HL7_CDS_VMR_XML4VMR_R1_D1_2014JAN

HL7 Virtual Medical Record for Clinical Decision Support (vMR-CDS) XML Specification, Release 1

March 2014

HL7 DSTU Specification

Sponsored by:
Implementable Technology Specifications Work Group
Clinical Decision Support Work Group

Project Coordinator and Document Editor
Kensaku Kawamoto, MD, PhD, University of Utah
Claude Nanjo, MPH, MAAS, Cognitive Medical Systems

Collaborators
David Shields, University of Utah
Victor Lee, MD, Zynx Health Incorporated
Aziz Boxwala, MD, PhD, FACMI, Meliorix Inc
Mark Roche, MD, MSMI, Roche Consulting
Bryn Rhodes, Veracity Solutions
Davide Sottara, PhD, Arizona State University
Andrew K. McIntyre, FRACP, MBBS, Medical-Objects
Yongjian Bao, PhD, GE Healthcare
Howard R. Strasberg, MD, MS, Wolters Kluwer Health
Peter R. Tattam, Tattam Software Enterprises Pty Ltd
Scott Bolte, MS, GE Healthcare
Peter Scott, MBBS, Medical-Objects
Keith Boone, GE Healthcare
Zhijing Liu, PhD, Siemens Healthcare
Chris Melo, Philips Healthcare
Nathan Hulse, PhD, Intermountain Healthcare
Jim Basilakis, MBBS, MS, University of Western Sydney
Robert Worden, Open Mapping Software, Limited
Daryl Chertcoff, HLN Consulting
Clayton Curtis, MD, PhD, U.S. Veterans Health Administration
Guilherme Del Fiol, MD, PhD, University of Utah
Emory Fry, MD, Uniformed Service University Health Sciences
Jean-Charles Dufour, MD, PhD, Université Aix-Marseille
Laurent CHARLOIS, Université de la Méditerranée

Project Sponsor: HL7 Implementable Technology Specifications Work Group
Co-Sponsor: HL7 Clinical Decision Support Work Group

HL7 Project #1016
Universal Realm Draft Standard for Trial Use Specification

Identifying Information for Specification:
Specification Name and Release Number: HL7 Virtual Medical Record for Clinical Decision Support (vMR-CDS) XML Specification, Release 1
Realm: Universal
Ballot Level: Draft Standard for Trial Use
Ballot Cycle: January 2014
Specification Date: March 2014
Version Number within Release 1: 3.0

Ballot History:
Version 1.0 of the Release 1 specification was successfully balloted as a DSTU in May 2013. Version 2.0 of the Release 1 specification was successfully balloted as an informative specification in September 2013. This specification represents Version 3.0 of the Release 1 specification.

Note Regarding Changes since Last Version:
This specification has been updated to reflect changes to the underlying logical model. The intent of this ballot is to transition this specification to a universal draft standard for trial use (DSTU).

Acknowledgments:
The authors also wish to acknowledge members of the **HL7 Technical Steering Committee** and its Task Force on CDS specifications related to the U.S. Standards and Interoperability Framework's Health eDecisions initiative (www.healthedecisions.org). These individuals have provided significant guidance on the direction and content of this specification.

Name	Organization
Austin Kreisler	Science Applications International Corporation (SAIC)
Anthony Julian	Mayo Clinic
Calvin Beebe	Mayo Clinic
Dale Nelson	Lantana Consulting Group
Jean-Henri Duteau	Duteau Design Inc.
John Quinn	Health Level 7 International
Kai Heitmann	Heitmann Consulting and Services
Keith Boone	GE Healthcare
Ken McCaslin	Quest Diagnostics, Incorporated
Ken Rubin	HP Enterprise Services
Lloyd McKenzie	Gordon Point Informatics Ltd.
Lorraine Constable	Constable Consulting Inc.
Lynn Laasko	Health Level 7 International
Patricia Van Dyke	Moda Health
Paul Knapp	Knapp Consulting Inc.
Ron Parker	Canada Health Infoway
Woody Beeler	Beeler Consulting LLC

We would also like to acknowledge the invaluable contributions from other HL7 Work Groups including Patient Care, Pharmacy, and Nutrition.

Table of Contents

1 Executive Summary

A Virtual Medical Record (vMR) for Clinical Decision Support (CDS) is a data model for representing the data that are analyzed and/or produced by CDS engines. The purpose of the vMR effort is to define a standard vMR that (i) can be used across CDS implementations and (ii) is simple and intuitive for a typical CDS knowledge engineer to understand, use, and implement.

The vMR XML Specification provides guidance on how to implement the semantics of the vMR Logical Model in an XML format. This specification is informed by the requirements and pilot activities of the Standards and Interoperability Framework's Health eDecisions initiative, which is using the vMR as a core underlying information model.

As specified in Section 2.1, conformant XML instances must validate against the accompanying XML schemas.

This document is intended to document the vMR XML Schema by providing a high-level overview of its key components. All accompanying examples and diagrams shall be considered supplemental in nature. The XML schemas are based on the vMR Logical Model Release 2, Version 3.0.

2 XML Specification for VMR

2.1 Conformance Statement

Conformant XML instances must validate against the accompanying XML schemas. Validation against the XML schemas is a necessary but not sufficient condition for a vMR represented in this XML format to be considered valid and conformant, as additional requirements may be specified in the comments in the vMR XML Schema. Of note, these comments in the XML Schema are faithful renditions of the text in the Logical Model. Additional conformance requirements may be specified in vMR templates. To summarize, valid and conformant instances are valid to the schema and any additional requirements specified in the schema comments and/or templates. The content of this paragraph is the sole normative section of the specification.

2.2 Overview

The vMR XML specification consists of 6 XSD schema files which follow a similar conceptual categorization compared to the categorization defined in the vMR Logical Model Release 2, Version 3.0:

1. ***datatypes.xsd*** – A constrained and relevant set of HL7 V3 R2 datatypes for CDS.
2. ***vmr.xsd*** – Main clinical concept types for CDS.
3. ***cdsInput.xsd*** - Input data for a CDS system. For example, the primary input data payload for a CDS guidance service compliant with the HL7 Decision Support Service standard.
4. ***cdsInputSpecification.xsd*** – Concept types relevant to specifying required CDS input, for example the CDS input used by a CDS guidance service compliant with the HL7 Decision Support Service standard.
5. c***dsOutput.xsd –*** Output data for a CDS system. For example, the primary output data payload for a CDS guidance service compliant with the HL7 Decision Support Service standard.
6. ***cdsOutputspecification.xsd*** – Concept types relevant to specifying details on the CDS output, for example the CDS output that will be returned by a CDS guidance service compliant with the HL7 Decision Support Service standard.

For a more in-depth description of the concepts represented by each schema above, please refer to the annotations provided within each schema and/or to the HL7 Virtual Medical Record for Clinical Decision Support (vMR-CDS) Logical Model, Release 2 specification. Also, the HL7 Decision Support Service Implementation Guide, Release 1 provides a concrete example of how CDS inputs, CDS input specifications, CDS outputs, and CDS output specifications can be leveraged to deliver CDS.

These schemas were **auto-generated from the vMR Logical Model** using the Enterprise Architect tool, with minimal post-processing as noted in the README accompanying the schemas. The reader is referred to the **vMR Logical Model specification for more detailed information on the semantics of the vMR, including guidance on how to**

represent common clinical concepts using the vMR and an example vMR derived from a Consolidated Clinical Document Architecture (CCDA) document.

Diagrams have been added in the document for illustrative purposes only.

2.3 Namespaces, Schema Dependencies, and Design Approaches

Each schema defines its collection of elements and types within its own namespace. The following table lists the schema namespaces.

Schema	Namespace
datatypes.xsd	urn:hl7-org:cdsdt:r2
vmr.xsd	urn:hl7-org:vmr:r2
cdsInput.xsd	urn:hl7-org:cdsinput:r2
cdsOutput.xsd	urn:hl7-org:cdsoutput:r2
cdsInputSpecification.xsd	urn:hl7-org:cdsinputspecification:r2
cdsOutputSpecification.xsd	urn:hl7-org:cdsoutputspecification:r2

Figure 1 - Schema Namespaces

A schema may import one or more other schemas residing in a different namespace. In that situation, a namespace prefix is associated with the imported schema's namespace and elements from the imported schema are prefixed accordingly. For instance, the vmr.xsd schema imports the datatype.xsd schema, and it prefixes any type or element in that schema with the prefix 'dt:'. The following diagram illustrates the 'import' relationship between the schemas.

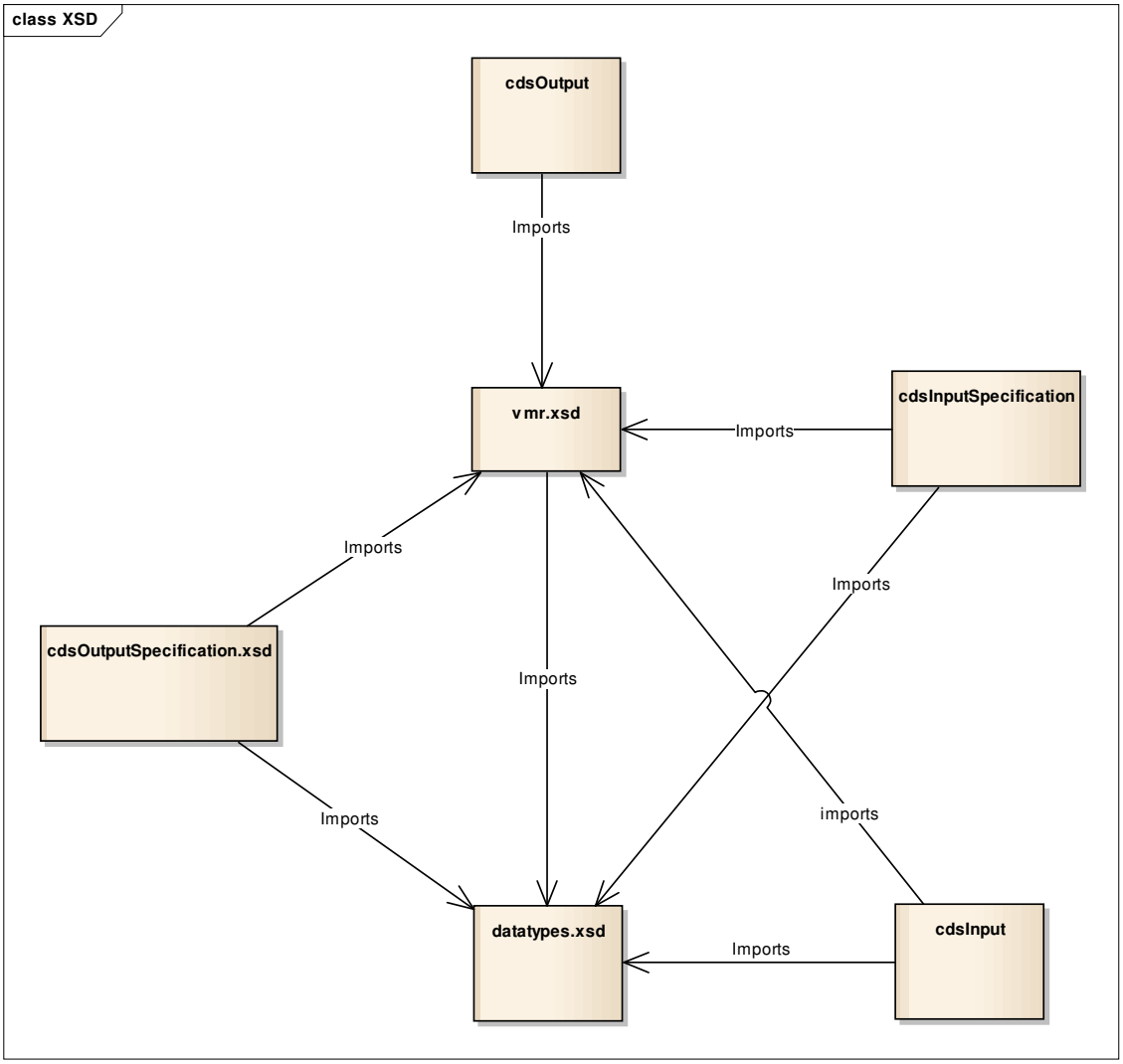


Figure 2 - Schema import relationships

This specification makes use of the XML Schema Instance Type (xsi:type) pattern in order to express polymorphism. For instance, a given patient (type EvaluatedPerson in the vMR) may have any number of associated clinical statements. The vMR schema represents this collection as a sequence of clinicalStatement elements of cardinality 0..* and of type 'ClinicalStatement'. The type ClinicalStatement is the super class for a number of specializations such as AdverseEvent, ObservationResult, ProcedureProposal, and so on. When creating an instance of that schema, one may then specify the attribute xsi:type to declare the formal type of the clinical statement under consideration as shown below:

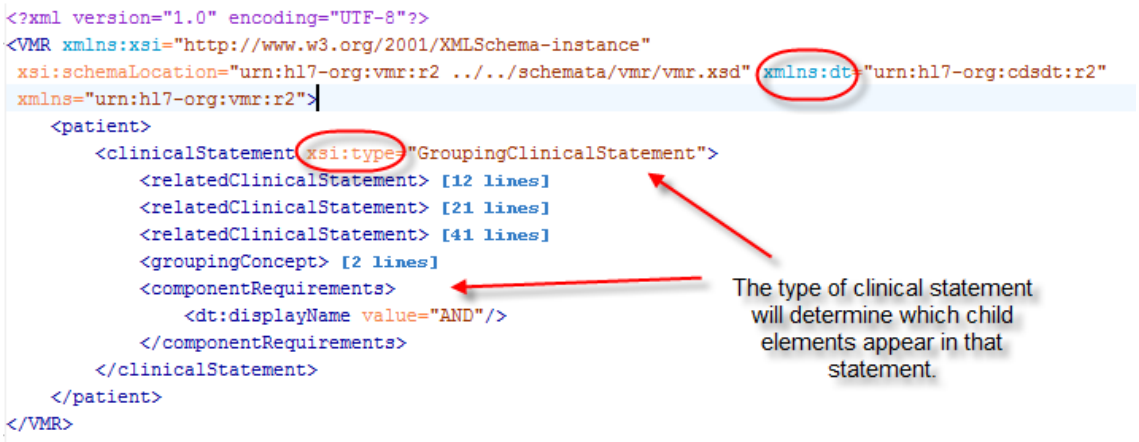
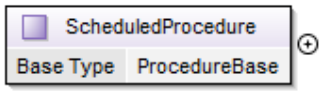


Figure 3 - Example use of xsi:type in an XML instance

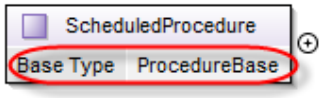
2.4 Quick Guide to the Diagrams

The following section provides a guide to understanding the diagrams presented in the rest of this document.

A type is represented by a box:



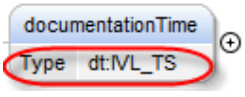
If a type is derived from another type, it is denoted in the diagram by the prefix 'Base Type':



Here, the ScheduledProcedure complexType is a specialization of the base type ProcedureBase.

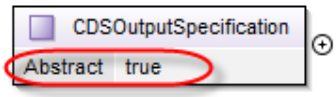
An element is represented by a rounded rectangle:

The type of the element is indicated below the element's name:



In this case, the element documentationTime is of type IVL_TS in the 'dt' namespace, the namespace prefix for the common vMR datatypes.

An abstract complex type is annotated with the 'Abstract' key word as shown below. It cannot be instantiated directly:



2.5 datatypes.xsd

The vMR's core datatypes are a constrained subset of the V3 R2 datatypes relevant for clinical decision support. These data types represent fundamental fine-grained clinical concepts including most primitive types such as Boolean (BL) and integer (INT) as well as composite types such as a physical quantity (PQ) or a time interval (IVL_TS). They are essential to ensure that models in the V3 family of models share common granularity with regards to these fundamental types. The datatype.xsd schema defines the serialization of the base vMR data types and any XML instance that validates against this schema shall validate against the HL7 V3 R2 schema as well. Please see the vMR Logical Model for further details.

This schema is utilized by all other schemas.

2.6 vmr.xsd

This schema specifies information about a patient relevant for CDS. Note that, associated with each evaluated person, such as a patient, is a set of clinical statements and demographic information about this person. An evaluated person may be associated with other entities such as people or facilities. Also note that clinical statements may be related to other clinical statements such as one or more ObservationResults tied to an AdverseEvent.

The vMR schema also allows for the addition of new attributes to clinical statements and entities using a name-value pair extension mechanism.

This schema is imported by the cdsInput.xsd, cdsInputSpecification.xsd, cdsOutput.xsd and cdsOutputSpecification.xsd schemas. The key components of the vmr.xsd schema are shown below. Please refer to the schema for the actual specification.

2.7 VMR Root Type

The VMR type specifies the root of the virtual medical record for CDS. It can be optionally constrained by a template and must specify the patient to whom this record applies. Note that in the diagram below, the patient element is of type EvaluatedPerson and will thus possess all elements specified by that type.

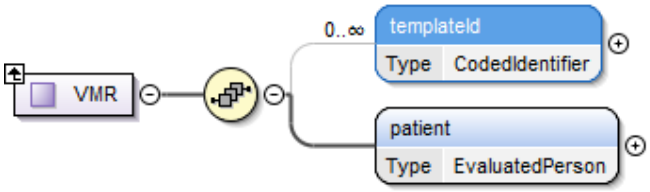


Figure 4 - The VMR complex type

The evaluated person, typically a patient, is represented by the complex type EvaluatedPerson. It inherits its attributes from the Person class and adds several demographic elements and attributes of its own. It also defines two very important relationships:

1. An evaluated person may have a number of associated clinical statements that define his or her clinical history.
2. An evaluated person may be associated with a number of related evaluated persons such as a family member.

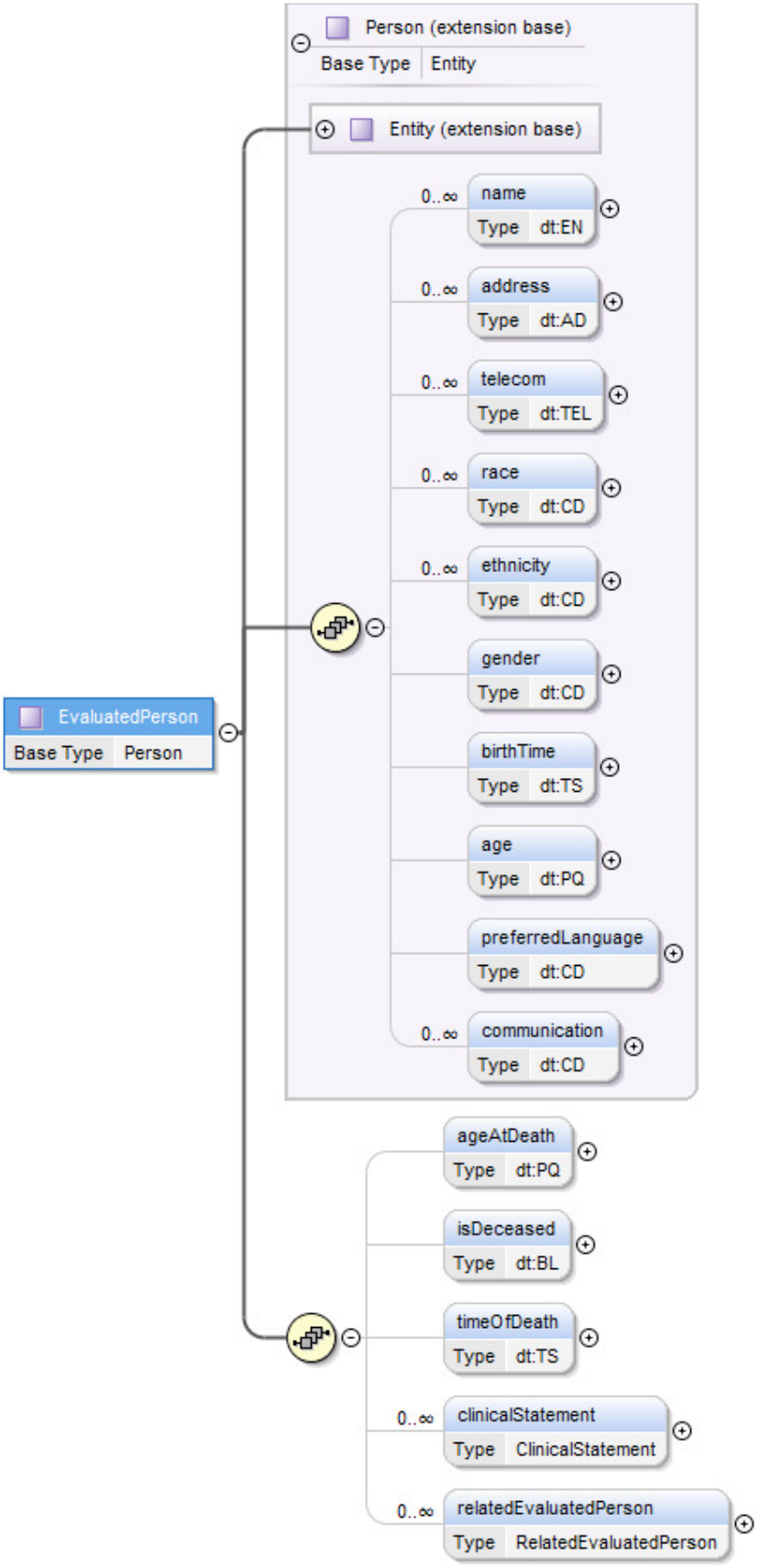


Figure 5 - EvaluatedPerson

Other key concepts in the vMR are the ClinicalStatement and Entity classes. The patient record for Clinical Decision Support is essentially a collection of statements made about the patient as well as a collection of related entities that define the patient and clinical context. The next two diagrams present these two concepts as they are represented in the vMR XSD.

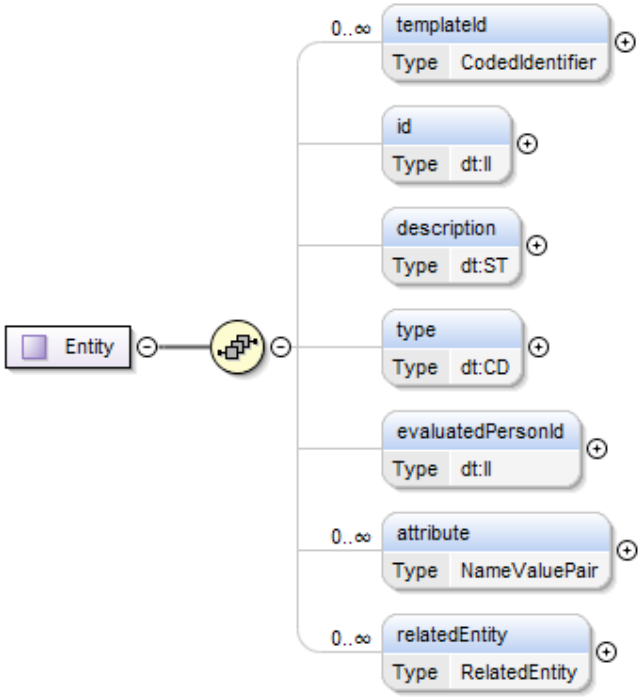


Figure 6 - The Entity complex type

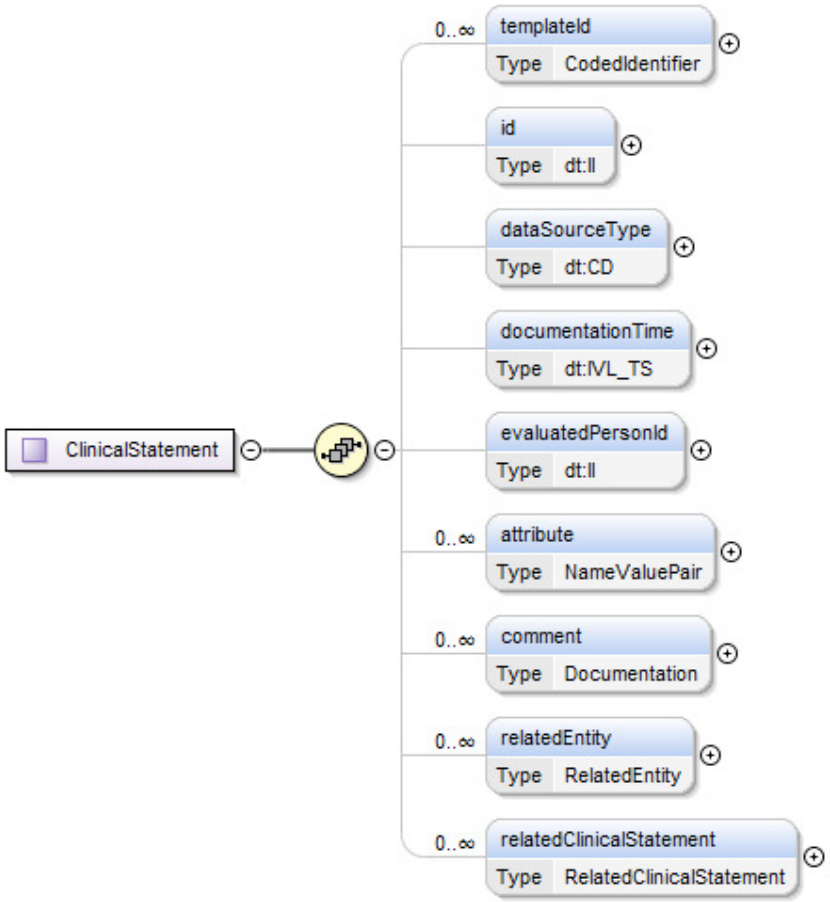


Figure 7 - The ClinicalStatement complex type

The vMR is a highly flexible model which supports a number of extension mechanisms. These fall under two main categories:

1. Defining new relationships
    a. Related entities
    b. Related clinical statements
2. Defining new attributes

Note that while the vMR is extensible, extensions, when defined, should be informed and constrained by templates based on the clinical use case and domain. Templates can be specified by providing the appropriate template reference information in the templateId attribute for those complex types such as Entity and ClinicalStatement, both of which support this attribute.

Relations in the vMR follow a common pattern and can be thought of as a Subject – Predicate – Object relationship or a triple. The type containing the relationship is the subject. The targetRole element of the relationship specifies the predicate. The target object of the relationship is the object of the triple. While any ad-hoc relationship can be constructed in this fashion, in order for these relationships to be interoperable, these must be defined in a template.

A RelatedEntity relationship specifies a predicate with Entity as its range (the target of the relationship). This means that any specialization of the Entity class is a valid target

of the relationship. However, as stated above, these targets can be constrained by templates.

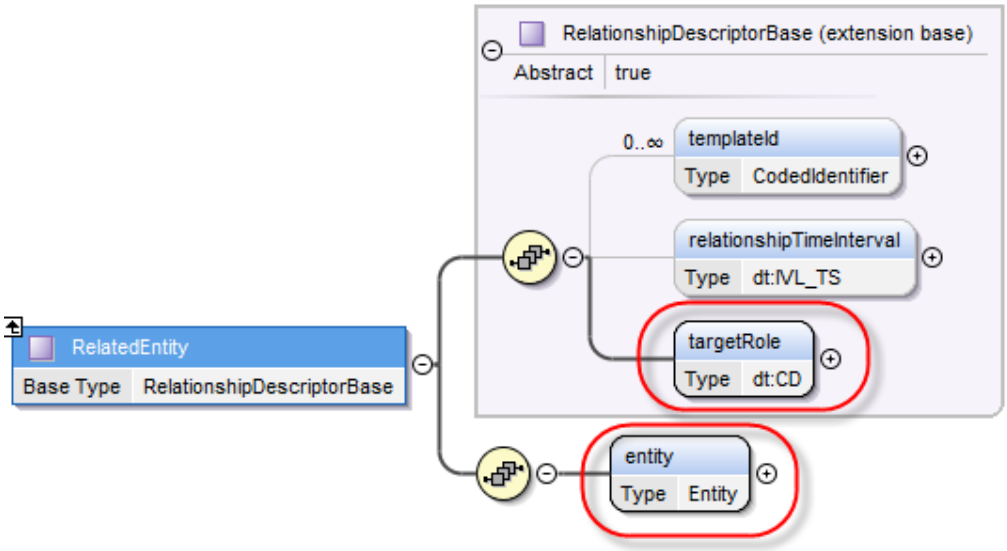


Figure 8 - RelatedEntity relates a target entity to a source entity or clinical statement

A RelatedClinicalStatement relationship specifies a predicate with ClinicalStatement objects as its range. This means that any specialization of the ClinicalStatement class is a valid target of the relationship. However, as stated above, these targets can be constrained by templates.

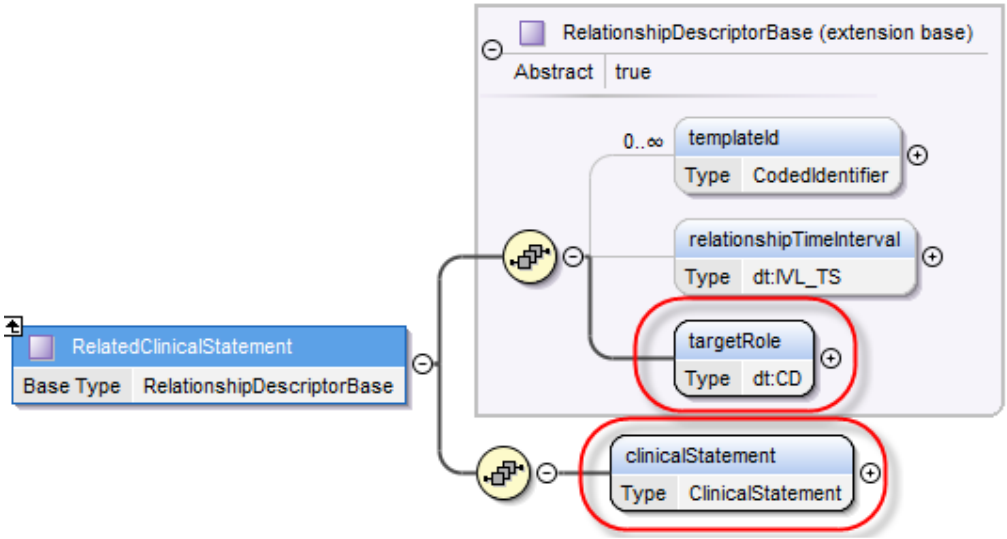


Figure 9 - RelatedClinicalStatement relates a target clinical statement to a source clinical statement

New attributes can be introduced to the Entity, ClinicalStatement, and ExtendedVmrTypeBase classes using the NameValuePair type. Entity, Clinical Statement, and the ExtendedVmrTypeBase specify an “attribute” element of multiple cardinality (0..*) to support such an extension mechanism. The name of the attribute is specified as a string, generally a token with no spaces such as ‘frequencyOfAdministration’. A concept descriptor (CD) type can be used to associate

this extension attribute with a semantic category (using the semanticCode field) chosen from a controlled terminology. The value of the attribute is of type ExtendedVmrTypeBase, an abstract class with an expressive number of specializations. This allows the value of an attribute to be any of the vMR datatypes or the higher-level types defined in the vMR such as NutrientModification for instance.

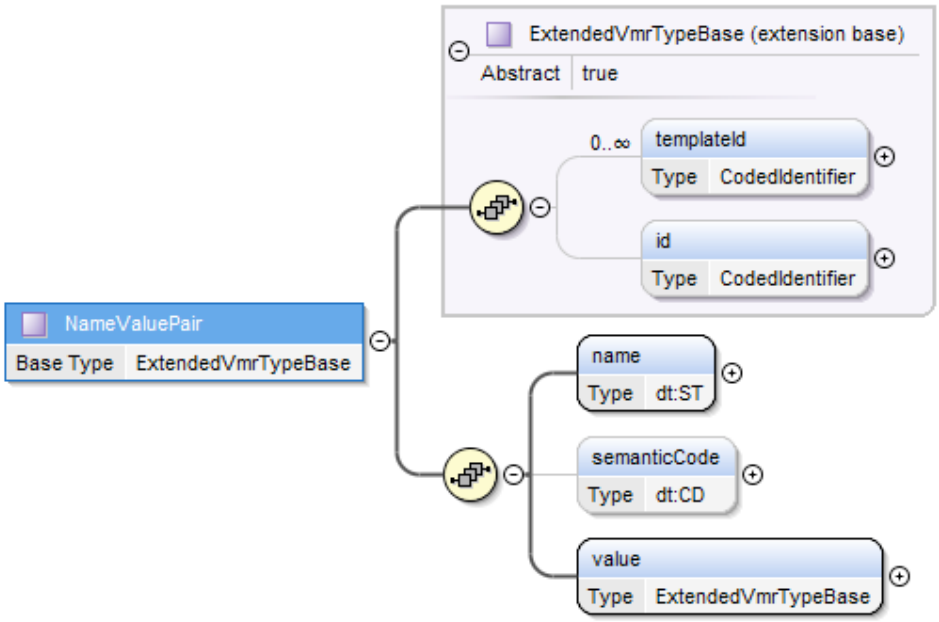


Figure 10 - Name-Value-Pair extension mechanism for clinical statement subclasses and entities

The following three diagrams illustrate how the attribute element can be used to extend an Entity, a ClinicalStatement, and an ExtendedVmrTypeBase. At this time, these three classes (and any specialization thereof) are the only vMR classes that can be extended in this manner.

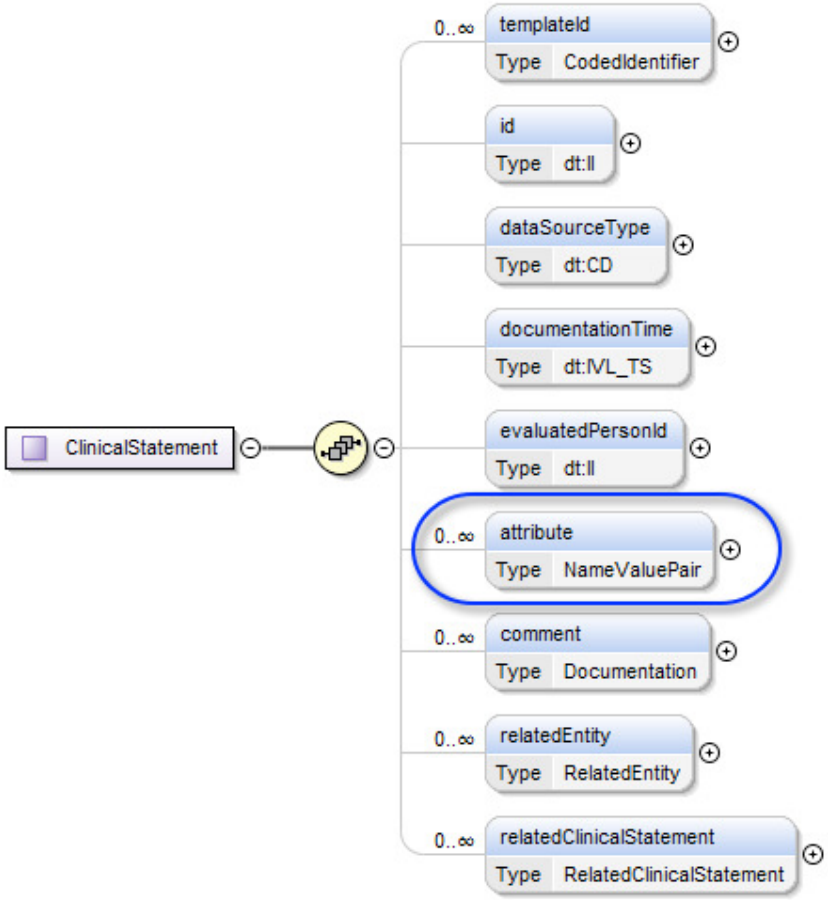


Figure 11 - Extending a clinical statement

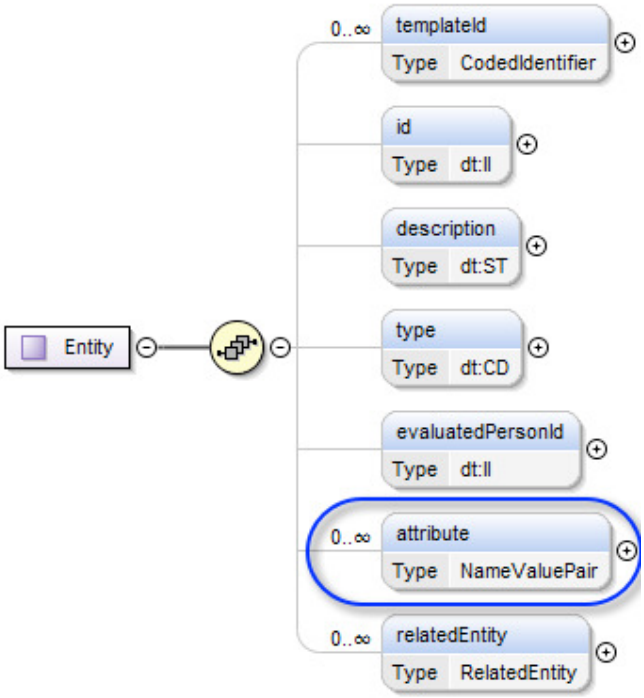


Figure 12 - Extending an entity

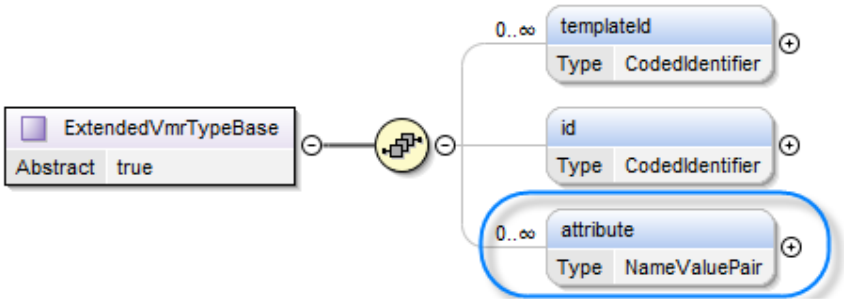


Figure 13 - Extending an ExtendedVmrTypeBase

2.8 cdsInput.xsd

CDSInput represents input data for a CDS system. As a specific example, a CDSInput may be used as the primary input data payload for a CDS guidance service compliant with the HL7 Decision Support Service standard. Further information regarding this type of use case can be found in the HL7 Decision Support Service specification and the HL7 Decision Support Service Implementation Guide. Further information is also available in the actual schema annotations.

The main component of the cdsInput.xsd schema is the CDSInput type. It is the container for the information that will be processed by a CDS engine necessary to provide patient guidance. It defines a number of elements that will aid in this task including the patient information payload contained within the vmrInput subtree (which is a vMR-XSD-compliant subtree) and the clinical context surrounding this request for guidance as specified by the cdsContext element.

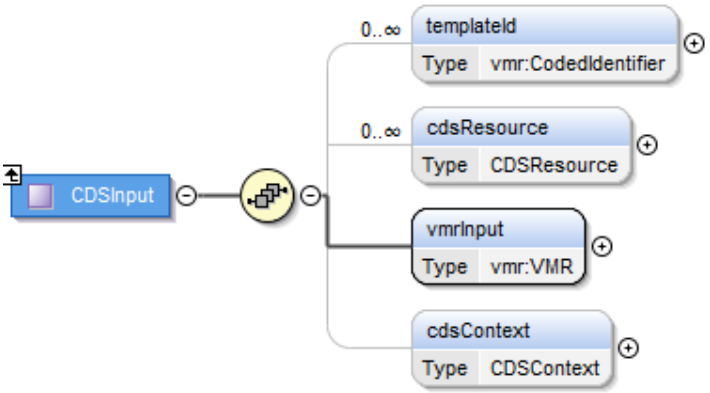


Figure 14 – CDSInput complex type

2.9 cdsInputSpecification.xsd

CDSInputSpecification specifies the CDS input required for a given CDS use case. As a specific example, a CDSInputSpecification may be used to specify required CDS input by a CDS guidance service compliant with the HL7 Decision Support Service standard. This type of specification can be encapsulated within the "query" section of a Decision Support Service's specification of knowledge module data requirements. Further information regarding this type of use case can be found in the HL7 Decision Support Service specification and the HL7 Decision Support Service Implementation Guide. Additional information is also available in the actual schema annotations.

The main components of the cdsInputSpecification.xsd schema are shown below.

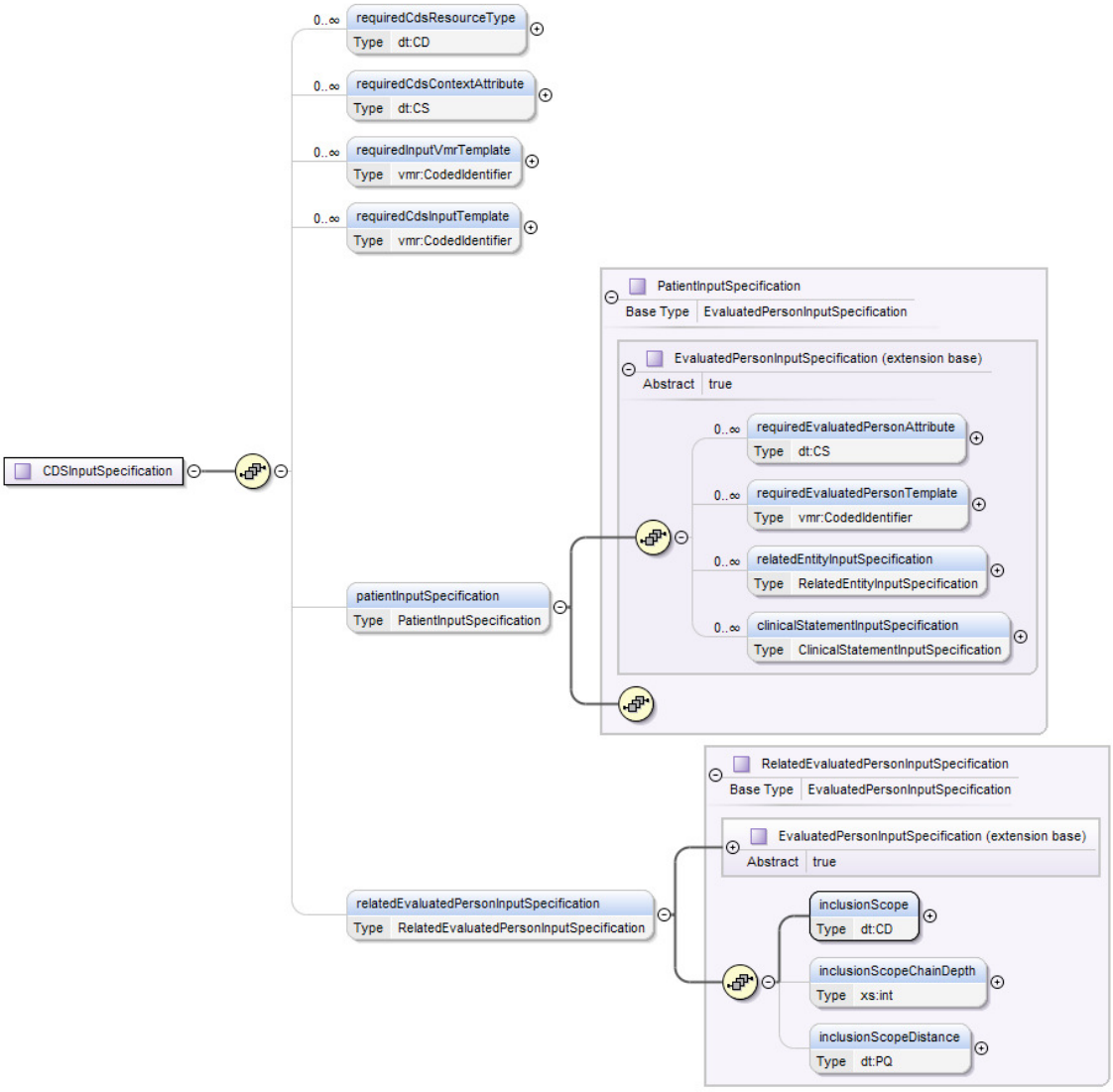


Figure 15 – CDSInputSpecification complex type

2.10 cdsOutput.xsd

CDSOutput represents the CDS output returned by a CDS system. As a specific example, a CDSOutput may be used as the primary output data payload for a CDS guidance service compliant with the HL7 Decision Support Service standard. Further information regarding this type of use case can be found in the HL7 Decision Support Service specification and the HL7 Decision Support Service Implementation Guide. Additional information is also available in the schema annotations.

The base CDSOutput complex type is an abstract type. It offers three concrete specializations:

1. CDSOutputAsDataType if the guidance provided can be summarized into a single value.
2. CDSOutputAsVMR if the guidance is broader. In this case, the output may be an updated vMR record or guidance in the form of a VMR record.
3. CDSOutputAsStringNameValuePairs should be used when the output is a collection of name-value-pairs.

Please refer to the vmr.xsd and cdsoutput.xsd schemas for additional information about each type and element illustrated in the diagrams below.

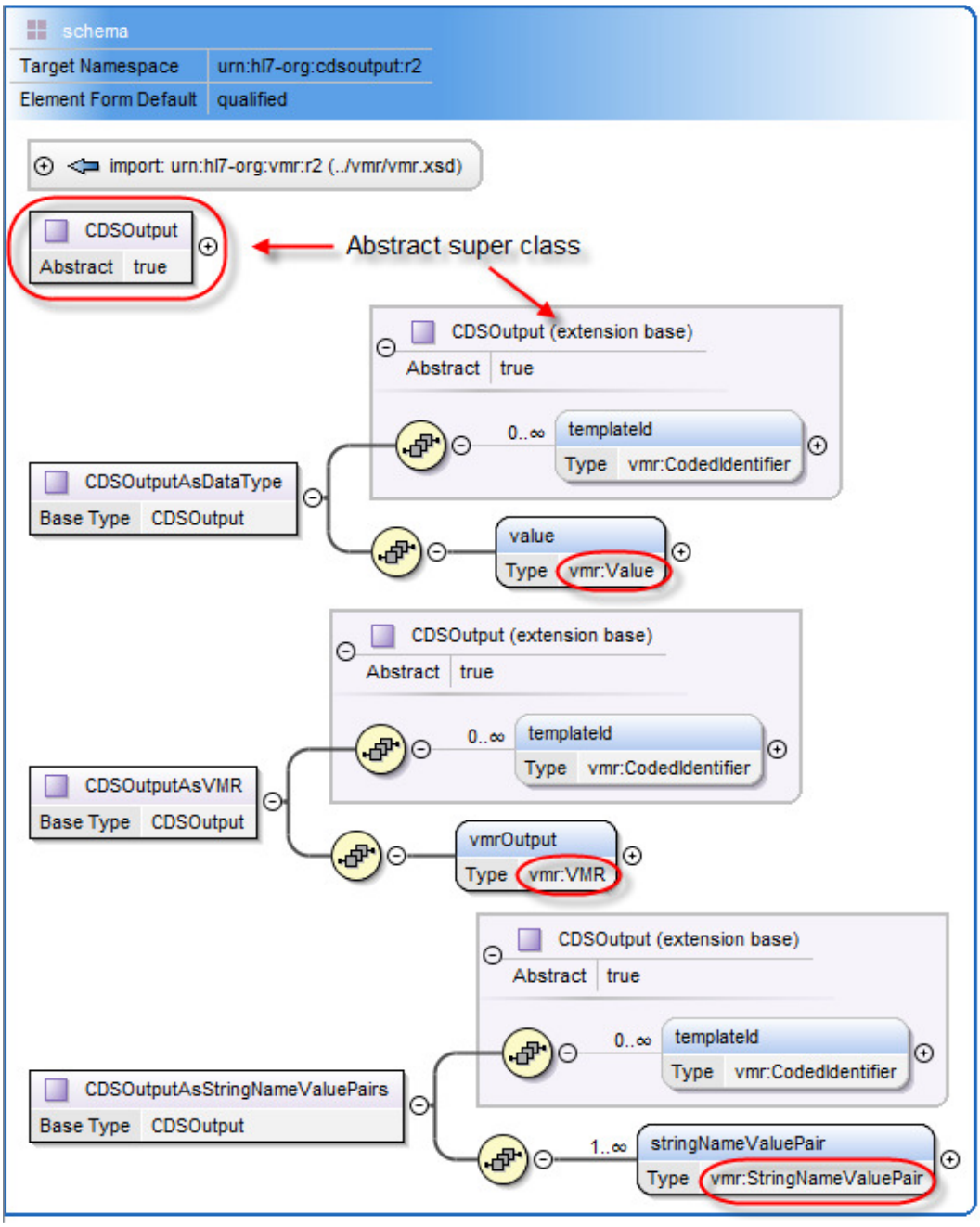


Figure 16 – CDSOutput complex type

2.11 cdsOutputSpecification.xsd

CDSOutputSpecification specifies the output to be returned by a CDS system. As a specific example, a CDSOutputSpecification may be used to specify details on the CDS output that will be returned by a CDS guidance service compliant with the HL7 Decision Support Service standard. This type of specification can be encapsulated within the "CDS output specification" section of a Decision Support Service's specification of knowledge module evaluation result semantics. Further information regarding this type of use case can be found in the HL7 Decision Support Service specification and the HL7 Decision Support Service Implementation Guide. Additional information is also available in the schema annotations.

The four main components of the CDS output specification schema consist of the following types:

1. CDSOutputSpecification – An abstract class from which constraints on three specializations of CDSOutput can be derived.
2. CDSOutputAsVMRSpecification – The specification of the CDSOutputAsVMR output.
3. CDSOutputAsDataTypeSpecification – The specification of the CDSOutputAsDataType output.
4. CDSOutputAsStringNameValuePairSpecification – The specification of the CDSOutputAsStringNameValuePair output.

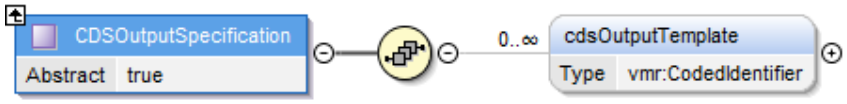


Figure 17 - CDSOutputSpecification is an abstract superclass

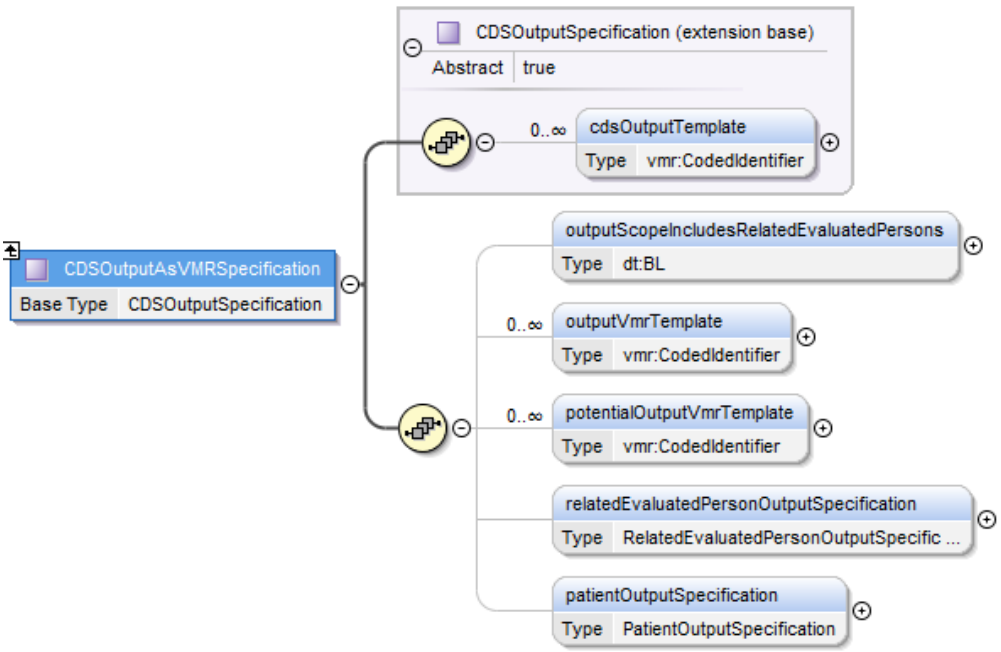


Figure 18 – CDSOutputAsVMRSpecification complex type

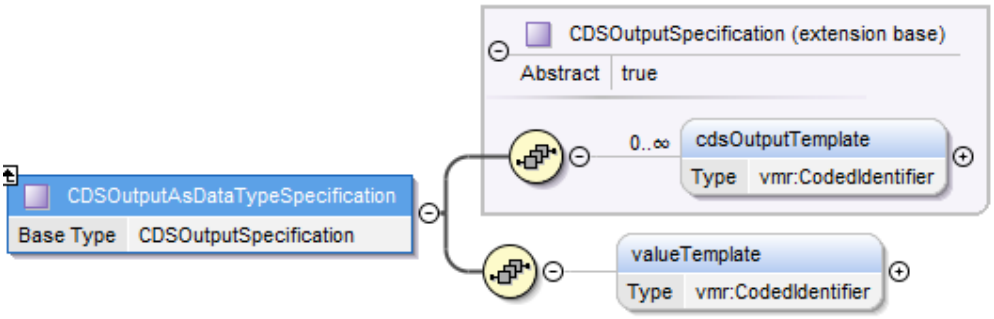


Figure 19 – CDSOutputAsDataTypeSpecification complex type

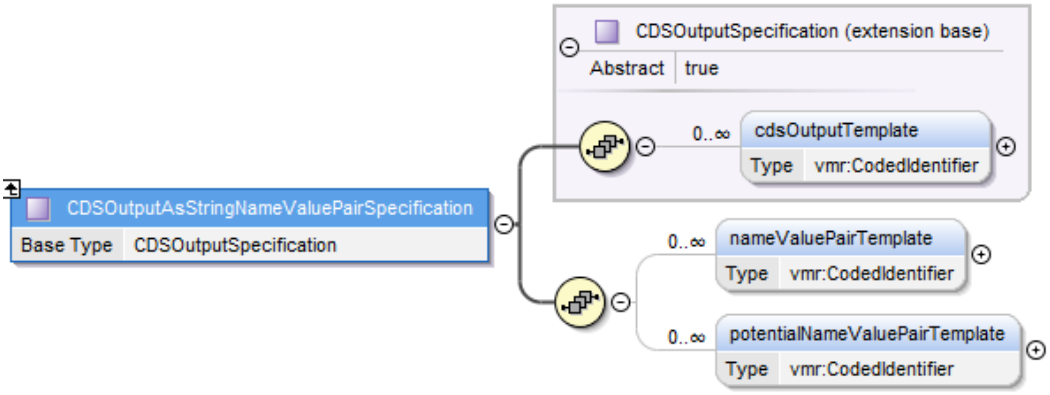


Figure 20 – CDSOutputAsStringNameValuePairSpecification complex type

2.12 Examples

An informative vMR is provided in the supplemental files.