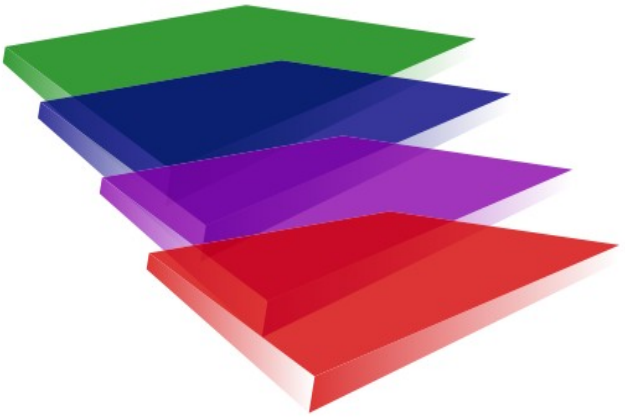
Multi-Level Health Information Modelling



MLHIM

Reference Manual

Release 2.3.0
2012-08-01

The goal of MLHIM is to be Minimalistic, Sustainable, Implementable AND Interoperable.

Table of Contents

Front Matter

Acknowledgements

This work has received financial and in-kind support from the following persons and organizations:

- National Institute of Science and Technology on Medicine Assisted by Scientific Computing (INCT-MACC), coordinated by the National Laboratory of Scientific Computing (macc.lncc.br)
- Multilevel Healthcare Information Modeling Laboratory, Associated to the INCT-MACC (mlhim.lampada.uerj.br)
- Timothy W. Cook, Independent Consultant
- Roger Erens, Independent Consultant

Using the MLHIM Reference Manual

This section describes typographical conventions and other information to help you get the most from this document.

The intended audience for this manual includes; software developers, systems analysts and knowledge workers in the healthcare domain. It is assumed that the reader has knowledge of object-oriented notation, concepts and software construction practices.

In the PDF release of these specifications cross reference links are denoted by a number inside square brackets i.e. [5].

Release Information

The official published version is in PDF format in the English language. The ODT version is always considered a work in progress. Each version of the PDF release will carry a version number that is the date of release followed by the language code and locality code. As an example, a release in English on January 1, 2011 will have the filename:

mlhim-ref-man-2011-01-01-en-US.pdf

Error Reporting

Please report all errors in documentation and/or in the specifications of the information model as bug reports at the Launchpad development site. It is easy to do and a great way to give back. See: https://bugs.launchpad.net/mlhim-specs

Pronunciation

MLHIM is pronounced muh-leem. Click Hear How It Sounds for when it is used in spoken English language such as presentations or general discussions.

Conformance

Conformance to these specification are represented in a Language Implementation Specification (LIS). A LIS is formal document detailing the mappings and conventions used in relation to these specifications.

A LIS is in direct conformance to these specifications when:

1. All datatypes are defined and mapped.
2. the value spaces of the healthcare datatypes used by the entity to be identical to the value spaces specified herein
3. to the extent that the entity provides operations other than movement or translation of values, define operations on the healthcare datatypes which can be derived from, or are otherwise consistent with the characterizing operations specified herein

Compliance

These specifications:

- are in indirect conformance with ISO/DIS 21090/2008
- are in compliance with applicable sections of ISO 18308/2008
- are in compliance with applicable sections of ISO/TR 20514:2005
- are in compliance with applicable sections of ISO 13606-1:2007

Introduction

The Multi-Level Health Information Modeling (MLHIM) specifications are partially derived from ISO Healthcare Information Standards and the *open*EHR 1.0.2 specifications and the intent is that MLHIM 1.x be technologically inter-operable with *open*EHR.

MLHIM 2.x (this document and related artifacts) introduces modernization through the use of XML technologies and improved modeling tools as well as application development platforms. These specifications can be implemented in any structured language. While a certain level of knowledge is assumed, the primary goal of these specifications is to make them 'readable' by the widest possible number of people. The primary motivator for these specifications is the complexity involved in the recording of the temporal-spatial relationships in healthcare information while maintaining the original semantics across all applications; for all time.

We invite you to join us in this effort to maintain the specifications and build great, translatable healthcare tools and applications for global use.

International input is encouraged in order for the MLHIM specifications to be truly inter-operable, available to everyone in all languages and most of all, implementable by mere mortals.

In actual implementation, the packages/classes should be implemented per the standard language naming format. A Language Implementation Specification (LIS) should be created for each language. For example MLHIM-Python-LIS.odt or MLHIM-Java-LIS.odt.

MLHIM intentionally does not specify behavior within a class. Only the data and constraints are specified. Behavior may differ between various applications and should not be specified at the information model level.

The generic class names in the specification documents are in CamelCase type. Since this is most typical of implementation usage.

Only the reference model is implemented in software. The domain knowledge models are implemented in the XML Schema language and they represent constraints on the reference model. These knowledge models are called Concept Constraint Definitions and the acronyms CCD and CCDs are used throughout MLHIM documents to mean these XML Schema files. This means that, since CCDs form a model that allows the creation of data instances from and according to a specific CCD, it is ensured that the data instances will be valid. However, any data instance should be able to be imported into any MLHIM based application since the root data model is the reference model. But, the full semantics of that data will not be known unless and until the defining CCD is available to that application.

The above highlighted paragraph describes the foundation of semantic interoperability in MLHIM implementations. You must understand this and the implications it carries to be successful with implementing MLHIM based applications. See the Constraint section for an in-depth discussion of Concept Constraint Definitions (CCDs).

The MLHIM Eco-System

It is important here to describe all of the components of the MLHIM conceptual eco-system in order for the reader to appreciate the scope of MLHIM and the importance of the governance policies.

At the base of the system is the Reference Model (RM). Though the reference implementation is in XML Schema format, in real world applications a chosen object oriented language will likely be used for implementations. Often, tools are available to automatically generate the reference model classes from the XML Schema. This is the basis for larger MLHIM compliant applications. We will later cover implementation options for small, purpose specific devices such as a home blood pressure monitor.

The next level of the multi-level hierarchy is the Concept Constraint Definition (CCD). The CCD is a set of constraints against the RM that narrow the valid data options to a point where they can represent a specific healthcare concept. The CCD is essentially an XML Schema that uses the RM complex types as base types. Basically this is inheritance in XML Schema.

Since a CCD can only narrow the constraints of the RM. Then any data instance that is compliant with a CCD is also compliant in any software application that implements the RM. Even if the CCD is not available, an application can know how to display and even analyze certain information. However, the MLHIM eco-system concept does expect that every CCD for any published information is available to the receiving system(s).

This is not to imply that all CCDs must be publicly available. It is possible to maintain a set of CCDs within a certain political jurisdiction or within a certain professional sector. Simply install a copy of the Healthcare Knowledge Component Repository (HKCR) (http://www.hkcr.net) and restrict access as required.

This is now the point where the MLHIM eco-system is in contrast to the top-down approach used by other multi-level modelling specifications. In the real world; we know that there can never be complete consensus across the healthcare spectrum of domains, cultures and languages; concerning the details of a specific concept. Therefore the concept of a “maximal data model”, though idealistically valid, is realistically unattainable. In MLHIM, participants at any level are encouraged to create concept models that fit their needs. The RM has very little semantic context in it to get in the way. This allows structures to be created as the modeler sees fit for purpose. There is no inherent idea of a specific application in the RM, such as an EHR, EMR, etc. This provides an opening for small, purpose specific apps such as mobile or portable device software as well.

In MLHIM, there is room for thousands of CCDs to describe blood pressure (or any other phenomena) vs. a single model that must encompass all descriptions/uses/etc. Each CCD is uniquely identified by a Version 4 Universal Unique Identifier (UUID)[1] prefixed with 'ccd_'. CCDs are pluggable so that modelers can use small CCDs to create any size concept, document, etc. needed. Modelers and developers can create systems that allow users to choose between a selection of CCDs to include at specific points, at runtime.

With MLHIM CCDs you can deliver your data with complete syntactic interoperability and as much semantic interoperability as the modeler chose to include in the CCD.

1 http://en.wikipedia.org/wiki/Universally_unique_identifier

The governance of CCDs is straight forward. A web application known as the HKCR lives at http://www.hkcr.net Anyone may install a copy of this for local use as well for performance reasons or if there is a need for some CCDs to not be published globally. Anyone may signup for an account on HKCR.net and anyone may create CCDs and submit them for publication. The only check that is made is that there is a valid XSD file. The content is completely up to the modeler and any potential users. Like other global meritocracies, those that create good and useful models will see theirs reused many times. Others, not so much.

MLHIM Modelling

Approach

The MLHIM specifications are arranged into packages. These packages represent logical groupings of classes providing ease of consistent implementation. The fundamental concepts, expressed in the reference model classes, are based on basic philosophical concepts of real world entities. These broad concepts can then be constrained to more specific concepts using models created by domain experts, in this case healthcare experts.

In MLHIM 1.x.y these constraints were known as archetypes, expressed in a domain specific language (DSL) called the archetype definition language (ADL). This language is based on a specific model called the archetype object model (AOM).

In MLHIM 2.x and later, we use an XML Schema (XSD) representation called a Concept Constraint Definition (CCD). This allows MLHIM to use the XML Schema language as the constraint language. This provides the MLHIM development community with an approach to using multi-level modelling in healthcare using standardized tools and technologies.

CCDs may contain other CCDs in a structure called a Slot. This provides a basis for selection and reuse, at runtime, of commonly occurring concepts within a larger context. A CCD is 'ideally', a maximal data model for a concept. However, as has been learned from other approaches, the concept of a maximal data model for a concept is still restricted to the knowledge and context of that particular domain expert. This means that from a global perspective there may be several CCDs that purport to fill the same need. There is no conflict in the MLHIM world in this case as CCDs are named using the UUID. The CCD may be further constrained at the implementation level through the use of implementation templates in the selected framework. These templates shall be constructed in the implementation and may or may not be sharable across applications. The MLHIM specifications do not play any role in defining what these templates look like or perform like. They are only mentioned here as a way of making note that applications will require a template layer to be functional.

The real advantage to using the MLHIM approach to health care information modelling is that it provides for a wide variety of healthcare applications to be developed based on the broad concepts defined in the reference model. Then by having domain experts within the healthcare field define and create the CCDs. They can then be shared across multiple applications so that the structure of the data is not locked into one specific application. But can be exchanged among many different applications. This properly implements the separation of roles between IT people and domain experts.

To demonstrate the differences between the MLHIM approach and the typical data model design approach we will use two common metaphors.

1. The first is for the data model approach to developing software. This is where a set of database definitions are created based on a requirements statement representing an information model. An application is then developed to support all of the functionality required to input, manipulate and output this data as information, all built around the data model. This approach is akin to a jigsaw puzzle (the software application) where the shape of the pieces are the syntax and the design and colors are the semantics, of the information represented in an aggregation of data components described by the model. This produces an application that, like the jigsaw puzzle, can provide for pieces (information) to be exchanged only between exact copies of the same puzzle. If you were to try to put pieces from one puzzle, into a different puzzle you might find that a piece has the same shape (syntax) but the picture on the piece (semantics) will not be the same. Even though they both belong to the same domain of jigsaw puzzles. You can see that getting a piece from one puzzle to correctly fit into another is going to require manipulation of the basic syntax (shape) and /or semantics (picture) of the piece. This can also be extended to the relationship that the puzzle has a defined limit of its six sides. It cannot, reasonably, be extended to incorporate new pieces (concepts) discovered after the initial design.
2. The multi-level approach used in MLHIM is akin to creating models (applications) using the popular toy blocks made by Lego and other companies. If you compare a box of these interlocking blocks to the reference model and the instructions to creating a specific toy model (software application), where these instructions present a concept constraint. You can see that the same blocks can be used to represent multiple toy models without any change to the physical shape, size or color of each block. Now we can see that when new concepts are created within healthcare, they can be represented as instructions for building a new toy model using the same fundamental building blocks that the original software applications were created upon.

Assumed Types

There are several types that are assumed to be supported by the underlying implementation technology. They are described here.

anyType

Also sometimes called **Object.** This is the base class of the implementation technology.

boolean

Two state only. Either true or false.

string

The string data type can contain characters, line feeds, carriage returns, and tab characters.

normalizedString

The normalized string data type also contains characters, but all line feeds, carriage returns, and tab characters are removed.

token

The token data type also contains characters, but the line feeds, carriage returns, tabs, leading and trailing spaces are removed, and multiple spaces are replaced with one space.

anyURI

Specifies a URI.

hash

An enumeration of any type with a key:value combination. The keys must be unique.

list

An ordered or unordered list of any type.

set

An ordered or unordered but unique list of any type.

Ordered Types:

dateTime

The dateTime data type is used to specify a date and a time.

The dateTime is specified in the following form "YYYY-MM-DDThh:mm:ss" where:

YYYY indicates the year
MM indicates the month
DD indicates the day
T indicates the start of the required time section
hh indicates the hour
mm indicates the minute
ss indicates the second

The following is an example of a dateTime declaration in a schema:
<xs:element name="startdate" type="xs:dateTime"/>

An element in your document might look like this:
<startdate>2002-05-30T09:00:00</startdate>

Or it might look like this:
<startdate>2002-05-30T09:30:10:05</startdate>

Time Zones

To specify a time zone, you can either enter a dateTime in UTC time by adding a "Z" behind the time - like this:
<startdate>2002-05-30T09:30:10Z</startdate>
or you can specify an offset from the UTC time by adding a positive or negative time behind the time - like this:
<startdate>2002-05-30T09:30:10-06:00</startdate>

or

<startdate>2002-05-30T09:30:10+06:00</startdate>

date

The date data type is used to specify a date.

The date is specified in the following form "YYYY-MM-DD" where:

YYYY indicates the year
MM indicates the month
DD indicates the day

An element in an XML Document might look like this:
<start>2002-09-24</start>

time

The time data type is used to specify a time.

The time is specified in the following form "hh:mm:ss" where:

hh indicates the hour
mm indicates the minute
ss indicates the second

The following is an example of a time declaration in a schema:
<xs:element name="start" type="xs:time"/>

An element in your document might look like this:
<start>09:00:00</start>

Or it might look like this:
<start>09:30:10:05</start>

Time Zones

To specify a time zone, you can either enter a time in UTC time by adding a "Z" behind the time - like this:
<start>09:30:10Z</start>

or you can specify an offset from the UTC time by adding a positive or negative time behind the time - like this:
<start>09:30:10-06:00</start> or <start>09:30:10+06:00</start>

duration

The duration data type is used to specify a time interval.

The time interval is specified in the following form "PnYnMnDTnHnMnS" where:

P indicates the period (required)
nY indicates the number of years
nM indicates the number of months
nD indicates the number of days
T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds)
nH indicates the number of hours
nM indicates the number of minutes
nS indicates the number of seconds

The following is an example of a duration declaration in a schema:
<xs:element name="period" type="xs:duration"/>

An element in your document might look like this:
<period>P5Y</period>

The example above indicates a period of five years.

Or it might look like this:
<period>P5Y2M10D</period>

The example above indicates a period of five years, two months, and 10 days.

Or it might look like this:
<period>P5Y2M10DT15H</period>

The example above indicates a period of five years, two months, 10 days, and 15 hours.

Or it might look like this:
<period>PT15H</period>

The example above indicates a period of 15 hours.
Negative Duration

To specify a negative duration, enter a minus sign before the P:
<period>-P10D</period>

The example above indicates a period of minus 10 days.

Partial Date Types

In order to provide for partial dates and times the following types are assumed to be available in the language or in a library.

Day – provide on the day of the month, 1 – 31
Month – provide only the month of the year, 1 – 12
Year – provide on the year, CCYY
MonthDay – provide only the Month and the Day (no year)
YearMonth – provide only the Year and the Month (no day)

real

The decimal data type is used to specify a numeric value.
Note: The maximum number of decimal digits you can specify is 18.

integer

The integer data type is used to specify a numeric value without a fractional component.

The Reference Model

Please note: Any discrepancies between this document and the Xmind template at: http://www.hkcr.net/tools/cdd-template-xmind/view are to be resolved in favor of the template. The template has directly informed the XML Schema implementation and there hasn't been man-power available to keep this document up to date.

The automatically generated XML Schema documentation is available in PDF form: http://www.mlhim.org/documentation/specs/schema-docs/pdf-2.3.0/view As well as a browse-able HTML file from the repository at: http://bazaar.launchpad.net/~mlhim-specs-dev/mlhim-specs/main/download/head:/mlhim2.html-20120719134245-ylmv7ow0asadxm75-4/mlhim2.html

The best way to get the most recent changes are to create a Bazaar branch using:
bzr branch lp:mlhim-specs

Datatypes

DvAny

Serves as a common ancestor of all datatypes in MLHIM models.
Inherits From: Any [11]
Abstract: True

Element Name	data_name
Datatype	string [11]
minOccurs	1
maxOccurs	1
Note	Used to tag the datatypes' value in instances.

Element Name	valid_time_begin
Datatype	dateTime [12]
minOccurs	0
maxOccurs	1
Note	Used to tag the earliest datetime that this data element is valid. If present this must be a valid datetime including timezone.

Element Name	valid_time_end
Datatype	dateTime [12]
minOccurs	0
maxOccurs	1
Note	Used to tag the expiration of the validity of this data item. If present this must be a valid datetime string including timezone

Element Name	ev
Datatype	ExceptionalValue [45]
minOccurs	0
maxOccurs	1
Note	The exceptional value. Often referred to as Null Flavour.

DvString

A text item, which may contain any amount of legal characters arranged as e.g. words, sentences etc. as its data value (DvString_dv)
Inherits From: DvAny [16]
Abstract: False

Element Name	language
Datatype	string [tring]
minOccurs	0
maxOccurs	1
Note	Optional indicator of the localised language in which the value is written. Coded IAW IETF RFC 5646. http://tools.ietf.org/html/rfc5646 Only used when the text object is in a different language from the enclosing CCD.

Element Name	DvString_dv
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Displayable rendition of the item.

DvCodedString

A text item whose dv attribute must be the long name or description from a controlled terminology, the key (i.e. the 'code') of which is the code_string attribute.

Inherits From:DvString [17]
Abstract: False

Element Name	terminology_code
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	The key used by the terminology service to identify a concept or coordination of concepts. This string is most likely parsable inside the terminology service, but nothing can be assumed about its syntax outside that context. In the NLM Metathesarus this would be the Concept Unique Identifier (CUI).

Element Name	terminology_name
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	Full Source Name from NLM Metathesarus or similar.

Element Name	terminology_abbrev
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	Version Source Abbreviation (VSAB) from NLM Metathesarus or similar

Element Name	terminology_version
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	The proper release/version ID from the releasing authority.

DvIdentifier

Type for representing identifiers of real-world entities. Typical identifiers include: drivers license number, social security number, veterans affairs number, prescription id, order id, and so on.

Inherits From: DvString [17]

Abstract: False

Element Name	issuer
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	Authority which issues the kind of id used in the id field of this object.

Element Name	assigner
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	Organization that assigned the id to the item or person being identified.

Element Name	name
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	The identifier common name, such as “Driver's License” or “SSN”.

DvBoolean

Items which represent boolean choices, such as true/false or yes/no answers. Use for such is impor - tant to devise the meanings (usually questions in subjective data) carefully, so that the only allowed results are in fact true or false.

Though the DvBoolean.dv attribute is a String type this is to easily allow responses that the user is more familiar with using in the context such as 'Yes', 'No' or 'True', 'False'. A conversion method is required to convert the valid_trues to True and the valid_falses to False.

Inherits From: DvAny [16]
Abstract: False

Element Name	DvBoolean_dv
Datatype	string [11]
minOccurs	0
maxOccurs	1
Note	A representative boolean value in the implementation language. The dv attribute may be a Void or Null value in which case the inherited 'ev' attribute cannot be empty.

Element Name	valid_trues
Datatype	Set [11]<String [11]>
minOccurs	1
maxOccurs	Unbound
Note	A Set of strings representing the valid strings that can be used as True.

Element Name	valid_falses
Datatype	Set [11]<String [11]>
minOccurs	1
maxOccurs	Unbound
Note	A Set of strings representing the valid strings that can be used as False.

DvURI

A reference to an object which conforms to the Universal Resource Identifier (URI) standard, as defined by W3C RFC 2936. See "Universal Resource Identifiers in WWW" by Tim Berners-Lee at http://www.ietf.org/rfc/rfc2396.txt. This is a World-Wide Web RFC for global identification of resources. See http://www.w3.org/Addressing for a starting point on URIs. See http://www.ietf.org/rfc/rfc2806.txt for new URI types like telephone, fax and modem numbers. Enables external resources to be referenced from within the content of the EHR. A number of functions return the logical subparts of the URI string.

Inherits From: DvAny [16]
Abstract: False

Element Name	DvURI_dv
Datatype	anyURI [11]
minOccurs	0
maxOccurs	1
Note	A representative boolean value in the implementation language. The dv attribute may be a Void or Null value in which case the inherited 'ev' attribute cannot be empty.

DvEncapsulated

Abstract class defining the common meta-data of all types of encapsulated data.

Inherits From: DvAny [16]
Abstract: True

Element Name	size
Datatype	Integer [15]
minOccurs	0
maxOccurs	1
Note	Original size in bytes of unencoded encapsulated data. I.e. encodings such as base64, hexadecimal etc do not change the value of this attribute.

Element Name	charset
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Name of character encoding scheme in which this value is encoded. Unicode is the default assumption in MLHIM, with UTF-8 being the assumed encoding. This attribute allows for variations from these assumptions.

DvParsable

Encapsulated data expressed as a parsable String. The internal model of the data item is not described in the MLHIM model in common with other encapsulated types, but in this case, the form of the data is assumed to be plaintext, rather than compressed or other types of large binary data.

Inherits From: DvEncapsulated [22]
Abstract: False

Element Name	dv
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	The string, which may validly be empty in some syntaxes

Element Name	formalism
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Name of the formalism, e.g. "MAG 1.0", “GLIF 1.0”, “PROforma” etc.

DvMedia

A specialisation of DvEncapsulated for audiovisual and biosignal types. Includes further metadata relating to multimedia types which are not applicable to other subtypes of DvEncapsulated.

Inherits From: DvEncapsulated [22]
Abstract: True

Element Name	mime_type
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	MIME type of the original data value (dv) w/o any compression. See IANA registered types: http://www.iana.org/assignments/media-types/index.html

Element Name	compression_type
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Compression/archiving mime-type. Void means no compression/archiving. For a list of common mime-types for compression/archiving see: http://en.wikipedia.org/wiki/List_of_archive_formats

Element Name	hash_result
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Hash function result of the 'dv'. There must be a corresponding hash function type listed for this to have any meaning. See: http://en.wikipedia.org/wiki/List_of_hash_functions#Cryptographic_hash_functions

Element Name	hash_function
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Hash function used to compute hash_result. See: http://en.wikipedia.org/wiki/List_of_hash_functions#Cryptographic_hash_functions

Element Name	uri
Datatype	DvURI [21]
minOccurs	0
maxOccurs	1
Note	URI reference to electronic information stored outside the record as a file, database entry etc, if supplied as a reference.

Element Name	alt_text
Datatype	DvString [17]
minOccurs	0
maxOccurs	1
Note	Text to display in lieu of multimedia display/replay

DvInterval

Generic class defining an interval (i.e. range) of a comparable type. An interval is a contiguous subrange of a comparable base type. Used to define intervals of dates, times, quantities (whose units match) and so on.

Inherits From: DvAny [16]
Abstract: False

Element Name	lower
Datatype	Ordered [12]
minOccurs	0
maxOccurs	1
Note	Lower boundary.

Element Name	upper
Datatype	Ordered [12]
minOccurs	0
maxOccurs	1
Note	Upper boundary.

Element Name	lower_included
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	Is the lower boundary included in the interval?

Element Name	upper_included
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	Is the upper boundary included in the interval?

Element Name	lower_unbounded
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	If True, there is no lower boundary

Element Name	upper_unbounded
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	If True, there is no upper boundary

ReferenceRange

Defines a named range to be associated with any Ordered datum. Each such range is particular to the patient and context, e.g. sex, age, and any other factor which affects ranges. May be used to represent normal, therapeutic, dangerous, critical etc ranges.

Inherits From: DvAny [16]
Abstract: False

Element Name	definition
Datatype	String [11]
minOccurs	1
maxOccurs	1
Note	Term whose value indicates the meaning of this range, e.g. “normal”, “critical”, “therapeutic”etc.

Element Name	data_range
Datatype	DvInterval [25]
minOccurs	1
maxOccurs	1
Note	The data range for this meaning.

DvOrdered

Abstract class defining the concept of ordered values, which includes ordinals as well as true quantities. It defines the functions less than and is_strictly_comparable_to, the latter of which must evaluate to True for instances being compared with the less than function, or used as limits in the DvInterval [25] class. Data value types which are to be used as limits must inherit from this class, and implement the function is_strictly_comparable_to to ensure that instances compare meaningfully. For example, instances of DvQuantity can only be compared if they measure the same kind of physical quantity.

Inherits From: DvAny [16]
Abstract: True

Element Name	normal_range
Datatype	DvInterval [25]
minOccurs	0
maxOccurs	1
Note	Optional normal range.

Element Name	other_reference_ranges
Datatype	List [11]<ReferenceRange [26]>
minOccurs	0
maxOccurs	1
Note	List of ReferenceRanges. Optional tagged other reference ranges for this value in its particular measurement context.

Element Name	normal_status
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Optional normal status indicator of value with respect to normal range for this value. Often included by lab, even if the normal range itself is not included. Coded by ordinals in series HHH, HH, H, (nothing), L, LL, LLL, etc.

DvOrdinal

Models rankings and scores, e.g. pain, Apgar values, etc, where there is a) implied ordering, b) no implication that the distance between each value is constant, and c) the total number of values is finite. Note that although the term 'ordinal' in mathematics means natural numbers only, here any integer is allowed, since negative and zero values are often used by medical professionals for values around a neutral point.

Examples of sets of ordinal values:

-3, -2, -1, 0, 1, 2, 3 -- reflex response values

0, 1, 2 -- Apgar values

Used for recording any clinical datum which is customarily recorded using symbolic values. Example: the results on a urinalysis strip, e.g. {neg, trace, +,++, +++} are used for leucocytes, protein, nitrites etc; for non-haemolysed blood {neg, trace, moderate}; for haemolysed blood {neg, trace,small, moderate, large}.

Inherits From: DvOrdered [27]

Abstract:False

Element Name	dv
Datatype	Integer [15]
minOccurs	0
maxOccurs	1
Note	Value in ordered enumeration of values. Any integer value can be used.

Element Name	symbol
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Coded textual representation of this value in the enumeration, which may be strings made from "+" symbols, or other enumerations of terms such as "mild", "moderate", "severe", or even the same number series as the values, e.g. "1", "2", "3".

DvQuantified

Abstract class defining the concept of true quantified values, i.e. values which are not only ordered, but which have a precise magnitude.

Inherits From: DvOrdered [27]
Abstract: True

Element Name	magnitude
Datatype	Real [15]
minOccurs	0
maxOccurs	1
Note	Numeric value of the quantity in canonical (i.e. single value) form. Implemented as constant, function or attribute in subtypes as appropriate. The type Ordered_numeric is mapped to the available appropriate type in each implementation technology.

Element Name	magnitude_status
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	Optional status of magnitude with values: • "=" : magnitude is a point value • "<" : value is < magnitude • ">" : value is > magnitude • "<=" : value is <= magnitude • ">=" : value is >= magnitude • "~" : value is approximately magnitude If not present, meaning is "=".

Element Name	error
Datatype	Integer [15]
minOccurs	1
maxOccurs	1
Note	Error margin of measurement, indicating error in the recording method or instrument (+/- %). Implemented in subtypes. A logical value of 0 indicates 100% accuracy, i.e. no error.

Element Name	accuracy
Datatype	Real [15]
minOccurs	1
maxOccurs	1
Note	Accuracy of the value in the magnitude attribute. 0% to +/- 100% A value of 0 means that the accuracy is unknown.

DvQuantity

Quantitified type representing “scientific” quantities, i.e. quantities expressed as a magnitude and units. Units were inspired by the Unified Code for Units of Measure (UCUM), developed by Gunther Schadow and Clement J. McDonald of The Regenstrief Institute. Can also be used for time durations, where it is more convenient to treat these as simply a number of individual seconds, minutes, hours, days, months, years, etc.

Inherits From: DvQuantified [29]
Abstract: False

Element Name	units
Datatype	DvString [17]
minOccurs	1
maxOccurs	1
Note	Stringified units, expressed in UCUM unit syntax, e.g. "kg/m2", “mm[Hg]", "ms-1", "km/h". http://unitsofmeasure.org/

DvCount

Countable quantities. Used for countable types such as pregnancies and steps (taken by a physiotherapy patient), number of cigarettes smoked in a day, etc.

Misuse: Not used for amounts of physical entities (which all have units)

Inherits From: DvQuantified [29]
Abstract: False

Element Name	count
Datatype	Integer [15]
minOccurs	1
maxOccurs	1
Note	Number of items counted.

DvRatio

Models a ratio of values, i.e. where the numerator and denominator are both pure numbers. Used for recording generic ratios that can not be cathegorized as rates or proportions, such as titers (e.g. 1:128), concentration ratios, e.g. Na:K (unitary denominator), albumin:creatinine ratio, even when presented as percentages, e.g. red cell distribution width (RDW). Should not be used to represent things like blood pressure which are often written using a '/' character, giving the misleading impression that the item is a ratio, when in fact it is a structured value. Similarly, visual acuity, often written as (e.g.) "6/24" in clinical notes is not a ratio but an ordinal (which includes non-numeric symbols like CF = count fingers etc).

Should not be used for formulations.

Rates and Proportions should use the DvRatio children classes, respectively DvRate [32] and DvProportion [33].

Inherits From: DvQuantified [29]
Abstract: False

Element Name	numerator
Datatype	Real [15]
minOccurs	1
maxOccurs	1
Note	numerator of ratio

Element Name	denominator
Datatype	Real [15]
minOccurs	1
maxOccurs	1
Note	denominator of ratio

Element Name	numerator_units
Datatype	DvString [17]
minOccurs	1
maxOccurs	1
Note	Stringified units, expressed in UCUM unit syntax, e.g. "kg/m2", “mm[Hg]", "ms-1", "km/h". http://unitsofmeasure.org/

Element Name	denominator_units
Datatype	DvString [17]
minOccurs	1
maxOccurs	1
Note	Stringified units, expressed in UCUM unit syntax, e.g. "kg/m2", “mm[Hg]", "ms-1", "km/h". http://unitsofmeasure.org/

DvRate

Models a ratio of values, i.e. where the numerator and denominator are both pure numbers, and the numerator is not contained (it is not a subset of the denominator). Example 1: Numerator = Number of episodes of seizures, Denominator = Number of days; Example 2 = Number of hospital admissions, Denominator = Number of bed-days. Should not be used to represent things like blood pressure which are often written using a ‘/’ character, giving the misleading impression that the item is a ratio, when in fact it is a structured value. Similarly, visual acuity, often written as (e.g.) “6/24” in clinical notes is not a ratio but an ordinal (which includes non-numeric symbols like CF = count fingers etc).

Should not be used for formulations.

Should not be used for generic Ratios (see DvRatio [31]) or for Proportions (see DvProportion[33])

Inherits From: DvRatio [31]
Abstract: False

Element Name	rate_type
Datatype	DvString [17]
minOccurs	1
maxOccurs	1
Note	Indicates semantic type of coefficient: pk_coefficient = coefficient type. Numerator and denominator may be any value. pk_unitary = Denominator must be 1. pk_per10^n = Denominator is 10^2, numerator is understood as a real number divided by an exponent of 10 (10^n). pk_fraction = Numerator and denominator are real numbers, allowing rational and irrational fractions, and the presentation method uses a slash, e.g. “1/2”, if the numerator is greater than the denominator, e.g. n=3, d=2, the presentation is “1 1/2”.

DvProportion

Models a ratio of values, i.e. where the numerator and denominator are both pure numbers, and the numerator is contained (it is a subset of the denominator). Example 1: Proportion of women submitted to hysterectomy; Example 2 = Cumulative Incidence. Should not be used to represent things like blood pressure which are often written using a ‘/’ character, giving the misleading impression that the item is a ratio, when in fact it is a structured value. Similarly, visual acuity, often written as (e.g.) “6/24” in clinical notes is not a ratio but an ordinal (which includes non-numeric symbols like CF = count fingers etc).

Should not be used for formulations.

Should not be used for generic Ratios (see DvRatio [31]) or for Rates (see DvRate [32])

Inherits From: DvRatio [31]
Abstract: False

Element Name	proportion_type
Datatype	DvString [17]
minOccurs	1
maxOccurs	1
Note	Indicates semantic type of proportion: pk_proportion = proportion type. Numerator and denominator may be any value. pk_unitary = Denominator must be 1. pk_per10^n = Denominator is 10^2, numerator is understood as a real number divided by an exponent of 10 (10^n). pk_fraction = Numerator and denominator are real numbers, allowing rational and irrational fractions, and the presentation method uses a slash, e.g. “1/2”, if the numerator is greater than the denominator, e.g. n=3, d=2, the presentation is “1 1/2”.

DvTemporal

Abstract class defining the concept of date and time types.

Inherits From: DvOrdered [27]
Abstract: True

DvDateTime

Represents an absolute point in time, specified to the second. Used for recording a precise point in real world time, and for approximate time stamps, e.g. the origin of a History in an Observation which is only partially known. All dates and times are assumed to be in the "current era", somewhere between 0001-01-01T00:00:00Z and 9999-12-31T23:59:59Z AD.

Inherits From: DvTemporal [34]
Abstract: False

Element Name	dv
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	The datetime value expressed as an ISO 8601 datetime string. If allow_partial is TRUE then partial entries are valid.

Element Name	allow_partial
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	If allow_partial is TRUE then partial entries are valid.

DvDate

Represents an absolute point in time, specified to the day.
Inherits From: DvTemporal [34]
Abstract: False

Element Name	dv
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	ISO8601:2004 date string

Element Name	allow_partial
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	If allow_partial is TRUE then partial entries are valid.

DvTime

Represents an absolute point in time, specified to the second.
Inherits From: DvTemporal [34]
Abstract: False

Element Name	dv
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	ISO8601:2004 time string

Element Name	allow_partial
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	If allow_partial is TRUE then partial entries are valid.

DvDuration

Durations are a component of time intervals and define the amount of intervening time in a time interval. They should only be used as part of a time interval as prescribed by the standard. Durations are represented by the format P[n]Y[n]M[n]DT[n]H[n]M[n]S or P[n]W as shown to the right. In these representations, the [n] is replaced by the value for each of the date and time elements that follow the [n]. Leading zeros are not required, but the maximum number of digits for each element should be agreed to by the communicating parties. The capital letters 'P', 'Y', 'M', 'W', 'D', 'T', 'H', 'M', and 'S' are designators for each of the date and time elements and are not replaced.

* P is the duration designator (historically called "period") placed at the start of the duration representation.
* Y is the year designator that follows the value for the number of years.
* M is the month designator that follows the value for the number of months.
* W is the week designator that follows the value for the number of weeks.
* D is the day designator that follows the value for the number of days.
* T is the time designator that precedes the time components of the representation.
* H is the hour designator that follows the value for the number of hours.
* M is the minute designator that follows the value for the number of minutes.
* S is the second designator that follows the value for the number of seconds.

For example, "P3Y6M4DT12H30M5S" represents a duration of "three years, six months, four days, twelve hours, thirty minutes, and five seconds". Date and time elements including their designator may be omitted if their value is zero, and lower order elements may also be omitted for reduced precision. For example, "P23DT23H" and "P4Y" are both acceptable duration representations. To resolve ambiguity, "P1M" is a one-month duration and "PT1M" is a one-minute duration (note the time designator, T, that precedes the time value). The smallest value used may also have a decimal fraction, as in "P0.5Y" to indicate half a year. This decimal fraction may be specified with either a comma or a full stop, as in "P0,5Y" or "P0.5Y". The standard does not prohibit date and time values in a duration representation from exceeding their "carry-over points" except as noted below. Thus, "PT36H" could be used as well as "P1DT12H" for representing the same duration. Alternately, a format for duration based on combined date and time representations may be used by agreement between the communicating parties either in the basic format PYYYYMMDDThhmmss or in the extended format P[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]. For example, the first duration shown above would be "P0003-06-04T12:30:05". However, individual date and time values cannot exceed their moduli (e.g. a value of 13 for the month or 25 for the hour would not be permissible).

Inherits From: DvTemporal [36]

Abstract: False

Element Name	dv
Datatype	String [11]
minOccurs	0
maxOccurs	1
Note	The duration in the form of a string.

Common

Locatable

Root class of all information model classes that can be located in a constraint model.

Inherits From: DvAny [16]
Abstract: True

Element Name	feeder_audit
Datatype	FeederAudit [40]
minOccurs	0
maxOccurs	1
Note	Audit trail from non-MLHIM system of original commit of information forming the content of this node, or from a conversion gateway which has synthesized this node.

Definition

Root class of all information model classes that can be expressed as a definition in a constraint model.

Inherits From: Locatable [38]
Abstract: True

FeederAuditDetails

Audit details for any system in a feeder system chain. Audit details here means the general notion of who/where/when the information item to which the audit is attached was created. None of the attributes is defined as mandatory, however, in different scenarios, various combinations of attributes will usually be mandatory. This can be controlled by specifying feeder audit details in legacy CCDs.

Inherits From: DvAny [16]
Abstract: False

Element Name	system_id
Datatype	DvIdentifier [19]
minOccurs	1
maxOccurs	1
Note	Identifier of the system which handled the information item.

Element Name	version_id
Datatype	DvString [17]
minOccurs	0
maxOccurs	1
Note	Any identifier used in the system such as “interim”, “final”, or numeric versions if available.

Element Name	provider
Datatype	PartyIdentified [44]
minOccurs	0
maxOccurs	1
Note	Optional provider(s) who created, committed, forwarded or otherwise handled the item.

Element Name	location
Datatype	Item [49]
minOccurs	0
maxOccurs	1
Note	Identifier of the particular site/facility within an organization which handled the item. For computability, this identifier needs to be e.g. a PKI identifier which can be included in the identifier list of the PartyIdentified object.

Element Name	time
Datatype	DvTemporal [34]
minOccurs	0
maxOccurs	1
Note	Time of handling the item. For an originating system, this will be time of creation, for an intermediate feeder system, this will be a time of accession or other time of handling, where available.

Element Name	subject
Datatype	PartyProxy [44]
minOccurs	0
maxOccurs	unbounded
Note	Subject(s) of the received information item.

FeederAudit

Audit and other meta-data for systems in the feeder chain.
Inherits From: xs:anyType
Abstract: False

Element Name	originating_system_audit
Datatype	FeederAuditDetails [38]
minOccurs	1
maxOccurs	1
Note	Any audit information for the information item from the originating system.

Element Name	originating_system_item_ids
Datatype	DvIdentifier [19]
minOccurs	1
maxOccurs	unbounded
Note	Identifiers used for the item in the originating system, e.g. filler and placer ids.

Element Name	feeder_system_audit
Datatype	FeederAuditDetails [38]
minOccurs	0
maxOccurs	1
Note	Any audit information for the information item from the feeder system, if different from the originating system.

Element Name	feeder_system_ids
Datatype	DvIdentifier [19]
minOccurs	0
maxOccurs	unbounded
Note	Identifiers used for the item in the feeder system, where the feeder system is distinct from the originating system.

Element Name	original_content
Datatype	DvEncapsulated [22]
minOccurs	1
maxOccurs	1
Note	Optional inline inclusion of or reference to original content corresponding to the MLHIM content at this node. Typically a URI reference to a document or message in a persistent store associated with the EHR.

Attestation

Record an attestation by a party of item(s) of record content. The type of attestation is recorded by the reason attribute, which may be coded.

Inherits From: mlhim2:Locatable
Abstract: False

Element Name	attested_view
Datatype	DvMedia [23]
minOccurs	0
maxOccurs	1
Note	Optional visual representation of content attested e.g. screen image.

Element Name	proof
Datatype	DvParsable [22]
minOccurs	1
maxOccurs	1
Note	Proof of attestation such as an GPG signature.

Element Name	reason
Datatype	DvCodedString [18]
minOccurs	1
maxOccurs	1
Note	Reason of this attestation. Optionally coded by the MLHIM Terminology group “attestation reason”, includes values like “authorization”, “witness” etc.

Element Name	commiter
Datatype	PartyProxy [44]
minOccurs	1
maxOccurs	1
Note	Identity and optional reference into identity management service, of user who committed the item.

Element Name	time_commited
Datatype	DvTemporal [34]
minOccurs	1
maxOccurs	1
Note	Time of committal of the item.

Element Name	is_pending
Datatype	Boolean [11]
minOccurs	1
maxOccurs	1
Note	True if this attestation is outstanding. False means it has been completed.

Participation

Model of a participation of a Party (any Actor or Role) in an activity. Used to represent any participation of a Party in some activity, which is not explicitly in the model, e.g. assisting nurse. Can be used to record past or future participations. Should not be used in place of more permanent relationships between demographic entities.

Inherits From: Any [11]

Abstract: False

Element Name	performer
Datatype	PartyProxy [44]
minOccurs	1
maxOccurs	1
Note	The id and possibly demographic system link of the party participating in the activity.

Element Name	function
Datatype	DvCodedString [18]
minOccurs	1
maxOccurs	1
Note	The function of the Party in this participation (note that a given party might participate in more than one way in a particular activity). This attribute should be coded.

Element Name	mode
Datatype	DvCodedString [18]
minOccurs	1
maxOccurs	1
Note	The mode of the performer / activity interaction, e.g. present, by telephone, by email etc.

Element Name	time
Datatype	DvTemporal [34]
minOccurs	1
maxOccurs	1
Note	The time interval during which the participation took place, if it is used in an observational context (i.e. recording facts about the past) or the intended time interval of the participation when used in future contexts, such as EHR Instructions.

PartyProxy

Abstract concept of a proxy description of a party, including an optional link to data for this party in a demographic or other identity management system. Sub-typed into PartyIdentified and PartySelf.

Inherits From: Locatable [38]
Abstract: True

Element Name	external_ref
Datatype	DvURI [21]
minOccurs	0
maxOccurs	1
Note	Optional reference to more detailed demographic or identification information for this party, in an external system.

PartySelf

Party proxy representing the subject of the record. Used to indicate that the party is the owner of the record. May or may not have external_ref set.
Inherits From: PartyProxy [44]
Abstract: False

PartyIdentified

Proxy data for an identified party other than the subject of the record, minimally consisting of human-readable identifier(s), such as name, formal (and possibly computable) identifiers such as NHS number, and an optional link to external data.

There must be at least one of name, identifier or external_ref present.Used to describe parties where only identifiers may be known, and there is no entry at all in the demographic system (or even no demographic system). Typically for health care providers, e.g. name and provider number of an institution. Should not be used to include patient identifying information.

Inherits From: PartyProxy [44]
Abstract: False

Element Name	name
Datatype	DvString [17]
minOccurs	0
maxOccurs	1
Note	Optional human-readable name (in String form).

Element Name	identifiers
Datatype	List [11]<DvIdentifier [19]>
minOccurs	0
maxOccurs	1
Note	List of DvIdentifiers - One or more formal identifiers (possibly computable).

ExceptionalValue

Subclasses are used to indicate why a value is missing (Null) or is outside a measurable range.

Inherits From: Any [11]
Abstract: True

Element Name	ev_name
Datatype	String [11]
minOccurs	1
maxOccurs	1
Note	The class names are abbreviations according to the ISO 21090 Null Flavours. This attribute is the full name.

Element Name	ev_meaning
Datatype	String [11]
minOccurs	1
maxOccurs	1
Note	The descriptive meaning/usage for the class/complexType.

NI

The value is exceptional (missing, incomplete, improper). No information is available as to the reason for being an exceptional value is provided. This is the most general and default value.

Inherits From: ExceptionalValue [45]
Abstract: False

MSK

There is information on this item available but it has not been provided by the sender due to security, privacy or other reasons. There may be an alternate method of obtaining the information.

Inherits From: NI
Abstract: False

NA

No proper value is applicable in this context e.g.,the number of cigarettes smoked per day by a non-smoker subject.

Inherits From: NI
Abstract: False

UNK

A proper value is applicable, but not known.

Inherits From: NI
Abstract: False

INV

The value as represented in the instance is not a member of the set of permitted data values in the constrained value domain of a variable.

Inherits From: NI
Abstract: False

DER

An actual value may exist, but it must be derived from the provided information, usually an expression is provided directly.

Inherits From: INV
Abstract: False

UNC

No attempt has been made to encode the information correctly but the raw source information is represented, usually in free text.

Inherits From: INV
Abstract: False

OTH

The actual value is not a member of the permitted data values in the variable. (e.g., when the value of the variable is not by the coding system)

Inherits From: INV
Abstract: False

ASKU

Information was sought but not found (e.g., patient was asked but did not know).

Inherits From: UNK
Abstract: False

ASKR

Information was sought but refused to be provided (e.g., patient was asked but refused to answer). This element is not part of ISO 21090 but was added in MLHIM2 to provide better coverage of missing values.

Inherits From: UNK
Abstract: False

NASK

This information has not been sought (e.g., patient was not asked).

Inherits From: UNK
Abstract: False

QS

The specific quantity is not known, but is known to non-zero and it is not specified because it makes up the bulk of the material; "Add 10mg of ingredient X, 50mg of ingredient Y and sufficient quantity of water to 100mL."

Inherits From: UNK
Abstract: False

TRC

The content is greater or less than zero but too small to be quantified.

Inherits From: UNK
Abstract: False

NINF

Negative infinity of numbers

Inherits From: OTH
Abstract: False

PINF

Positive infinity of numbers

Inherits From: OTH
Abstract: False

NAV

Information is unavailable at this time but is expected that it will be available later.
Abstract: False
Inherits From: ASKU

Structures

Item

The abstract parent of Slot, Cluster and Element representation classes.

Inherits From: Definition [38]
Abstract: True

Slot

A structure allowing the inclusion of one CCD inside a CCD. An unbounded list of allowable CCDs to choose from should be available at runtime.

Inherits From: Item [49]
Abstract: False

Element Name	ccd
Datatype	DvString [17]
minOccurs	1
maxOccurs	1
Note	The CCD ID selected at run-time.

Element Name	allowed_ccds
Datatype	Set [11]<DvString [17]>
minOccurs	1
maxOccurs	1
Note	A set of allowed CCD IDs of which one is selected at runtime.

Element

The leaf variant of Item, to which a DvAny instance is attached.

Inherits From: mlhim2:Item
Abstract: False

Element Name	dv
Datatype	DvAny [16]
minOccurs	1
maxOccurs	1
Note	Data value of this leaf

Cluster

The grouping variant of Item, which may contain further instances of Item, in an ordered list.

Inherits From: Item [49]
Abstract: False

Element Name	items
Datatype	List [11]<Item [49]>
minOccurs	1
maxOccurs	1
Note	List of Items.

Content

Entry

The abstract parent of all ENTRY subtypes. An ENTRY is the root of a logical item of "hard" clinical information created in the "clinical statement" context, within a clinical session. There can be numerous such contexts in a clinical session. Observations and other Entry types only ever document information captured/created in the event documented by the enclosing Composition. An Entry is also the minimal unit of information any query should return, since a whole Entry (including sub-parts) records spatial structure, timing information, and contextual information, as well as the subject and generator of the information.

Inherits From: Locatable [38]
Abstract: True

Element Name	language
Datatype	DvCodedString [18]
minOccurs	1
maxOccurs	1
Note	Mandatory indicator of the localised language in which this Entry is written.

Element Name	encoding
Datatype	DvCodedString [18]
minOccurs	0
maxOccurs	1
Note	Name of character set in which text values in this Entry are encoded.

Element Name	subject
Datatype	PartyProxy [44]
minOccurs	1
maxOccurs	1
Note	Id of human subject of this Entry, e.g.: organ donor, foetus, family member, another clinically relevant person.

Element Name	provider
Datatype	PartyProxy [44]
minOccurs	0
maxOccurs	1
Note	Optional identification of provider of the information in this Entry, which might be: the patient, a patient agent, e.g. parent, guardian, the clinician, a device or software. Generally only used when the recorder needs to make it explicit. Otherwise, Composition composer and other participants are assumed.

Element Name	other_participations
Datatype	List [11]<Participation [42]>
minOccurs	0
maxOccurs	1
Note	Other participations at Entry level.

Element Name	workflow_id
Datatype	String [11]
minOccurs	
maxOccurs	
Note	Identifier of externally held workflow engine data for this workflow execution, for this subject of care.

AdminEntry

Entry subtype for administrative information, i.e. information about setting up the clinical process, but not itself clinically relevant. CCDs will define contained information. Used for admistrative details of admission, episode, ward location, discharge, appointment (if not stored in a practice management or appointments system). Not used for any clinically significant or demographic information.

Inherits From: Entry [51]
Abstract: False

Element Name	data
Datatype	Item [49]
minOccurs	1
maxOccurs	1
Note	The data of the Entry.

DemographicEntry

Entry subtype for demographic information. CCDs will define contained information. Used for demographic details of the subject of care such as name, address, sex, birth data, death data. Not used for any clinically significant or administrative information.

Inherits From: Entry [51]
Abstract: False

Element Name	data
Datatype	Item [49]
minOccurs	1
maxOccurs	1
Note	The data of the Entry.

CareEntry

The abstract parent of all clinical Entry subtypes. Defines protocol and guideline attributes for all clinical Entry subtypes. Not used for any demographic or administrative information.

Inherits From: Entry [51]
Abstract: True

Element Name	protocol
Datatype	Item [49]
minOccurs	0
maxOccurs	1
Note	Description of the method (i.e. how) the information in this entry was arrived at. For Observations, this is a description of the method or instrument used. For Evaluations, how the evaluation was arrived at. For Instructions, how to execute the Instruction. This may take the form of references to guidelines, including manually followed and executable, knowledge references such as a paper in Medline, clinical reasons within a larger care process.<

Element Name	guideline_id
Datatype	DvIdentifier [19]
minOccurs	0
maxOccurs	1
Note	Optional external identifier of guideline creating this action if relevant.

Constraint

CCD

Concept Constraint Definition

Inherits From: Any [11]
Abstract: False

Element Name	definition
Datatype	Definition [38]
minOccurs	1
maxOccurs	1
Note	Contains one Definition entry as the root term for this CCD.

Constraint Definitions

A Constraint Definition Designer (CDD) is being developed. The CDD can be used now to create a shell XSD with the correct metadata entries. Each release is available in the Tools section of HKCR.net.

There is also a conceptual model of the information using the mind mapping software, XMind. It provides domain experts a copy/paste method of building up the structures to define a certain concept.

Healthcare Knowledge Component Repository

Managing CCDs

An open source content management system, the Healthcare Knowledge Component Repository (HKCR) is being deployed to provide an easy path for the development and distribution of CCDs on a global basis. See the HKCR documentation for more information.

OSHIP

The Open Source Health Information Platform (OSHIP) is a generic acronym for all implementations of the the MLHIM reference model in all programming languages. The basic concept is to supply a common information model for all healthcare applications, irregardless of the implementation language. For most implementation languages it is suggested that they use available tools and create bindings to the XML Schema provided as the reference implementation.

OSHIPpy

The MLHIM Reference Model implementation in Python. The reference model has been generated using generateDS. It is expected that implementers using Plone, Grok, Django, Web2py, etc. will create apps based on the reference model.

OSHIPjava

The MLHIM Reference Model implementation in Java. (pending implementation; taking volunteers)

OSHIPrb

The MLHIM Reference Model implementation in Ruby. (pending implementation; taking volunteers)

OSHIPcpp

The MLHIM Reference Model implementation in C++. (pending implementation; taking volunteers)

OSHIPlua

The MLHIM Reference Model implementation in Lua. (pending implementation; taking volunteers)