactional document to send both what the current state of the data is, and how it came to be there.
Granularity	(All | Metadata | AdminData | ReferenceData | AllClinicalData | SingleSite | SingleSubject)	*(optional)*	Granularity is intended to give the sender a shorthand way to describe the breadth of the information in the document, for certain common types of documents. All means the entire study; Metadata means the MetaDataVersion element; AdminData and ReferenceData mean the corresponding elements; AllClinicalData, SingleSite, and SingleSubject are successively more tightly focused subset of the study's clinical data. If these shorthand categories are not sufficient, use the Description attribute to give details.
Archival	(Yes | No)	*(optional)*	Set this attribute to Yes to indicate that the file is intended to meet the requirements of an electronic record as defined in 21 CFR 11. See Single Files and Collections for an fuller discussion of the meaning of this attribute, as well as its interaction with other ODM attributes. Use this attribute *only* when FileType is Transactional.
FileOID	oid		A unique identifier for this file.
CreationDateTime	datetime		Time of creation of the file containing the document.
PriorFileOID	oidref	*(optional)*	Reference to the previous file (if any) in a series.
AsOfDateTime	datetime	*(optional)*	The date/time at which the source database was queried in order to create this document.
ODMVersion	(1.2 | 1.2.1 | 1.3 | 1.3.1 | 1.3.2)	*(optional)*	The version of the ODM standard used.
Originator	text	*(optional)*	The organization that generated the ODM file.
SourceSystem	text	*(optional)*	The computer system or database management system that is the source of the information in this file.
SourceSystemVersion	text	*(optional)*	The version of the "SourceSystem" above.
ID	ID	*(optional)*	May be used by the ds:Signature element to refer back to this element.

The AsOfDateTime is the latest date/time of any new data or updates to data included in the current file. This attribute is implied, and if omitted, it is assumed that the AsOfDateTime is equal to the CreationDateTime. It is an error to supply an AsOfDateTime that is later than the CreationDateTime.

The ODMVersion attribute was not defined in ODM 1.1, so a missing ODMVersion should be interpreted as "1.1".

Documents based on ODM 1.3.2 must have ODMVersion="1.3.2" set in order to use the new features in this version. Documents based on ODM 1.2.0 must have ODMVersion="1.2", documents based ODM 1.3.0 must have ODMVersion="1.3" and documents based on ODM 1.3.1 must have ODMVersion="1.3.1" for backwards compatibility purposes.

There is an extended discussion of these attributes in the section titled Single Files and Collections.

3.1.1 Study

Body:

(**GlobalVariables**, **BasicDefinitions**?, **MetaDataVersion***)

Attributes:

OID oid

Contained in:
ODM

This element collects static structural information about an individual study.

3.1.1.1 GlobalVariables

Body:
(**StudyName**, **StudyDescription**, **ProtocolName**)

Attributes:
NONE

Contained in:
Study

GlobalVariables includes general summary information about the Study.

3.1.1.1.1 StudyName

Body:
name

Attributes:
NONE

Contained in:
GlobalVariables

This is a short external name for the study.

3.1.1.1.2 StudyDescription

Body:
text

Attributes:
NONE

Contained in:
GlobalVariables

This is a free-text description of the study.

3.1.1.1.3 ProtocolName

Body:
name

Attributes:
NONE

Contained in:
GlobalVariables

This is the sponsor's internal name for the protocol.

3.1.1.2 BasicDefinitions

Body:
(**MeasurementUnit***)

Attributes:
NONE

Contained in:

Study

3.1.1.2.1 MeasurementUnit

Body:
(**Symbol**, **Alias***)

Attributes:
OID oid
Name text

Contained in:
BasicDefinitions

The physical unit of measure for a data item or value. The meaning of a MeasurementUnit is determined by its Name attribute. Examples include kilograms, centimeters, cells/milliliter, etc.

Note: Standardizing the names of measurement units is beyond the scope of the ODM.

3.1.1.2.1.1 Symbol

Body:
(**TranslatedText**+)

Attributes:
NONE

Contained in:
MeasurementUnit

A human-readable name for a measurement unit.

3.1.1.2.1.1.1 TranslatedText

Body:
text

Attributes:
xml:lang languageTag *(optional)*

Contained in:
Decode, ErrorMessage, Question, Symbol, Description

Human-readable text that is appropriate for a particular language. TranslatedText elements typically occur in a series, presenting a set of alternative textual renditions for different languages.

To find the text appropriate for a target language with tag LT, search for a TranslatedText element whose xml:lang attribute matches LT exactly (ignoring case). If that fails, remove the ending subtag from LT and repeat. If that fails, search for a TranslatedText without an xml:lang attribute and use that. If none is found, there is no suitable text available. E.g.

TranslatedTexts Present	Requested	Process
<TranslatedText xml:lang="fr-CA">... <TranslatedText xml:lang="en-GB">... <TranslatedText>...	fr-FR	Look for xml:lang="fr-FR". This is not found, so look for xml:lang="fr". This is not found, so look for a TranslatedText with no xml:lang. This is the text that should be used.

To avoid ambiguity, a particular language tag must not occur more than once in a series of TranslatedText elements. Also, it is not permitted to have more than one TranslatedText element without an xml:lang attribute within the same parent.

Note: The xml:lang attribute is part of the XML standard.

3.1.1.3 Metadata Elements

The metadata for a study is defined in a series of MetaDataVersion elements. Through this mechanism (multiple MetaDataVersion elements), the model supports the incremental deployment of "mid-study changes", and thus can handle a situation where multiple versions of the metadata are being used simultaneously (perhaps due to delays in IRB approval at various sites).

The way metadata versioning works is as follows. An individual Study element may contain multiple MetaDataVersions, reflecting one or more mid-stream study design changes. The initial version contains a full set of metadata. Each subsequent version typically contains only modified or newly added metadata elements, inheriting the previous metadata by an explicit reference. The same metadata elements in different versions will have the same OID. This approach is used to allow the older versions of the metadata to remain intact, while simultaneously providing a concise way to represent changes.

3.1.1.3 MetaDataVersion

Body:

(**Include**?, **Protocol**?, **StudyEventDef***, **FormDef***, **ItemGroupDef***, **ItemDef***, **CodeList***, **ImputationMethod*** ***Deprecated***, **Presentation***, **ConditionDef***, **MethodDef***)

Attributes:

OID	oid	
Name	name	
Description	text	*(optional)*

Contained in:

Study

A metadata version defines the types of study events, forms, item groups, and items that form the study data.

The Include element references a prior metadata version. This causes all the definitions in that prior metadata version to be automatically included in this one. Any of the included definitions can be replaced (overridden) by explicitly giving a new version of the definition (with the same OID) in the new metadata version. See Include for more information on how the Include element works. New definitions (with new OIDs) can be added in the same way.

Note: The current metadata model does not include any scheduling information such as the time ordering of events, the spacing of events, conditional occurrence of events, and so on. The analogous information for forms is also not included.

3.1.1.3.1 Include

Body:

EMPTY

Attributes:

StudyOID	oidref	References the Study that provides a prior metadata version.
MetaDataVersionOID	oidref	References the prior MetaDataVersion (within the above Study).

Contained in:

MetaDataVersion

A reference to a prior metadata version. This version must be present earlier in the same ODM file or in a previous file in the series. See the PriorFileOID attribute of the ODM element.

When a metadata element such as an ItemDef is re-defined in a MetaDataVersion that includes a previous MetaDataVersion, any metadata element of the same type and with the same OID in a preceding MetaDataVersion is completely replaced. This includes all attributes and child elements. It also means that all attributes and child elements not present in the new definition are effectively deleted from the previous version.

In the following example, question "I.003" has been added to the ItemGroup "IG.001" by adding an ItemRef element to the ItemGroupDef. The new version of the ItemGroupDef now contains only one of the Aliases, and the other Alias defined in MetaDataVersion "MDV.001" has been removed. This same mechanism is used to remove ItemRefs from an ItemGroupDef, or to re-order them.

```
<Study OID="S.001">
  <MetaDataVersion OID="MDV.001" Name="First Metadata version">
    <ItemGroupDef OID="IG.001" Name="First ItemGroup">
      <ItemRef ItemOID="I.001" Mandatory="Yes" OrderNumber="1"/>
      <ItemRef ItemOID="I.002" Mandatory="Yes" OrderNumber="2"/>
      <Alias Context="Context1" Name="IG1"/>
      <Alias Context="Context2" Name="FIRST"/>
    </ItemGroupDef>
  </MetaDataVersion>

  <MetaDataVersion OID="MDV.002" Name="Second Metadata version">
    <Include StudyOID="S.001" MetaDataVersionOID="MDV.001"/>
    <ItemGroupDef OID="IG.001" Name="First ItemGroup (modified)">
      <ItemRef ItemOID="I.001" Mandatory="Yes" OrderNumber="1"/>
      <ItemRef ItemOID="I.003" Mandatory="Yes" OrderNumber="2"/>
      <ItemRef ItemOID="I.002" Mandatory="Yes" OrderNumber="3"/>
      <Alias Context="Context1" Name="IG1"/>
    </ItemGroupDef>
```

```
  </MetaDataVersion>
</Study>
```

Note: Definitions (e.g. ItemDef) that have been defined in a previous included MetaDataVersion cannot be removed in later MetaDataVersions, only references to those Items can be removed. Referenced Items are removed by redefining the parent definition (e.g. ItemGroupDef) and omitting the undesired reference.

Note: The StudyOID attribute allows an Include element to reference a metadata version in another study. Thus, it is possible for many studies to share a set of common metadata definitions. In fact, a study that contains nothing but metadata can serve as a metadata library.

Note: By placing this information in a subelement of MetaDataVersion (rather than in attributes) we allow for future evolution of this capability. For example, it might be useful to allow inclusion of two common metadata dictionaries, or to add new common metadata into a previous study-specific metadata version.

3.1.1.3.2 Protocol

Body:
(**Description**?, **StudyEventRef***, **Alias***)

Attributes:
NONE

Contained in:
MetaDataVersion

The Protocol lists the kinds of study events that can occur within a specific version of a Study. All clinical data must occur within one of these study events.

Note: A study whose metadata does not contain a protocol definition cannot have any clinical data. Such studies can serve as "common metadata dictionaries" -- allowing sharing of metadata across studies.

3.1.1.3.2.1 Description

Body:
(**TranslatedText**+)

Attributes:
NONE

Contained in:
Protocol, StudyEventDef, FormDef, ItemGroupDef, ItemDef, ConditionDef, MethodDef

A free-text description of the containing study metadata component.

3.1.1.3.2.2 StudyEventRef

Body:
EMPTY

Attributes:

StudyEventOID	oidref		Reference to the StudyEventDef.
OrderNumber	integer	*(optional)*	
Mandatory	(Yes | No)		
CollectionExceptionConditionOID	oidref	*(optional)*	Reference to a ConditionDef

Contained in:
Protocol

A reference to a StudyEventDef as it occurs within a specific version of a Study. The list of StudyEventRefs identifies the types of study events that are allowed to occur within the study. The StudyEventRefs within a Protocol must not have duplicate StudyEventOIDs nor duplicate OrderNumbers.

The OrderNumbers provide an ordering on the StudyEventDefs for use whenever a list of StudyEventDefs is presented to a user. They do not imply anything about event scheduling, time ordering, or data correctness.

The Mandatory flag indicates that the clinical data for the containing MetaDataVersion would be incomplete without an instance of this type of Study Event. ODM clinical data files that are incomplete in this sense may be considered incomplete for study review and analysis purposes.

If the CollectionExceptionConditionOID attribute is provided, it references a ConditionDef that describes the circumstances under which data for this Study Event should not be collected.

3.1.1.3.3 StudyEventDef

Body:
(**Description**?, **FormRef***, **Alias***)

Attributes:

OID	oid	
Name	name	
Repeating	(Yes | No)	
Type	(Scheduled | Unscheduled | Common)	
Category	text	*(optional)*

Contained in:
MetaDataVersion

A StudyEventDef packages a set of forms. Scheduled Study Events correspond to sets of forms that are expected to be collected for each subject as part of the planned visit sequence for the study. Unscheduled Study Events are designed to collect data that may or may not occur for any particular subject such as a set of forms that are completed for an early termination due to a serious adverse event. A common Study Event is a collection of forms that are used at several different data collection events such as an Adverse Event or Concomitant Medications log.

The Repeating flag indicates that this type of study event can occur repeatedly within any given subject.

The Category attribute is typically used to indicate the study phase appropriate to this type of study event. Examples might include Screening, PreTreatment, Treatment, and FollowUp.

3.1.1.3.3.1 FormRef

Body:
EMPTY

Attributes:

FormOID	oidref		Reference to the FormDef.
OrderNumber	integer	*(optional)*	
Mandatory	(Yes | No)		
CollectionExceptionConditionOID	oidref	*(optional)*	Reference to a ConditionDef

Contained in:
StudyEventDef

A reference to a FormDef as it occurs within a specific StudyEventDef. The list of FormRefs identifies the types of forms that are allowed to occur within this type of study event. The FormRefs within a single StudyEventDef must not have duplicate FormOIDs nor OrderNumbers.

The OrderNumbers provide an ordering on the Forms (within a containing Study Event) for use whenever a list of Forms is presented to a user. They do not imply anything about event scheduling, time ordering, or data correctness.

The Mandatory flag indicates that the clinical data for an instance of the containing study event would be incomplete without an instance of this type of form. ODM clinical data files that are incomplete in this sense may be considered incomplete for study review and analysis purposes.

If the CollectionExceptionConditionOID attribute is provided, it references a ConditionDef that describes the circumstances under which data for this Form should not be collected.

3.1.1.3.4 FormDef

Body:
(**Description**?, **ItemGroupRef***, **ArchiveLayout***, **Alias***)

Attributes:

OID	oid
Name	name
Repeating	(Yes | No)

Contained in:
MetaDataVersion

A FormDef describes a type of form that can occur in a study.

The Repeating flag indicates that this type of form can occur repeatedly within a containing study event.

3.1.1.3.4.1 ItemGroupRef

Body:
EMPTY

Attributes:

ItemGroupOID	oidref		Reference to the ItemGroupDef.
OrderNumber	integer	*(optional)*	
Mandatory	(Yes | No)		
CollectionExceptionConditionOID	oidref	*(optional)*	Reference to a ConditionDef

Contained in:
FormDef

A reference to an ItemGroupDef as it occurs within a specific FormDef. The list of ItemGroupRefs identifies the types of item groups that are allowed to occur within this type of form. The ItemGroupRefs within a single FormDef must not have duplicate ItemGroupOIDs nor OrderNumbers.

The OrderNumbers provide an ordering on the ItemGroupDefs (within the containing FormDef) for use whenever a list of ItemGroupDefs is presented to a user. They do not imply anything about event scheduling, time ordering, or data correctness.

The Mandatory flag indicates that the clinical data for an instance of the containing form would be incomplete without an instance of this type of item group. ODM clinical data files that are incomplete in this sense may be considered incomplete for study review and analysis purposes.

If the CollectionExceptionConditionOID attribute is provided, it references a ConditionDef that describes the circumstances under which data for this ItemGroup should not be collected.

3.1.1.3.5 ItemGroupDef

Body:
(**Description**?, **ItemRef***, **Alias***)

Attributes:

OID	oid		
Name	name		
Repeating	(Yes | No)		
IsReferenceData	(Yes | No)	*(optional)*	
SASDatasetName	sasName	*(optional)*	
Domain	text	*(optional)*	
Origin	text	*(optional)*	
Role	name	*(optional)*	*Deprecated*
Purpose	text	*(optional)*	
Comment	text	*(optional)*	

Contained in:
MetaDataVersion

An ItemGroupDef describes a type of item group that can occur within a study.

The Repeating flag indicates that this type of item group can occur repeatedly within the containing form (or reference data).

If IsReferenceData is Yes, this type of item group can occur only within a ReferenceData element. If IsReferenceData is No, this type of item group can occur only within a ClinicalData element. The default for this attribute is No.

The Domain, Origin, Purpose, and Comment attributes carry submission information as described in the CDISC Submission Metadata Model located in the SDTM Metadata Submission Guidelines.

Note: The Role attribute can be considered a synonym for Purpose. New applications should use Purpose rather than Role.

3.1.1.3.5.1 ItemRef

Body:
EMPTY

Attributes:

ItemOID	oidref		Reference to the ItemDef.
OrderNumber	integer	*(optional)*	
Mandatory	(Yes | No)		
KeySequence	integer	*(optional)*	
MethodOID	oidref	*(optional)*	
ImputationMethodOID	*Deprecated*		
Role	text	*(optional)*	
RoleCodeListOID	oidref	*(optional)*	
CollectionExceptionConditionOID	oidref	*(optional)*	Reference to a ConditionDef

Contained in:

ItemGroupDef

A reference to an ItemDef as it occurs within a specific ItemGroupDef. The list of ItemRefs identifies the types of items that are allowed to occur within this type of item group.

The ItemRefs within a single ItemGroupDef must not have duplicate ItemOIDs nor OrderNumbers.

The OrderNumbers provide an ordering on the Items (within a containing Item Group) for use whenever a list of Items is presented to a user. They do not imply anything about event scheduling, time ordering, or data correctness.

The Mandatory flag indicates that the clinical data for an instance of the containing item group would be incomplete without an instance of this type of item. ODM clinical data files that are incomplete in this sense may be considered incomplete for study review and analysis purposes.

The KeySequence (if present) indicates that this item is a key for the enclosing item group. It also provides an ordering for the keys.

The MethodOID references a MethodDef used to derive the value of this item.

The Role attribute provides a single role name describing the use of this data item.

If the Role is defined by a standard terminology, RoleCodeListOID may be used to reference a CodeList that defines the full set roles from which the Role attribute value is to be taken. This attribute should not be present unless the Role attribute is defined. If Role is defined, RoleCodeListOID is still optional.

If the CollectionExceptionConditionOID is provided, it references a ConditionDef that defines the circumstances under which collection of the Item may be omitted, i.e. when the FormalExpression evaluates to True. For example, a data collection instrument designed to collect data for an item 'IsPregnant' for an item group 'InclusionCriteria' may be omitted for male subjects. The referenced ConditionDef would be defined as illustrated below. When the FormalExpression evaluates to True, the Item with OID IG.ISPREG need not be collected.

```
<ItemGroupDef OID="INCLUSION" Name="Inclusion Criteria" >
                  ...
                  <ItemRef ItemOID="IDef.GENDER" Mandatory="Yes"/>
                  <ItemRef ItemOID="IDef.ISPREG" Mandatory="Yes" CollectionExceptionConditionOID="CECID.ISMALE"/>
</ItemGroupDef>

<ConditionDef OID="CECID.ISMALE">
     <TranslatedText xml:lang="en">Do not collect data for Male subjects </TranslatedText>
         <FormalExpression context="PL/SQL">
                  INCLUSION.IDef.GENDER = 'Male'
         </FormalExpression>
</ConditionDef>
```

3.1.1.3.6 ItemDef

Body:

(**Description**?, **Question**?, **ExternalQuestion**?, **MeasurementUnitRef***, **RangeCheck***, **CodeListRef**?, **Role*** ***Deprecated***, **Alias***)

Attributes:

OID	oid	
Name	name	
DataType	(text | integer | float | date | time | datetime | string | boolean | double | hexBinary | base64Binary | hexFloat | base64Float | partialDate | partialTime | partialDatetime | durationDatetime | intervalDatetime | incompleteDatetime | incompleteDate | incompleteTime | URI)	
Length	positiveInteger	*(optional)*
SignificantDigits	nonNegativeInteger	*(optional)*
SASFieldName	sasName	*(optional)*

SDSVarName	sasName	*(optional)*
Origin	text	*(optional)*
Comment	text	*(optional)*

Contained in:

MetaDataVersion

An ItemDef describes a type of item that can occur within a study. Item properties include name, datatype, measurement units, range or codelist restrictions, and several other properties.

The DataType attribute specifies how the corresponding value elements are to be interpreted for comparison and storage. The Length attribute is required when DataType is text or string, optional when DataType is integer or float, and should not be given for the other datatypes. The SignificantDigits attribute is optional when DataType is float, and should not be given for the other datatypes. When DataType is float both Length and SignificantDigits must be given or both be absent.

Note: In version 1.3.0 the usage of Length and SignificantDigits was ambiguous. The above text clarifies the intention of these two attributes.

If DataType=integer, Length=N is a requirement that the receiving system be able to process and store all whole number values of magnitude less than 10^N. Larger values may be rejected.

If DataType=float, Length=N and SignificantDigits=S is a requirement that the receiving system be able to process and store all numeric values of magnitude less than 10^{N-S} that are multiples of 10^{-S}. Larger values may be rejected. Intermediate values may be rounded to the nearest multiple of 10^{-S}.

If DataType=text, Length=N is a requirement that the receiving system be able to process and store all text string values of length less than or equal to N. All characters are allowed in text string values. Data of type Text should be transmitted in an ItemDataString element.

Note: Length and SignificantDigits are statements about an item's data values, not the number of characters used to represent these values in value elements. For example, the character "<" might be represented as "<".

Note: Data characters that are not included in the encoding character set for a particular ODM file must be represented using XML entities or character references. For example, Æ could be represented as "Æ".

The SDSVarName, Origin, and Comment attributes carry submission information as described in the latest version of CDISC SDTM .

> **Note:** In the ODM model, all internal keys are assumed to be unchangeable. This was done to make the audit trail issues work: if the SubjectKey in the model were the actual external subject identifier (or randomization ID) of a patient, and that value is sent incorrectly in one ODM file, there would be no way to correct the mistake in a followup file. In doing this, we intend that the external subject keys (and other externally visible key variables) should be defined as Items in the metadata. Thus they can be modified through normal modify/audit mechanism. While this solves the problem of supporting modification of study keys, it leaves the user without a way to identify which ItemDefs have special meaning or what the meaning is. The most obvious place where this is a problem is in matching up patients when loading data from an external source. If you can't find the patient ID how do you do the matching?
>
> The answer is to use the SDSVarName attribute of ItemDef. SDSVarName is an optional attribute which can be used to tag the Item with a business meaning. Rather than try to enumerate all possible meanings in the ODM model, the ODM working group thought it best to rely on the set of variable names defined in the CDISC SDTM, since this list covers the core variables used in managing clinical data. Software that is processing an ODM-compliant XML instance can therefore use specific values of the SDSVarName attribute to identify standard, frequently used variables. The use of this attribute is restricted to variables defined in the SDTM model. In tagging a variable, you are identifying it as matching the SDTM definition for that variable. A partial list of commonly used values includes:
>
> - STUDYID (Study Identifier Unique within a Submission)
> - USUBJID (Study Identifier Unique within a Submission),
> - SUBJID (Subject Identifier Unique within a Study),
> - SITEID (Unique Identifier for a study Site)
> - SEX (Sex or Gender, coded value),
> - VISITNUM (Clinical encounter Number)
> - VISIT (Protocol-defined description of clinical encounter),
> - VISITDY (Planned study day of VISIT)
>
> See the SDTM Specification and Implementation Guide for more information about SDTM variables.

The Question element contains the text shown to a human user when prompted to provide data for this Item. The ExternalQuestion element does the same but refers to an externally defined question. If both are present, they should be consistent.

The MeasurementUnitRefs list the acceptable measurement units for this type of item. Only numeric items can have measurement units. If only one MeasurementUnitRef is present, all items of this type carry this measurement unit by default, i.e. if a MeasurementUnitRef is defined on the ItemDef, and no MeasurementUnitRef is given on the corresponding ItemData, the value given by the ItemData has the units given by the ItemDef-MeasurementUnitRef.
If no MeasurementUnitRef is present on the definition of a numeric Item, the Item's value is scalar (i.e., a pure number).

The RangeChecks constrain the acceptable values for items of this type.

The CodeListRef (if present) constrains the acceptable values for items of this type to be members of the referenced codelist.

Note: Items do not repeat within an item group.

3.1.1.3.6.1 Question

Body:
(**TranslatedText**+)

Attributes:
NONE

Contained in:
ItemDef

A label shown to a human user when prompted to provide data for an item on paper or on a screen.

3.1.1.3.6.2 ExternalQuestion

Body:
EMPTY

Attributes:

Dictionary	text	*(optional)*	The name of the external question dictionary.
Version	text	*(optional)*	The version designator of the external question dictionary.
Code	text	*(optional)*	A code selecting the particular question within the external dictionary.

Contained in:
ItemDef

A reference to an externally defined question.

3.1.1.3.6.3 MeasurementUnitRef

Body:
EMPTY

Attributes:

MeasurementUnitOID	oidref	Reference to the MeasurementUnit definition.

Contained in:
ItemData, ItemDef, RangeCheck

A reference to a measurement unit definition.

3.1.1.3.6.4 RangeCheck

Body:
((**CheckValue**+ | **FormalExpression**+), **MeasurementUnitRef**?, **ErrorMessage**?)

Attributes:

Comparator	(LT | LE | GT | GE | EQ | NE | IN | NOTIN)	(optional)	Comparison operator used to compare the item and value(s).	(optional)
SoftHard	(Soft | Hard)			

Contained in:
ItemDef

A RangeCheck defines a constraint on the value of the enclosing item. It represents an expression that evaluates to True when the ItemData value is valid or False when the ItemData value is invalid. The expression is specified using either Comparator and CheckValue or using FormalExpressions.

Range Checks using Comparator and CheckValue
When using Comparator and Check Value each Range Check represents a one-sided constraint. Multiple Range Checks can be used to specify more complex constraints, e.g., an upper and lower bound would require two Range Checks.

Each constraint is equivalent to:

itemValue **comparator** checkValue(s)

If an actual data value fails the constraint, it is either rejected (a Hard constraint) or a warning is produced (a Soft constraint).

For the following comparison operators, one Check Value is required.

LT	Less than
LE	Less than or equal to
GT	Greater than
GE	Greater than or equal to
EQ	Equal to
NE	Not equal to

A set of Check Values is required for these comparators:

IN	One of listed values
NOTIN	Not any of list values

If a Measurement Unit is specified, the corresponding Item values must have interconvertible Measurement Units (either explicitly or by default). Proper conversion of units must be done as part of the Range Check. If a Measurement Unit is not specified, the corresponding Item values must not have Measurement Units (either explicitly or by default).

Examples:

Value must be positive

```
<RangeCheck Comparator="GT">
    <CheckValue>0</CheckValue>
</RangeCheck>
```

Value must be between 18 and 65 inclusive

```
<RangeCheck Comparator="GE">
    <CheckValue>18</CheckValue>
</RangeCheck>
<RangeCheck Comparator="LE">
    <CheckValue>65</CheckValue>
</RangeCheck>
```

Value must be one of (1, 5, 7)

```
<RangeCheck Comparator="IN">
    <CheckValue>1</CheckValue>
    <CheckValue>3</CheckValue>
    <CheckValue>5</CheckValue>
</RangeCheck>
```

Range Checks using FormalExpression
When using FormalExpression a Range Check can represent anything, e.g., one sided or multi-sided checks. These type of checks must not provide CheckValue, Comparator or MeasurementUnitRef as they would all be expressed in the FormalExpression itself. The FormalExpression takes the value of the ItemData element and returns a boolean value which is the result of the expression. Multiple FormalExpressions can be provided if each has a different Context attribute, allowing the same expression to be represented in forms appropriate to multiple systems. Multiple different expressions, with different meanings, must be represented as separate RangeChecks

3.1.1.3.6.4.1 CheckValue

Body:

value

Attributes:

NONE

Contained in:

RangeCheck

A comparison value used in a range check. It must match the value type declared in the containing ItemDef.

3.1.1.3.6.4.2 ErrorMessage

Body:

(**TranslatedText**+)

Attributes:

NONE

Contained in:

RangeCheck

Error message to be used when a range check is violated.

3.1.1.3.6.5 CodeListRef

Body:
EMPTY

Attributes:

CodeListOID	oidref	Reference to the CodeList definition.

Contained in:
ItemDef

A reference to a CodeList definition. The DataType attributes of the referenced CodeList and the containing ItemDef must be the same.

3.1.1.3.6.6 Alias

Body:
EMPTY

Attributes:

Context	text
Name	text

Contained in:
Protocol, StudyEventDef, FormDef, ItemGroupDef, ItemDef, CodeList, CodeListItem, EnumeratedItem, MethodDef, ConditionDef

An Alias provides an additional name for an element. The Context attribute specifies the application domain in which this additional name is relevant.

3.1.1.3.7 CodeList

Body:
(**Description**?, (**CodeListItem**+ | **EnumeratedItem**+ | **ExternalCodeList**), **Alias***)

Attributes:

OID	oid	
Name	name	
DataType	(integer | float | text | string)	
SASFormatName	sasFormat	*(optional)*

Contained in:
MetaDataVersion

Defines a discrete set of permitted values for an item. The definition can be an explicit list of values (CodeListItem+ | EnumeratedItem+) or a reference to an externally defined codelist (ExternalCodeList).

The DataType restricts the values that can appear in the CodeList whether internal or external.

The SASFormatName must be a legal SAS format.

3.1.1.3.7.1 CodeListItem

Body:
(**Decode**, **Alias***)

Attributes:

CodedValue	text		Value of the codelist item (as it would occur in clinical data).
Rank	float	*(optional)*	Numeric significance of the CodeListItem relative to others in the CodeList. Rank is optional, but if given for any CodeListItems in a CodeList it must be given for all.
OrderNumber	integer	*(optional)*	Ordering on the Items (within a containing CodeListItem) for use whenever a list of Items is presented to a user. OrderNumber is optional, but if given for any CodeListItems in a CodeList it must be given for all.

Contained in:
CodeList

Defines an individual member value of a codelist including display format. The actual value is given, along with a set of print/display-forms. The CodedValue must be an acceptable value of the DataType of the containing CodeList.

The CodeListItems within a single CodeList must not have duplicate CodedValues (as interpreted by the CodeList's DataType).

The Rank attribute may be used where the relative value corresponding to an enumeration cannot or should not be determined by its lexical order. For example, if you have a list of enumerated text values including "Low", "Medium", and "High" and wish to assign these relative numeric values 1, 2, and 3 respectively, you should include a Rank attribute for each CodeListItem defined. Without the applied rank attribute, the normal lexical ordering would be "High", "Low", and "Medium".

The OrderNumbers provide an ordering on the CodeListItems (within a containing CodeList) for use whenever a list of CodeList Items is presented to a user. They do not imply anything about event scheduling, time ordering, or data correctness.

The CodeListItems within a single CodeList must not have duplicate Ranks or OrderNumbers.

3.1.1.3.7.1.1 Decode

Body:
(**TranslatedText**+)

Attributes:
NONE

Contained in:
CodeListItem

The displayed value relating to the CodedValue.

3.1.1.3.7.2 ExternalCodeList

Body:
EMPTY

Attributes:

Dictionary	text	*(optional)*	The name of the external codelist.
Version	text	*(optional)*	The version designator of the external codelist.
ref	text	*(optional)*	Reference to a local instance of the dictionary.
href	text	*(optional)*	URL of an external instance of the dictionary.

Contained in:
CodeList

A reference to an externally defined codelist.

3.1.1.3.7.3 EnumeratedItem

Body:
(**Alias***)

Attributes:

CodedValue	text		Value of the enumerated item (as it would occur in clinical data).
Rank	float	*(optional)*	Numeric significance of the CodeListItem relative to others in the CodeList. Rank is optional, but if given for any EnumeratedItems in a CodeList it must be given for all.
OrderNumber	integer	*(optional)*	Ordering on the Items (within a containing EnumeratedItem) for use whenever a list of Items is presented to a user. OrderNumber is optional, but if given for any EnumeratedItems in a CodeList it must be given for all.

Contained in:
CodeList

Defines an individual member value of a valid value list. Only the coded value is given. If print/display-forms are needed, CodeListItem should be used. CodeListItems and EnumeratedItems may not be mixed within a single codelist. The CodedValue must be an acceptable value of the DataType of the containing CodeList.

The EnumeratedItems within a single CodeList element must not have duplicate CodedValues (as interpreted by the CodeList's DataType).

The Rank attribute may be used where the relative value corresponding to an enumeration cannot or should not be determined by its lexical order. For example, if you have a list of enumerated text values including "Low", "Medium", and "High" and wish to assign these relative numeric values 1, 2, and 3 respectively, you should include a Rank attribute for each EnumeratedItem defined. Without the applied rank attribute, the normal lexical ordering would be "High", "Low", and "Medium".

The OrderNumbers provide an ordering on the EnumeratedItems (within a containing CodeList) for use whenever a list of Enumerated Items is presented to a user. They do not imply anything about event scheduling, time ordering, or data correctness.

The EnumeratedItems within a single CodeList must not have duplicate Ranks or OrderNumbers.

3.1.1.3.8 ArchiveLayout

Body:
EMPTY

Attributes:

OID	oid		
PdfFileName	fileName		Name of a Adobe PDF file containing the screen layout.
PresentationOID	oidref	*(optional)*	Reference to a Presentation definition.

Contained in:
FormDef

A visual image of the screen layout used to collect the data on a form. Useful in archiving a study.

3.1.1.3.9 MethodDef

Body:
(**Description**, **FormalExpression***, **Alias***)

Attributes:

OID	oid
Name	name
Type	(Computation | Imputation | Transpose | Other)

Contained in:
MetaDataVersion

A MethodDef defines how a data value can be obtained from a collection of other data values. The Description element must be provided and should include a prose description. This is the normative content of the MethodDef.

If a FormalExpression is provided, it must contain a machine-readable expression that implements the Description and will return a value. Multiple FormalExpressions can be provided if each has a different Context attribute, allowing the same expression to be represented in forms appropriate to multiple systems.

3.1.1.3.10 Presentation

Body:
text

Attributes:

OID	oid	
xml:lang	languageTag	*(optional)*

Contained in:
MetaDataVersion

A Presentation defines how information about the study is presented to the user of a system. The xml:lang attribute helps select the appropriate presentation. See TranslatedText.

Note: The Presentation element is not defined further in CDISC ODM V1.3. It is a placeholder for future development.

3.1.1.3.11 ConditionDef

Body:
(**Description**, **FormalExpression***, **Alias***)

Attributes:

OID	oid
Name	name

Contained in:
MetaDataVersion

A ConditionDef defines a boolean condition. The Description element must be provided and should include a prose description. This is the normative content of the ConditionDef. The ConditionDef is referenced by the CollectionExceptionConditionOID attribute within a study metadata component which may be omitted under circumstances defined by the condition, i.e., when the FormalExpression evaluates to True.

If a FormalExpression is provided, it must contain a machine-readable expression that will evaluate to True or False. Multiple FormalExpressions can be provided if each has a different Context attribute, allowing the same expression to be represented in forms appropriate to multiple systems.

If an application cannot interpret any of the FormalExpressions or does not normally support conditional data collection, data for the referencing study metadata component should be collected as though no Condition is specified.

3.1.1.3.11.1 FormalExpression

Body:
(PCDATA)

Attributes:

Context	text	A free-form qualifier to suggest an appropriate computer language to be used when evaluating the FormalExpression content.

Contained in:
ConditionDef, MethodDef, RangeCheck

A FormalExpression used within a ConditionDef or a RangeCheck must evaluate to True or False. A FormalExpression referenced within a MethodDef having Type Imputation, Computation or Transpose must evaluate to the correct DataType for an Item that may be imputed or computed using the Method.

3.1.2 Administrative Elements

3.1.2 AdminData

Body:
(**User***, **Location***, **SignatureDef***)

Attributes:

StudyOID	oidref	*(optional)*	Reference to a StudyDef.

Contained in:
ODM

Administrative information about users, locations, and electronic signatures.

If StudyOID is not provided, it is assumed that AdminData definitions apply to all studies defined within the current ODM file and/or to all studies within a sequence of ODM files linked through the PriorFileOID mechanism.

3.1.2.1 User

Body:
(**LoginName**?, **DisplayName**?, **FullName**?, **FirstName**?, **LastName**?, **Organization**?, **Address***, **Email***, **Picture**?, **Pager**?, **Fax***, **Phone***, **LocationRef***, **Certificate***)

Attributes:

OID	oid	
UserType	(Sponsor | Investigator | Lab | Other)	*(optional)*

Contained in:
AdminData

Information about a specific user of a clinical data collection system. This may be an investigator, a CRA, or data management staff. Study subjects are *not* users in this sense.

3.1.2.1.1 LoginName

Body:
text

Attributes:
NONE

Contained in:
User

The user's login identification.

3.1.2.1.2 DisplayName

Body:
text

Attributes:
NONE

Contained in:
User

A short displayable name for the user.

3.1.2.1.3 FullName

Body:
text

Attributes:
NONE

Contained in:
User

The user's full formal name.

3.1.2.1.4 FirstName

Body:
text

Attributes:
NONE

Contained in:
User

The user's initial given name or all given names.

3.1.2.1.5 LastName

Body:
text

Attributes:
NONE

Contained in:
User

The user's surname (family name).

3.1.2.1.6 Organization

Body:
text

Attributes:
NONE

Contained in:
User

The user's organization.

3.1.2.1.7 Address

Body:
(**StreetName***, **City**?, **StateProv**?, **Country**?, **PostalCode**?, **OtherText**?)

Attributes:
NONE

Contained in:
User

The user's postal address.

3.1.2.1.8 StreetName

Body:
text

Attributes:
NONE

Contained in:
Address

The street address part of a user's postal address.

3.1.2.1.9 City

Body:
text

Attributes:
NONE

Contained in:
Address

The city name part of a user's postal address.

3.1.2.1.10 StateProv

Body:
text

Attributes:
NONE

Contained in:
Address

The state or province name part of a user's postal address.

3.1.2.1.11 Country

Body:
text

Attributes:
NONE

Contained in:
Address

The country name part of a user's postal address. This must be represented by an ISO 3166 two-letter country code.

Example: FR for France, JP for Japan.

3.1.2.1.12 PostalCode

Body:
text

Attributes:
NONE

Contained in:
Address

The postal code part of a user's postal address.

3.1.2.1.13 OtherText

Body:
text

Attributes:
NONE

Contained in:
Address

Any other text needed as part of a user's postal address.

3.1.2.1.14 Email

Body:
text

Attributes:
NONE

Contained in:
User

The user's email address.

3.1.2.1.15 Picture

Body:
EMPTY

Attributes:

PictureFileName	fileName	
ImageType	name	*(optional)*

Contained in:
User

A visual depiction of the user.

3.1.2.1.16 Pager

Body:
text

Attributes:
NONE

Contained in:
User

The phone number of the user's pager.

3.1.2.1.17 Fax

Body:
text

Attributes:
NONE

Contained in:
User

The phone number of the user's facsimile machine.

3.1.2.1.18 Phone

Body:
text

Attributes:
NONE

Contained in:
User

The user's voice phone number.

3.1.2.1.19 LocationRef

Body:
EMPTY

Attributes:
LocationOID oidref Reference to a Location definition.

Contained in:
AuditRecord, Signature, User

A reference to the user's physical location.

3.1.2.1.20 Certificate

Body:
text

Attributes:
NONE

Contained in:
User

The user's digital signing certificate.

Note: The Certificate element is not defined further in CDISC ODM V1.3. It is a placeholder for future development.

3.1.2.2 Location

Body:
(**MetaDataVersionRef**+)

Attributes:
OID oid
Name name
LocationType (Sponsor | Site | CRO | Lab | Other) *(optional)*

Contained in:
AdminData

A physical location -- typically a clinical research site or a sponsor's office.

3.1.2.2.1 MetaDataVersionRef

Body:
EMPTY

Attributes:

StudyOID	oidref	References the Study that uses this metadata version.
MetaDataVersionOID	oidref	References the MetaDataVersion (within the above Study).
EffectiveDate	date	

Contained in:
Location

A reference to a MetaDataVersion used at the containing Location. The EffectiveDate expresses the fact that the metadata used at a location can vary over time.

3.1.2.3 SignatureDef

Body:
(**Meaning**, **LegalReason**)

Attributes:

OID	oid	
Methodology	(Digital | Electronic)	*(optional)*

Contained in:
AdminData

Defines a kind of electronic signature, including the *meaning* as required by 21 CFR Part 11. If the signature is Digital, it is based on cryptography. Otherwise the signature is Electronic.

Note: Tracking of data edits is done using AuditRecords not signatures.

3.1.2.3.1 Meaning

Body:
text

Attributes:
NONE

Contained in:
SignatureDef

A short name for this kind of signature (e.g., authorship, approval).

3.1.2.3.2 LegalReason

Body:
text

Attributes:
NONE

Contained in:
SignatureDef

The responsibility statement associated with a signature (e.g., "The signer accepts responsibility for the accuracy of this data.").

3.1.3 Reference Data Elements

3.1.3 ReferenceData

Body:
(**ItemGroupData***, **AuditRecords***, **Signatures***, **Annotations***)

Attributes:

StudyOID	oidref	References the Study that defines the metadata for this reference data.
MetaDataVersionOID	oidref	References the MetaDataVersion (within the above Study) for this reference data.

Contained in:

ODM

Reference data provides information on how to interpret clinical data. For example, reference data might include lab normal ranges. Signature elements nested within ReferenceData have no meaning, and should be ignored.

The StudyOID and MetaDataVersionOID attributes select a particular metadata version. All metadata references (OIDs) occurring within this ReferenceData element refer to definitions within the selected metadata version.

The TransactionType attribute behaves the same within ReferenceData as it does within ClinicalData.

Note: Since reference data can be independent of any particular study, it may be desirable to keep the reference metadata separate from clinical metadata. This can be done by creating a Study element with no Protocol, StudyEventDef, or FormDef elements. All the ItemGroupDefs would have IsReferenceData=Yes. Such a study would have no clinical data.

3.1.4 Clinical Data Elements

3.1.4 ClinicalData

Body:

(**SubjectData***, **AuditRecords***, **Signatures***, **Annotations***)

Attributes:

StudyOID	oidref	References the Study that uses the data nested within this element.
MetaDataVersionOID	oidref	References the MetaDataVersion (within the above Study) that governs the data nested within this element.

Contained in:

ODM

Clinical data for multiple subjects.

The StudyOID and MetaDataVersionOID attributes select a particular metadata version. All metadata references (OIDs) occurring within this ClinicalData element refer to definitions within the selected metadata version.

3.1.4.1 SubjectData

Body:

(**AuditRecord**?, **Signature**?, **InvestigatorRef**?, **SiteRef**?, **Annotation***, **StudyEventData***)

Attributes:

SubjectKey	subjectKey		
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	The TransactionType attribute need not be present in a Snapshot document.

Contained in:

ClinicalData

Clinical data for a single subject.

The SubjectKey is used to identify a specific subject. This key uniquely identifies a subject within the study specified by the parent ClinicalData element. If the same SubjectKey is used in multiple ClinicalData elements in a single study, this is interpreted as being the same subject.

Example:

```
<ClinicalData StudyOID="P2006-101" MetadataVersionOID="101.01">
   <SubjectData SubjectKey="1000" TransactionType="Insert">
      <StudyEventData StudyEventOID="Screen">
         <FormData FormOID="DEMOG">
            <ItemGroupData ItemGroupOID="DM">
                <ItemDataString ItemOID="USUBJID">101-001-001</ItemDataString>
                <ItemDataString ItemOID="SEX">F</ItemDataString>
            </ItemGroupData>
        </FormData>
        <FormData FormOID="LABDATA">
            <ItemGroupData ItemGroupOID="LB">
                <ItemDataDatetime ItemOID="LBDTC">2006-07-14T14:48</ItemDataDatetime>
                <ItemDataString ItemOID="LBTESTCD">ALT</ItemDataString>
                <ItemDataString ItemOID="LBORRES">245</ItemDataString>
            </ItemGroupData>
        </FormData>
      </StudyEventData>
```

```
  </SubjectData>
</ClinicalData>
<ClinicalData StudyOID="P2006-101" MetadataVersionOID="101.02">
  <SubjectData SubjectKey="1000" TransactionType="Insert">
      <StudyEventData>
        <FormData FormOID="AENONSER">
            <ItemGroupData ItemGroupOID="AE">
                <ItemDataString ItemOID="AETERM">Fever</ItemDataString>
                <ItemDataDate ItemOID="AESTDTC">2006-08-21</ItemDataDate>
            </ItemGroupData>
        </FormData>
        <FormData FormOID="LABDATA">
            <ItemGroupData ItemGroupOID="LB">
                <ItemDataDatetime ItemOID="LBDTC">2006-07-14T14:48</ItemDataDatetime>
                <ItemDataString ItemOID="LBTESTCD">ALT</ItemDataString>
                <ItemDataString ItemOID="LBORRES">300</ItemDataString>
            </ItemGroupData>
        </FormData>
      </StudyEventData>
  </SubjectData>
</ClinicalData>
```

3.1.4.1.1 StudyEventData

Body:

(**AuditRecord**?, **Signature**?, **Annotation***, **FormData***)

Attributes:

StudyEventOID	oidref		Reference to the StudyEventDef.
StudyEventRepeatKey	repeatKey	*(optional)*	A key used to distinguish between repeats of the same type of study event for a single subject.
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	The TransactionType attribute need not be present in a Snapshot document.

Contained in:

SubjectData

Clinical data for a study event (visit). The model supports repeating study events (for example, when the same set of information is collected for a series of patient visits).

The StudyEventOID and StudyEventRepeatKey are used together to identify a particular study event. This pair of values uniquely identifies a StudyEvent within the containing subject. The StudyEventRepeatKey is present if and only if the StudyEventDef is repeating.

3.1.4.1.1.1 FormData

Body:

(**AuditRecord**?, **Signature**?, **ArchiveLayoutRef**?, **Annotation***, **ItemGroupData***)

Attributes:

FormOID	oidref		Reference to the FormDef.
FormRepeatKey	repeatKey	*(optional)*	A key used to distinguish between repeats of the same type of form within a single study event.
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	The TransactionType attribute need not be present in a Snapshot document.

Contained in:

StudyEventData

Clinical data for a form (page). The model supports repeating forms in a single study event (for example, when several adverse events are recorded in a single patient visit).

The FormOID and FormRepeatKey are used together to identify a particular form. This pair of values uniquely identifies a Form within the containing StudyEvent. The FormRepeatKey is present if and only if the FormDef is repeating.

3.1.4.1.1.1.1 ItemGroupData

Body:

(**AuditRecord**?, **Signature**?, **Annotation***, (**ItemData*** | **ItemData*[TYPE]***))

Attributes:

ItemGroupOID	oidref		Reference to the ItemGroupDef.
ItemGroupRepeatKey	repeatKey	*(optional)*	A key used to distinguish between repeats of the same

			type of item group within a single form.
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	The TransactionType attribute need not be present in a Snapshot document.

Contained in:

FormData, ReferenceData

Clinical data for an item group (record). The model supports repeating item groups in a single form (for example, when several previous hospitalizations are reported in one history), or within reference data.

The ItemGroupOID and ItemGroupRepeatKey are used together to identify a particular item group. This pair of values uniquely identifies an ItemGroup within the containing Form. The ItemGroupRepeatKey is present if and only if the ItemGroupDef is repeating.

ItemGroupData can also occur as reference data. In that case, the ItemGroupOID and ItemGroupRepeatKey pair must be unique within the reference data.

3.1.4.1.1.1.1 ItemData

Body:

(**AuditRecord**?, **Signature**?, **MeasurementUnitRef**?, **Annotation***)

Attributes:

ItemOID	oidref		Reference to the ItemDef.
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	The TransactionType attribute need not be present in a Snapshot document.
Value	text	*(optional)*	The data collected for an item. This data is represented according to DataType attribute of the ItemDef.
IsNull	(Yes)	*(optional)*	IsNull is a flag to signify that an item's value is to be set to null. If the Value attribute is set, IsNull must not be set. If IsNull is set, the Value attribute must not be provided. In the interest of creating non-verbose XML instances, one should not use ItemData elements with IsNull set to "Yes" to indicate uncollected data. The better practice is to transmit only collected data.

Contained in:

ItemGroupData

The ItemData element may be used for un-typed transmission of the clinical data for an item. The model does *not* support repeating items within a single item group.

The ItemOID is used to identify a particular item definition. This value uniquely identifies an Item within the containing ItemGroup.

The referenced ItemDef defines the DataType of this item. The Value attribute string must obey the rules for that format of data listed in the Data Formats section. For text, string, integer or float datatypes, it must be possible to store the item value within the Length constraint defined in the ItemDef.

Typed and untyped data must not both be used within a single ODM file.

3.1.4.1.1.1.2 ItemData*[TYPE]*

Body:

(PCDATA)

Attributes:

ItemOID	oidref		Reference to the ItemDef.
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	The TransactionType attribute need not be present in a Snapshot document.
AuditRecordID	IDREF	*(optional)*	Reference to AuditRecord.
SignatureID	IDREF	*(optional)*	Reference to Signature.
AnnotationID	IDREF	*(optional)*	Reference to Annotation.
MeasurementUnitOID	oidref	*(optional)*	Reference to the MeasurementUnit definition.
IsNull	(Yes)	*(optional)*	Only allowed for ItemDataAny.

IsNull is a flag to signify that an item's value is to be set to null. If element content (PCDATA) is given, IsNull must not be set. If IsNull is set, element content (PCDATA) must not be provided. In the interest of creating non-verbose XML instances, one should not use ItemData elements with IsNull set to "Yes" to indicate uncollected data. The better practice is to transmit only collected data.

Contained in:

ItemGroupData

ItemData*[TYPE]* elements provide typed data content as element PCDATA. The PCDATA datatype must match the DataType attribute in the corresponding ItemDef. For example, an ItemDataInteger element can be used only for items where the corresponding ItemDef Datatype attribute value is integer. Additionally in this case, the PCDATA content must be a parseable integer value. See also ItemDataAny for content parsing exceptions.

Typed and untyped data must not both be used within a single ODM file.

The following ItemData*[TYPE]* elements are valid:

ItemData**Any**
ItemData**String**
ItemData**Integer**
ItemData**Float**
ItemData**Date**
ItemData**Time**
ItemData**Datetime**
ItemData**Boolean**
ItemData**HexBinary**
ItemData**Base64Binary**
ItemData**HexFloat**
ItemData**Base64Float**
ItemData**PartialDate**
ItemData**PartialTime**
ItemData**PartialDatetime**
ItemData**DurationDatetime**
ItemData**IntervalDatetime**
ItemData**IncompleteDatetime**
ItemData**IncompleteDate**
ItemData**IncompleteTime**
ItemData**URI**

To transmit a value where the DataType does not match the *TYPE*, an ItemDataAny element must be used. An application receiving ItemDataAny data is not required to load these data values into the corresponding data fields.

To transmit a value where the DataType does not match the TYPE, an ItemDataAny element can be used.

Examples:

[Typed (version 1.3) Data Transmission]

Valid data	<ItemDataInteger ItemOID="ID.INT">1</ItemDataInteger>	
Invalid data	<ItemDataAny ItemOID="ID.INT">text</ItemDataAny>	**correct**
Invalid data	<ItemDataInteger ItemOID="ID.INT">text</ItemDataInteger>	**incorrect**

[Un-typed (version 1.2) Data Transmission]

Valid data	<ItemData ItemOID="ID.INT" Value="1"/>
Invalid data	<ItemData ItemOID="ID.INT" Value="text"/>

3.1.4.1.1.2 ArchiveLayoutRef

Body:

EMPTY

Attributes:

ArchiveLayoutOID oidref Reference to the ArchiveLayout.

Contained in:
FormData

A reference to the archive layout used to collect the form's data.

On Update, a NULL ArchiveLayoutOID is permitted.

3.1.4.1.2 AuditRecord

Body:
(**UserRef**, **LocationRef**, **DateTimeStamp**, **ReasonForChange**?, **SourceID**?)

Attributes:

EditPoint	(Monitoring | DataManagement | DBAudit)	*(optional)*	
UsedImputationMethod	(Yes | No)	*(optional)*	
ID	ID	*(optional)*	If an AuditRecord is contained within an AuditRecords element, the ID attribute must be provided.

Contained in:
FormData, ItemData, ItemGroupData, StudyEventData, SubjectData, AuditRecords

An AuditRecord carries information pertaining to the creation, deletion, or modification of clinical data. This information includes who performed that action, and where, when, and why that action was performed.

The EditPoint attribute identifies the phase of data processing in which action occurred, and the UsedImputationMethod attribute indicates whether the action involved the use of a Method. (Note: In ODM 1.3, the new element MethodDef was introduced and the ImputationMethod element was deprecated).

Note: The monitoring phase includes any data collection and correction involving the clinician, study-site personnel, or study monitor. The data management phase includes any changes made for internal data processing reasons (for example, coding or updating derived data) and before database lock. The DB audit phase occurs after database lock.

AuditRecord information describes a change to clinical data, but is not itself clinical data. The value of some clinical data can always be changed by a subsequent transaction, but history cannot be changed -- only added to.

Whenever an element has a TransactionType (either explicit or inherited), an AuditRecord must be provided. However, an AuditRecord has no meaning in a Snapshot transfer.

As with the TransactionType attribute, an AuditRecord is inherited by any subelement that is allowed to have an AuditRecord and does not already have one attached.

Note: Contrast with Signature.

3.1.4.1.2.1 UserRef

Body:
EMPTY

Attributes:

UserOID	oidref	Reference to the User definition.

Contained in:
AuditRecord, Signature

3.1.4.1.2.2 DateTimeStamp

Body:
datetime

Attributes:
NONE

Contained in:
AuditRecord, Signature

The date/time that the data entry, modification, or signature was performed. This applies to the initial occurrence of the action, not to subsequent transfers between computer systems.

The DateTimeStamp datatype is datetime. A value conforming to the ISO 8601 Complete Representation must be provided. Partial or incomplete forms are not valid.

3.1.4.1.2.3 ReasonForChange

Body:
text

Attributes:
NONE

Contained in:
AuditRecord

A user-supplied reason for a data change.

3.1.4.1.2.4 SourceID

Body:
text

Attributes:
NONE

Contained in:
AuditRecord

Information that identifies the source of the data within an originating system. It is only meaningful within the context of that system.

Example: A SourceID attached to an ItemGroup could be used to carry the originating system's internal record OID for that data.

3.1.4.1.3 Signature

Body:
(**UserRef**, **LocationRef**, **SignatureRef**, **DateTimeStamp**,**CryptoBindingManifest? *Deprecated***)

Attributes:
ID ID *(optional)* If a Signature element is contained within a Signatures element, the ID attribute is required.

Contained in:
FormData, ItemData, ItemGroupData, StudyEventData, SubjectData, Signatures

An electronic signature applies to a collection of clinical data. This indicates that some user accepts legal responsibility for that data. See 21 CFR Part 11. The signature identifies the person signing, the location of signing, the signature meaning (via the referenced SignatureDef), the date and time of signing, and (in the case of a digital signature) an encrypted hash of the included data.

An electronic signature applies to the entity to which it is attached (typically a form). The signature covers all clinical data in that entity at the time of the signing, including all clinical data in any subentities. Thus, a signature on a form includes all clinical data at the form, item group, and item levels.

For the purpose of this definition, clinical data includes all attributes and element content *except* for TransactionType attributes, Signature elements, and AuditRecord elements.

Note: A Signature applies to data at a particular point in time. A Signature cannot be modified. However, new signatures can be applied when clinical data changes.

Note: A Signature applies to the entire contents of the *entity*, not just the content that is mentioned in the current ODM *element*. A Signature applies to (static) content, not to a change in content.

Note: The practice of initialing and dating changes on paper forms is a way of maintaining an audit trail. It should not be confused with 21 CFR Part 11 electronic signatures (Signature elements).

Note: Contrast with AuditRecord, and ds:Signature.

Note: the CryptobindingManifest element has been deprecated.

3.1.4.1.3.1 SignatureRef

Body:
EMPTY

Attributes:

SignatureOID	oidref	Reference to the SignatureDef.

Contained in:

Signature

3.1.4.1.4 Annotation

Body:

(Comment?, Flag*)

Attributes:

SeqNum	integer		
TransactionType	(Insert | Update | Remove | Upsert | Context)	*(optional)*	
ID	ID	*(optional)*	If an Annotation is contained with an Annotations element, the ID attribute is required.

Contained in:

Association, FormData, ItemData, ItemGroupData, StudyEventData, SubjectData, Annotations

A general note about clinical data. If an annotation has both a comment and flags, the flags should be related to the comment.

The SeqNum attribute (a small positive integer) uniquely identifies the annotation within its parent entity.

An empty Annotation (one with no comment and no flags) is not allowed unless the TransactionType is Remove. On Update, the entire value of the annotation is replaced.

3.1.4.1.4.1 Comment

Body:

text

Attributes:

SponsorOrSite	(Sponsor | Site)	*(optional)*	Source of the comment.

Contained in:

Annotation

A free-text (uninterpreted) comment about clinical data. The comment may have come from the Sponsor or the clinical Site.

3.1.4.1.4.2 Flag

Body:

(**FlagValue**, **FlagType**?)

Attributes:

NONE

Contained in:

Annotation

A machine-processable annotation on clinical data.

3.1.4.1.4.2.1 FlagValue

Body:

text

Attributes:

CodeListOID	oidref	Reference to the CodeList definition.

Contained in:

Flag

The value of the flag. The meaning of this value is typically dependent on the associated FlagType. The actual value must be a member of the referenced CodeList.

3.1.4.1.4.2.2 FlagType

Body:
name

Attributes:
CodeListOID oidref Reference to the CodeList definition.

Contained in:
Flag

The type of flag. This determines the purpose and semantics of the flag. Different applications are expected to be interested in different types of flags. The actual value must be a member of the referenced CodeList.

3.1.4.1.5 InvestigatorRef

Body:
EMPTY

Attributes:
UserOID oidref Reference to a User definition.

Contained in:
SubjectData

3.1.4.1.6 SiteRef

Body:
EMPTY

Attributes:
LocationOID oidref Reference to a Location definition.

Contained in:
SubjectData

3.1.4.2 AuditRecords

Body:
(**AuditRecord***)

Attributes:
NONE

Contained in:
ReferenceData, ClinicalData

Groups AuditRecord elements referenced by ItemData*[TYPE]* elements.

3.1.4.3 Signatures

Body:
(**Signature***)

Attributes:
NONE

Contained in:
ReferenceData, ClinicalData

Groups Signature elements referenced by ItemData*[TYPE]* elements.

3.1.4.4 Annotations

Body:
(**Annotation***)

Attributes:
NONE

Contained in:
ReferenceData, ClinicalData

Groups Annotation elements referenced by ItemData*[TYPE]* elements.

3.1.5 Association

Body:
(**KeySet**, **KeySet**, **Annotation**)

Attributes:

StudyOID	oidref	
MetaDataVersionOID	oidref	

Contained in:
ODM

An association permits an annotation to be placed on an ordered pair of entities rather than on just one. The first and second KeySets identify the start and end of the annotated "link".

The StudyOID and MetaDataVersionOID provide a context for interpreting CodeList references within the Annotation. They do not affect the meaning of the KeySets.

3.1.5.1 KeySet

Body:
EMPTY

Attributes:

StudyOID	oidref	
SubjectKey	subjectKey	*(optional)*
StudyEventOID	oidref	*(optional)*
StudyEventRepeatKey	repeatKey	*(optional)*
FormOID	oidref	*(optional)*
FormRepeatKey	repeatKey	*(optional)*
ItemGroupOID	oidref	*(optional)*
ItemGroupRepeatKey	repeatKey	*(optional)*
ItemOID	oidref	*(optional)*

Contained in:
Association

A KeySet references a single entity: a study, a subject, a study event, etc. Only the attributes needed to specify the particular entity are required, and all others must be omitted. (See the section on Clinical Data Keys.)

4 Digital Signatures

4.1 ds:Signature

Body:
(**SignedInfo**, **SignatureValue**, **KeyInfo**?, **Object***)

Attributes:

xmlns	CDATA	
Id	ID	*(optional)*

Contained in:
ODM

The ds:Signature element provides the ability to protect the integrity of an ODM document with a digital signature. This element and its subelements are taken from the W3C's XML Digital Signature Specification. See that specification for the semantics of digital signatures.

Note: A Signature element expresses the fact that some human being is taking responsibility for the completeness and accuracy of a set of clinical data. In contrast a ds:Signature element is used by applications to ensure that modifications to the document's contents can be detected.

5 Element Index (non-normative)

ItemDataFloat
ItemDataHexBinary
ItemDataHexFloat
ItemDataIncompleteDate
ItemDataIncompleteDatetime
ItemDataIncompleteTime
ItemDataInteger
ItemDataIntervalDatetime
ItemDataPartialDate
ItemDataPartialDatetime
ItemDataPartialTime
ItemDataString
ItemDataTime
ItemDataURI
ItemDef
ItemGroupData
ItemGroupDef
ItemGroupRef
ItemRef
KeySet
LastName
LegalReason
Location
LocationRef
LoginName
Meaning
MeasurementUnit
MeasurementUnitRef
MetaDataVersion
MetaDataVersionRef
MethodDef
ODM
Organization
OtherText
Pager
Phone
Picture
PostalCode
Presentation
Protocol
ProtocolName
Question
RangeCheck
ReasonForChange
ReferenceData
Role ***Deprecated 1.2***
Signature
SignatureDef
SignatureRef
Signatures
SiteRef
SourceID
StateProv
StreetName
Study
StudyDescription
StudyEventData
StudyEventDef
StudyEventRef
StudyName

SubjectData
Symbol
TranslatedText
User
UserRef
ds:Signature

6 Attribute Index (non-normative)

This index lists all the attributes used, along with the elements that use them.

AnnotationID	ItemData*[TYPE]*
Archival	ODM
ArchiveLayoutOID	ArchiveLayoutRef
AsOfDateTime	ODM
AuditRecordID	ItemData*[TYPE]*
Category	StudyEventDef
Code	ExternalQuestion
CodedValue	CodeListItem
CodeListOID	CodeListRef, FlagType, FlagValue
CollectionExceptionConditionOID	StudyEventRef, FormRef, ItemGroupRef, ItemRef
Comment	ItemDef, ItemGroupDef
Comparator	RangeCheck
Context	Alias,FormalExpression
CreationDateTime	ODM
DataType	CodeList, ItemDef
Description	MetaDataVersion, ODM
Dictionary	ExternalCodeList, ExternalQuestion
Domain	ItemGroupDef
EditPoint	AuditRecord
EffectiveDate	MetaDataVersionRef
FileOID	ODM
FileType	ODM
FormOID	FormData, FormRef, KeySet
FormRepeatKey	FormData, KeySet
Granularity	ODM
ID	ODM, AuditRecord, Signature, Annotation
Id	ds:Signature
ImageType	Picture
ImputationMethodOID	***Deprecated 1.3***
IsNull	ItemData
IsReferenceData	ItemGroupDef
ItemGroupOID	ItemGroupData, ItemGroupRef, KeySet
ItemGroupRepeatKey	ItemGroupData, KeySet
ItemOID	ItemData, ItemRef, KeySet
KeySequence	ItemRef
Length	ItemDef
LocationOID	LocationRef, SiteRef
LocationType	Location
Mandatory	FormRef, ItemGroupRef, ItemRef, StudyEventRef
MeasurementUnitOID	MeasurementUnitRef, ItemData*[TYPE]*
MetaDataVersionOID	Association, ClinicalData, Include, MetaDataVersionRef, ReferenceData
MethodOID	ItemRef
Methodology	SignatureDef
Name	Alias, CodeList, ConditionDef, FormDef, ItemDef, ItemGroupDef, Location, MeasurementUnit, MetaDataVersion, MethodDef, StudyEventDef
ODMVersion	ODM
OID	ArchiveLayout, CodeList, ConditionDef, FormDef, ItemDef, ItemGroupDef, Location, MeasurementUnit, MetaDataVersion, MethodDef, Presentation, SignatureDef, Study, StudyEventDef, User
OrderNumber	FormRef, ItemGroupRef, ItemRef, StudyEventRef, CodeListItem, EnumeratedItem

Origin	ItemDef, ItemGroupDef
Originator	ODM
PdfFileName	ArchiveLayout
PictureFileName	Picture
PresentationOID	ArchiveLayout
PriorFileOID	ODM
Purpose	ItemGroupDef
Rank	CodeListItem, EnumeratedItem
ref	ExternalCodelist
Repeating	FormDef, ItemGroupDef, StudyEventDef
Role	ItemGroupDef, ItemRef
RoleCodeListOID	ItemRef
SASDatasetName	ItemGroupDef
SASFieldName	ItemDef
SASFormatName	CodeList
SDSVarName	ItemDef
SeqNum	Annotation
SignatureID	ItemData*[TYPE]*
SignatureOID	SignatureRef
SignificantDigits	ItemDef
SoftHard	RangeCheck
SourceSystem	ODM
SourceSystemVersion	ODM
SponsorOrSite	Comment
StudyEventOID	KeySet, StudyEventData, StudyEventRef
StudyEventRepeatKey	KeySet, StudyEventData
StudyOID	AdminData, Association, ClinicalData, Include, KeySet, MetaDataVersionRef, ReferenceData
SubjectKey	KeySet, SubjectData
TransactionType	Annotation, FormData, ItemData, ItemGroupData, StudyEventData, SubjectData
Type	StudyEventDef, MethodDef
UsedImputationMethod	AuditRecord
UserOID	InvestigatorRef, UserRef
UserType	User
Value	ItemData
Version	ExternalCodeList, ExternalQuestion
xml:lang	Presentation, TranslatedText
xmlns	ds:Signature

7 Review Period; Licensing Obligations, Representations and Warranties; Limitations of Liability, and Disclaimers

The following text is excerpted from the CDISC IP Policy available at http://www.cdisc.org/about/bylaws_pdfs/CDISCIPPolicy-FINAL.pdf.

From Section 5:

5 Treatment of Non-Contributed Necessary Claims

5.1 Review Period / Licensing Obligations for Non-Contributed Necessary Claims.

To ensure that all Participants have the opportunity to review each proposed Draft Standard as a complete document for purposes of identifying Non-Contributed Necessary Claims, CDISC shall, at least thirty (30) calendar days prior to the finalization of the Draft Standard, notify all Participants via e-mail and post the Draft Standard on the CDISC website for open review. During the review period, each Participant in the CDISC Group that developed the Draft Standard under review (as well as others that choose to take part in such review) shall:

(a) review the Draft Standard;

(b) disclose, pursuant to Section 5.2, any of its Non-Contributed Necessary Claim(s) covering such Draft Standard that the Participant is not willing to license to all implementers of the resulting Final Standard under the compensation-free and otherwise reasonable and non-discriminatory commitment of Section 3; and

(c) submit, prior to the end of the review period, a written licensing declaration for any Non-Contributed Necessary Claim disclosed under subclause (b) indicating whether the Participant will commit to license such disclosed claim(s) at least on reasonable and non-discriminatory (.RAND.) terms and conditions (note, Necessary Claims that cover one's own Contributions must be licensed compensation-free pursuant to Section 3).

From Section 6.2

CDISC Patent Disclaimers. It is possible that implementation of and compliance with this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any claim or of any patent rights in connection therewith. CDISC, including the CDISC Board of Directors, shall not be responsible for identifying patent claims for which a license may be required in order to implement this standard or for conducting inquiries into the legal validity or scope of those patents or patent claims that are brought to its attention.

From Section 9:

9.3 Representations and Warranties.

Each Participant in the development of this standard shall be deemed to represent, warrant, and covenant, at the time of a Contribution by such Participant (or by its Representative), that to the best of its knowledge and ability: (a) it holds or has the right to grant all relevant licenses to any of its Contributions in all jurisdictions or territories in which it holds relevant intellectual property rights; (b) there are no limits to the Participant's ability to make the grants, acknowledgments, and agreements herein; and (c) the Contribution does not subject any Contribution, Draft Standard, Final Standard, or implementations thereof, in whole or in part, to licensing obligations with additional restrictions or requirements inconsistent with those set forth in this Policy, or that would require any such Contribution, Final Standard, or implementation, in whole or in part, to be either: (i) disclosed or distributed in source code form; (ii) licensed for the purpose of making derivative works (other than as set forth in Section 4.2 of the CDISC Intellectual Property Policy ("the Policy")); or (iii) distributed at no charge, except as set forth in Sections 3, 5.1, and 4.2 of the Policy. If a Participant has knowledge that a Contribution made by any Participant or any other party may subject any Contribution, Draft Standard, Final Standard, or implementation, in whole or in part, to one or more of the licensing obligations listed in Section 9.3, such Participant shall give prompt notice of the same to the CDISC President who shall promptly notify all Participants.

9.4 No Other Warranties/Disclaimers.

ALL PARTICIPANTS ACKNOWLEDGE THAT, EXCEPT AS PROVIDED UNDER SECTION 9.3 OF THE CDISC INTELLECTUAL PROPERTY POLICY, ALL DRAFT STANDARDS AND FINAL STANDARDS, AND ALL CONTRIBUTIONS TO FINAL STANDARDS AND DRAFT STANDARDS, ARE PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, AND THE PARTICIPANTS, REPRESENTATIVES, THE CDISC PRESIDENT, THE CDISC BOARD OF DIRECTORS, AND CDISC EXPRESSLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR OR INTENDED PURPOSE, OR ANY OTHER WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, FINAL STANDARDS OR DRAFT STANDARDS, OR CONTRIBUTION.

9.5 Limitation of Liability.

IN NO EVENT WILL CDISC OR ANY OF ITS CONSTITUENT PARTS (INCLUDING, BUT NOT LIMITED TO, THE CDISC BOARD OF DIRECTORS, THE CDISC PRESIDENT, CDISC STAFF, AND CDISC MEMBERS) BE LIABLE TO ANY OTHER PERSON OR ENTITY FOR ANY LOSS OF PROFITS, LOSS OF USE, DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS POLICY OR ANY RELATED AGREEMENT, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.