HL7IG_CDS_KNART_R1_D1_2013JAN

HL7 Implementation Guide: Clinical Decision Support Knowledge Artifact Implementations, Release 1.1

March 2014

HL7 DSTU Specification

Sponsored by:

Clinical Decision Support in collaboration with the Health and Human Services Standards and Interoperability Framework Health eDecisions Working Group

Copyright © 2014 Health Level Seven International ® ALL RIGHTS RESERVED. The reproduction of this material in any form is strictly forbidden without the written permission of the publisher. HL7 and Health Level Seven are registered trademarks of Health Level Seven International. Reg. U.S. Pat & TM Off.

Use of this material is governed by HL7's **IP Compliance Policy**.

Identifying Information for Specification:

Specification Name and Release Number: HL7 Implementation Guide: Clinical Decision Support Knowledge Artifact Implementations, Release 1.1

Realm: Universal

Ballot Level: Draft Standard for Trial Use (DSTU)

Ballot Cycle: January 2013

Specification Date: March 2014

Version Number within Release 1: DSTU Update 1

Note Regarding Changes since Last Version:

Compared to Release 1 of the specification, this specification utilizes the latest available versions of the HL7 Virtual Medical Record (vMR) specifications, which were approved during the January 2014 HL7 ballot cycle.

© 2014 Health Level Seven International. All rights reserved

Acknowledgments

The authors wish to recognize the S&I Framework Health eDecisions Initiative Workgroup and the HL7 CDS Workgroup for their contributions to this document.

Copyrights

This material includes SNOMED Clinical Terms ® (SNOMED CT®) which is used by permission of the International Health Terminology Standards Development Organization (IHTSDO). All rights reserved. SNOMED CT was originally created by The College of American Pathologists. "SNOMED ®" and "SNOMED CT ®" are registered trademarks of the IHTSDO.

This material contains content from LOINC® (http://loinc.org). The LOINC table, LOINC codes, and LOINC panels and forms file are copyright (c) 1995-2011, Regenstrief Institute, Inc. and the Logical Observation Identifiers Names and Codes (LOINC) Committee and available at no cost under the license at http://loinc.org/terms-of-use.

This material includes content used from the Clinical Decision Support Consortium (CDSC), specifically the CDSC L3 schema.

This material includes content used from Allscripts and the CREF schema, which is used with permission of Allscripts.

This material includes content from HL7 Arden Syntax, which is used with permission of HL7.

This material includes content used from the Guidelines Element Model (GEM) which is used by permission of Yale University.

This material includes content from the Agency for Healthcare Research and Quality (AHRQ) and its eRecs project.

© 2014 Health Level Seven International. All rights reserved

EXECUTIVE SUMMARY

In support of the United States' national objectives for healthcare reform, the Office of the National Coordinator for Health Information Technology (ONC) Standards and Interoperability (S&I) Framework has sponsored the development of harmonized interoperability specifications. These specifications are designed to support national health initiatives and healthcare priorities, including Meaningful Use, the Nationwide Health Information Network, and the ongoing mission to improve population health.

The nation is reaching a critical mass of Electronic Health Record systems (EHRs) that comply with data and vocabulary standards. Providers seeking to meaningfully use EHRs face a variety of challenging tasks. Those tasks include assessing needs, selecting and negotiating with a system vendor or reseller, implementing project management, and instituting workflow changes to improve clinical performance, control costs, and ultimately, improve outcomes. Additionally, many providers face the challenge of integration and interoperation with disparate systems. Many institutions use their own proprietary vocabularies and data models. Though it may offer some internal flexibility, it comes with a high, often hidden, long term maintenance cost.

In support of this wide deployment of EHRs, there is an opportunity to implement a health learning system that includes clinical decision support (CDS) and provides a broad range of benefits that can contribute towards improved health of individuals and the population as a whole (refer to "Digital Infrastructure for the Learning Health System: The Foundation for Continuous Improvement in Health and Health Care: Workshop Series Summary" listed in Appendix A – Referenced Documents).

The S&I Framework Health eDecisions Initiative (HeD) is developing a foundational specification, reusing much of the work currently done in CDS standardization, to enable the structuring and encoding of CDS content for use as "knowledge artifacts." These artifacts can be used in support of many areas of the healthcare system, including quality and utilization measures, disease outbreaks, comparative effectiveness analysis, efficacy of drug treatments, and monitoring health trends. One of the key benefits of this proposed approach is the definition of a 'lingua franca' for the exchange of CDS knowledge and artifacts. Rather than having an un-scalable network of point-to-point communication channels, each with its own set of transformations, different organizations will only need to transform their content to an HeD-compatible format to communicate effectively with any other point in the network of providers that comprises today's health care system. If the models and vocabularies are rich enough, in the future, some CDS vendors may opt to use HeD as an internal specification.

This implementation guide is developed in support of the HeD Artifact Sharing Use Case and is intended to assist implementers in the development of Clinical Decision Support (CDS) Knowledge Artifacts. The approach adopted in this implementation guide is designed to be flexible and reusable, and to provide a baseline for CDS vendors and CDS Knowledge Artifact implementers.

© 2014 Health Level Seven International. All rights reserved

Table of Contents

1 INTRODUCTION 1
1.1 Design Approach and Rationale 1
1.1.1 How a CDS Knowledge Artifact Works 3
1.1.2 Lifecycle of a CDS knowledge artifact 3
1.1.3 Extensibility of the HeD CDS Knowledge Artifact schema 4
1.1.4 Representation Rationale 4
1.2 Audience 6
1.2.1 Requisite Knowledge 6
1.3 Scope of the Guide 7
1.3.1 Contents of the Proposed Ballot 7
1.4 Alignment to HeD Artifact Sharing Use Case 8
1.4.1 Use Case Assumptions and Conditions 9
1.4.2 Usage Conformance Testing Recommendations 9
1.5 Organization of this Guide 9
1.5.1 Definitions of Terms 9
1.5.2 Definitions of Actors 11
1.5.3 Conventions Used 12
1.5.4 Schema Roadmap 15
2 CDS KNOWLEDGE ARTIFACT TYPES 16
2.1 Conformance to the Health eDecisions Knowledge Artifact Specification 16
2.2 HeD Knowledge Artifacts 16
2.2.1 Metadata Elements 16
2.2.2 Def Elements 17
2.2.3 Expression Elements 17
2.2.4 Trigger Elements 17
2.2.5 Condition Elements 17
2.2.6 Behavior Elements 17
2.2.7 Action Elements 17
2.3 Event Condition Action (ECA) Rules 18
2.3.1 Conceptual Overview 18
2.3.2 ECA Rule CONFORMANCE REQUIREMENTS 19
2.4 Order Sets 21
2.4.1 Conceptual Overview 21
2.4.2 Order Set Conformance Requirements 23
2.5 Documentation Templates 24

© 2014 Health Level Seven International. All rights reserved

2.5.1 Conceptual Overview 24
2.5.2 Documentation Template Conformance Requirements 26
2.6 Value Sets and Vocabulary Codes 28
2.6.1 Value Set Use Case 28
2.6.2 Coded Data versus Free Text 28
3 CDS KNOWLEDGE ARTIFACT COMPONENTS 30
3.1 Metadata 30
3.1.1 Clinical Data Models 30
3.2 Actions 31
3.2.1 Types of Actions 31
3.2.2 Creating Action Sentence Expressions 33
3.2.3 Organizing Actions 34
3.2.4 Conditional Actions 34
3.2.5 Behaviors 35
3.3 Triggers and Events 40
3.3.1 DataEvent 40
3.3.2 PeriodicEvent 40
3.4 Expressions 40
3.4.1 Basic Values (Scalars) 41
3.4.2 Comparison Operators 42
3.4.3 Logical Operators 43
3.4.4 Nullological Operators 44
3.4.5 Conditional Operators 44
3.4.6 Arithmetic Operators 46
3.4.7 String Operators 47
3.4.8 Date and Time Operators 48
3.4.9 List Values 49
3.4.10 Aggregate Operators 56
3.4.11 Interval Values 58
3.4.12 Structured Values 61
3.4.13 Reusing Expressions 62
3.4.14 External Data 63
3.4.15 Parameters 66
3.5 Extending the Schema 67
3.5.1 Extending Types 67
3.5.2 Extending Enumerations 67
3.5.3 Other Extensions and Modifications 68

© 2014 Health Level Seven International. All rights reserved

4 LANGUAGE IMPLEMENTATION 69
4.1 Clinical Data Retrieval in HeDS Artifacts 69
4.1.1 Defining Clinical Data 69
4.1.2 Conformance Levels 70
4.1.3 Artifact Data Requirements 70
4.2 Expression Language Conceptual Model 71
4.2.1 Data Model 71
4.2.2 Language Elements 73
4.2.3 Semantic Validation 73
4.2.4 Execution Model 74
5 SCHEMA REFERENCES 77
6 APPENDIX A – REFERENCED DOCUMENTS 438
7 APPENDIX B – ACRONYMS 440
8 APPENDIX C – DEFINITIONS 441
9 APPENDIX D – HED SCHEMA FRAMEWORK 442
9.1 Overview 442
9.1.1 Technology 442
9.1.2 Solution Structure 442
9.1.3 Design Goals 443
9.2 Components 444
9.2.1 Maps 444
9.2.2 Model 445
9.2.3 Reading 445
9.2.4 Writing 445
9.3 Verification 445
9.3.1 Type Resolution 446
9.3.2 Operator Resolution 446
9.3.3 Node Verification 447
9.3.4 Symbol Resolution 448
9.4 Translation 448
9.4.1 Artifact Translation 448
9.4.2 Node Translation 448
9.4.3 Model Translation 448
9.5 CREF Translation 449
9.6 Metadata 449
9.7 Syntax 449
9.7.1 External Data and Expressions 449
9.7.2 Parameters 450

© 2014 Health Level Seven International. All rights reserved

9.7.3 Scalar Values 450
9.7.4 Complex Types 450
9.7.5 Casting and Conversion Operators 450
9.7.6 Null-Handling Operators 451
9.7.7 Unary Operators 451
9.7.8 Binary Operators 451
9.7.9 Logical Operators 452
9.7.10 Set/List Operators 452
9.7.11 Date/Time Operators 452
9.8 Model 453
9.8.1 Patient Age 454
9.8.2 Negation Rationale 454
9.8.3 Procedures and Medications 455
9.8.4 Encounter Locations 455
9.9 Value Sets 455
9.10 Guidance 456
9.10.1 Severity 456
9.10.2 Constructed Guidance 456
9.10.3 Dynamic Guidance 457
10 APPENDIX E – EXAMPLES 458
10.1 FLACC Example 458
10.2 RespiratoryOrder Example 464
10.3 DopamineComplexIVOrderWithComplexLiteral Example 470
10.4 HeartFailureAdmissionToMedSurgOrderSet Example 475

© 2014 Health Level Seven International. All rights reserved

1 INTRODUCTION

The S&I Framework is an approach adopted by ONC's Office of Standards & Interoperability to fulfill its charge of enabling harmonized interoperability specifications to support national health outcomes and healthcare priorities. The S&I Framework is a collaborative community of participants from the public and private sectors who are focused on providing the tools, services and guidance to facilitate the functional exchange of health information. More information about the S&I Framework can be found here: http://siframework.org/

The S&I Framework uses a set of integrated functions, processes, and tools that enable execution of specific value-creating initiatives. Each S&I Initiative focuses on a single, narrowly-scoped, broadly-applicable challenge. The goal of the Health eDecisions (HeD) Initiative is to identify, define and harmonize standards and specifications that facilitate the emergence of systems and services whereby shareable clinical decision support (CDS) can be implemented. Additional information about the HeD initiative and CDS, including a Project Charter, can be found here:
http://wiki.siframework.org/Health+eDecisions+Project+Charter+and+Members

The HeD Initiative, based on stakeholder input and subject matter expert (SME) guidance, developed the HeD Artifact Sharing Use Case (HeD Use Case 1) to define the functional requirements for building a schema for the contents of three specific CDS Knowledge Artifact types – Event, Condition Action (ECA) Rules, Order Sets, and Documentation Templates. Consensus on Use Case 1 was achieved on Thursday, September 13, 2012, and the consensus approved Use Case can be found here:
http://wiki.siframework.org/Health+eDecisions+Use+Case

Following consensus on the functional requirements outlined in HeD Use Case 1, the Harmonization phase kicked off. This included development of a Consensus Statement to formalize the direction and technical approach adopted by the initiative, and to serve as a roadmap during the definition of the technical artifacts and supporting documentation. The consensus statement for the HeD Initiative can be found here:
http://wiki.siframework.org/Health+eDecisions+Consensus+Statement

Based off of the functional requirements outlined in the HeD Artifact Sharing Use Case, and the guidance outlined in the HeD Initiative Consensus Statement, the initiative harmonized a set of existing industry standards and specifications, and developed a CDS Knowledge Artifact schema, which represents each CDS Knowledge Artifact component in a standardized format. The Schema can be found here: http://code.google.com/p/health-e-decisions/source/browse/#svn%2Fbranches%2Fv1.1%2Fsrc%2Fmain%2Fschema

This Implementation Guide serves as a companion document for the CDS Knowledge Artifact schema, and includes additional background, contextual information, and detailed documentation and guidance in support schema implementation. In this introductory section, the approach and purpose for the CDS Knowledge Artifact Implementation Guide is defined and discussed. The specific approach to conceptually designing the CDS Knowledge Artifact is outlined, using existing implementation schemas and standards, and the lifecycle and extensibility of the CDS Knowledge Artifact are explored.

1.1 Design Approach and Rationale

The specification defined in this implementation guide is based on a harmonization of existing standards and specifications as the primary mechanism for structuring knowledge artifacts. This guide is laid out to support the following implementation objectives:

© 2014 Health Level Seven International. All rights reserved

1. To provide an overview and list of the standards/specifications upon which the Health eDecisions CDS Knowledge Artifact schema is built and explain how each contributes to the HeD CDS Knowledge Artifact schema.
2. To provide the rationale for why each standard/specification was chosen and to specify how they play together to solve this important problem.
3. To specify what piece of each standard/specification is used in the HeD CDS Knowledge Artifact schema and why.
4. To specify where and how the documentation for each supporting standard/specification may be obtained.
5. To explain the key concepts and components defined in this the guide and how they relate to one another.

The technical approach adopted by drawing on existing standards and specifications is designed to provide a catalog of components for generating knowledge artifacts, with a specific focus on defining the structure of the components and how they fit together. The intent of the CDS Knowledge Artifact specification is to provide a format for sharing. Because it is intended to provide an unambiguous mechanism for sharing, it may also be used as a format for processing and execution. This specification is not intended as a requirement for implementation, but as a vehicle for sharing CDS Knowledge Artifacts.

This implementation guide focuses not only on structure but also on semantics through the use of standard terminologies, value sets and taxonomies such as SNOMED-CT. The specifications highlight terminology bindings where defined and required.

This technical approach, of drawing on existing standards and specifications, is designed to support multiple goals for implementers:

1) **Maintenance**: Any subsequent change to components and supporting standards/specifications used for generating knowledge artifacts will be made in this single-source guide, which will then be republished and made available to the clinical decision support community.
2) **Translation**: This implementation guide is the natural home for transformation and mapping rules to convert HeD CDS Knowledge Artifact components into other relevant formats. Examples of the transformations supported through this guide include translations from the CDS Knowledge Artifact schema to the HL7 Order sets DSTU and HL7 Arden Syntax. The methods of transformation and specific examples to support the transforms are included in this implementation guide.
3) **Modularity**: Those familiar with the CDS Knowledge Artifact Implementation Guide and its defined components are able to mix and match specific components needed to support the development and/or consumption of CDS knowledge artifacts. This lessens complexity of implementation and integration by supporting customization to the CDS vendor environment.
4) **Compatibility**: Different types of knowledge artifacts share common substructures that are addressed within one harmonized schema which can structurally map to multiple types of CDS knowledge artifacts, so that all EHR and CDS systems operate using the same set of components, constraints, terminologies, and value sets.

© 2014 Health Level Seven International. All rights reserved

5) **Ease of Use:** The structure of this implementation guide is designed in a way that makes the process of implementing and integrating the CDS Knowledge Artifact easier, by providing enough constraint to ensure CDS Knowledge Artifacts are free of technical content errors, and ensure that the artifact blends easily into the user's workflow.

The approach taken to develop the CDS Knowledge Artifact Implementation Guide has some similarity to existing approaches that have been used to develop balloted implementation guides, such as the approach used to develop a template library within the HL7 Implementation Guide for CDA® Release 2: IHE Health Story Consolidation, DSTU Release 1.1 (US Realm).

The CDS Knowledge Artifact Implementation Guide is an implementation specification structured as a series of layered constraints for each of the components used in the CDS Knowledge Artifact. This implementation guide adopts specific constraints defined in each of the harmonized schemas and specifications (see Section 1.5.3) and then adds constraints through conformance statements that further define and restrict the sequence and cardinality of components and the vocabularies for coded elements. These conformance statements are defined within the implementation guide at a high level to support initial piloting and adoption of the CDS Knowledge Artifact.

1.1.1 How a CDS Knowledge Artifact Works

The foundation of each component is the CDS Knowledge Artifact schema, which represents each CDS Knowledge Artifact component in a standardized format for generation and consumption. The schema is a harmonized aggregation of multiple existing CDS standards and specifications.

Each of the three types of CDS Knowledge Artifacts detailed in this guide is comprised of reusable blocks of schema, called "components." Each of the components defined in this schema represents an individual building block that can be used to assemble a CDS Knowledge Artifact. Each component is further broken down into "types" to enable further granularity when applying constraints. The modular approach is based on the concept of defining specific constraints for each of the components defined within the CDS Knowledge Artifact, to allow for implementation of components regardless of CDS vendor environment. The components used by each artifact type are defined in Section 2 of this guide, with Section 3 covering detailed documentation of all types leveraged in constructing the schema.

This approach is based on several key technical decisions made by the Health eDecisions Initiative Workgroup, through the S&I Framework process, including the use of a harmonized schema that seeks to define a new standardized format for multiple types of CDS Knowledge Artifacts.

1.1.2 Lifecycle of a CDS knowledge artifact

The CDS Knowledge Artifact is the structured and encoded format that this implementation guide supports. It is important to understand the lifecycle of a CDS Knowledge Artifact. Various actions may be carried out on a CDS Knowledge Artifact, each one resulting in a state change of the artifact.

The ovals in red show the different actions that may be performed on a CDS Knowledge Artifact – creation of the artifact (**Created**), pre-publication of the artifact (**Pre-Published**), publication of the artifact by a CDS Content Supplier (**Published**), review of the CDS Knowledge Artifact (**Reviewed**), replacement of an existing artifact with another artifact (**Superseded**) and retirement of the artifact from use (**Withdrawn**).

© 2014 Health Level Seven International. All rights reserved

The rectangles in green show the different states that a CDS Knowledge Artifact goes through – creation prior to publication (**Draft**), in testing (**InTest**), active once published (**Active**), and retired (**Inactive**).

It should be noted that these states and actions apply to a particular version of a CDS Knowledge Artifact, and that any change to a CDS Knowledge Artifact version should result in a new version of that artifact.

FIGURE 1 – CDS KNOWLEDGE ARTIFACT LIFECYCLE

1.1.3 Extensibility of the HeD CDS Knowledge Artifact schema

The CDS Knowledge Artifact types and components are designed to support extensibility by implementers, which allows for CDS vendors to employ flexibility when assembling the artifacts. The CDS Knowledge Artifact schema incorporates a base level of conformance, using conformance language as structured using guidance in Section 1.5.3.4, and then allows for flexibility by implementers in determining the additional attributes and values that may be applied to artifact components.

1.1.4 Representation Rationale

A critical component of the representation of a CDS Knowledge Artifact is the Expression component. In fact, much of the body of the artifact, including the sections for triggers, data requests, conditions, and actions are specified as computable expressions. This is in contrast to document artifacts containing data which contain information specified in the form of data instances. This component is described in more detail in later chapters, but it is important to note the rationale for the way that expressions are represented.

© 2014 Health Level Seven International. All rights reserved

1.1.4.1 Data Model

The HeD Schema does not directly deal with any particular clinical data model. Instead, the schema provides a mechanism to identify the data model that is used by an artifact. The first reason for this separation is that clinical data models are quite volatile, in that they need to react to changing clinical, business, regulatory, and other requirements.

By contrast, the operations described by artifacts tend be more stable. In other words, the language used to reason about clinical data models evolves much more slowly than the clinical data models themselves. Keeping a clear separation between the representation of the logic and the clinical data models on which that logic operates, minimizes the impact of those changes.

In addition, keeping the data model separate from the representation of the logic results in a more flexible specification, as the same general purpose language can be used to deal with data from multiple models if necessary.

And finally, the separation results in a simpler implementation, as the details of dealing with particular clinical data models can be isolated from the implementation of the operations of the expression language.

1.1.4.2 Expression Logic

CDS Knowledge Artifacts represent expression logic within the artifact using XML types that correspond directly to language elements. This approach is designed to achieve an optimal balance between human readability and simplicity of implementation and machine processing, and is based on the concept of an Abstract Syntax Tree from traditional compiler implementation.

The following diagram depicts the steps performed by a traditional compiler:

© 2014 Health Level Seven International. All rights reserved

As shown above, the XML representation is defined as an Abstract Syntax Tree, eliminating the need for lexical analysis and parsing steps, and allowing implementations to concentrate on the core representation of the logic.

In addition, this approach avoids potential ambiguity that must be resolved with operator precedence and/or the use of parentheses in traditional expression languages.

The result is a dramatic reduction in the complexity of processing CDS Knowledge Artifacts, whether that processing involves translation to another format, evaluation of the logic, or building a user-interface for authoring or visual representation of the artifact.

1.2 Audience

The audience of this implementation guide includes, but is not limited to, software developers, CDS and EHR vendors, and other HIT implementer parties that are interested in developing and consuming CDS knowledge artifacts. This implementation guide also specifically covers consumers and integrators of the CDS Knowledge Artifact.

1.2.1 Requisite Knowledge

This section includes pre-requisites for implementers and other users of the CDS Knowledge Artifact Implementation Guide. Specific prerequisites for CDS implementations using the CDS Knowledge Artifact Implementation Guide include the following (summarized into required information and optional information):

Required Information	Optional Information
Implementers must have a strong knowledge of the S&I Framework HeD Artifact Sharing Use Case and an understanding of Section 11 (Dataset Requirements)	Implementers should have a strong knowledge of the HL7 standards underlying this implementation guide, including: • The HL7 Virtual Medical Record (vMR) standard • The HL7 Arden Syntax standard
Implementers must have a strong understanding in the use of XML, specifically XML Schema.	Implementers should reference existing documentation and schemas on the CDSC L3 schema (as noted in Appendix A – References)
	Implementers should reference existing documentation from the Agency for Healthcare Research and Quality (AHRQ) and their eRecs project.
	Implementers should reference existing documentation and schemas on the Allscripts CREF schema (as noted in Appendix A – References)
	Implementers should reference existing documentation and schemas on the Guidelines Element Model (GEM)
	Implementers should have a basic understanding of the following vocabularies/value sets: • CPT • CVX • NDF-RT • LOINC • SNOMED-CT

TABLE 1 – IMPLEMENTATION PREREQUISITES – AUDIENCE

For those implementers without requisite knowledge in these areas, it is HIGHLY RECOMMENDED to read each of the documents defined in Appendix A – Referenced Documents. Note that the table above categorizes requisite knowledge for concepts covered

© 2014 Health Level Seven International. All rights reserved

in this implementation guide in its entirety – this categorization of prerequisites may not apply for those implementations where only a specific CDS Knowledge Artifact and/or specific components are being considered.

1.3 Scope of the Guide

As defined in HeD Use Case 1, the scope of this implementation guide is limited to the CDS Knowledge Artifact and the different components that will be included in the knowledge artifact. This includes how to structure and encode the data elements for each of these components, and the structuring of the data elements into different knowledge artifact types. The three artifact types that are in scope are:

1) Event Condition Action (ECA) Rules
2) Order Sets
3) Documentation Templates

Specifically out of scope is the usage of the knowledge artifact with defined system behaviors, such as how to generate the artifact, how to export and import it, or how to populate a knowledge artifact using data from an existing system, such as an Electronic Health Record (EHR). Each of these system behaviors is specifically excluded from this implementation guide.

As part of the scope, validation of the knowledge artifact is included within scope, for high level conformance with the CDS Knowledge Artifact schema, which is the XML schema based off of the harmonized standards and specifications identified by HeD to satisfy Use Case 1 requirements. Conformance requirements are currently defined at a high level and are limited to the structure and encoding of the knowledge artifact. Additionally, schematron rules will be defined to express conformance constraints.

Wherever possible, specific validation rules are also included for terminologies, such as the use of SNOMED-CT within a CDS Knowledge Artifact.

1.3.1 Contents of the Proposed Ballot

This table summarizes the materials included in the proposed ballot package for the S&I Framework Health eDecisions initiative CDS Knowledge Artifact submission. This ballot delivery package is intended for review by the HL7 community and other interested parties in the November-January timeframe:

Filename	Description	Standards Applicability
CDS Knowledge Artifact Implementation Guide	Implementation Guide	DSTU
action.xsd	Contains the action types needed for defining actions (covers Actions as defined in the HeD Artifact Sharing Use Case)	Informative
actor.xsd	Contains the Actor type, needed for all CDS Knowledge Artifacts	Informative
base.xsd	Contains the base types needed for assembling CDS Knowledge Artifacts	Informative
behavior.xsd	Contains the types needed for defining behaviors at the artifact, action, and group level	Informative

© 2014 Health Level Seven International. All rights reserved

Filename	Description	Standards Applicability
catalogItem.xsd	Contains the types needed to build documentation templates	Informative
clinicalExpression.xsd	Contains the types associated with data mapping to the external models such as the vMR	Informative
condition.xsd	Contains the condition types, needed for ECA rules.	Informative
literalExpression.xsd	Contains the types needed to create literals of datatypes within expressions	Informative
datatypes.xsd	Supports the base datatypes needed for CDS Knowledge Artifacts Imports ISO 21090 data types	Informative
expression.xsd	Contains the types needed for building Expressions (covers Expressions as defined in the HeD Artifact Sharing Use Case)	Informative
extdatatypes.xsd	Contains all the extensions to base ISO 21090 data types that are needed to support the HeD schema	Informative
knowledgeDocument.xsd	The main container for a CDS Knowledge Artifact	Informative
metadata.xsd	Contains the Metadata types for a CDS Knowledge Artifact (covers the Knowledge Artifact metadata as defined in the HeD Artifact Sharing Use Case)	Informative
xhtml1-strict.xsd	Per W3C, this schema defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three Document Type Definitions (DTDs) corresponding to the ones defined by HTML 4. The semantics of the elements and their attributes are defined in the W3C Recommendation for HTML 4.	Informative
HeDSchema.eap	Contains the CDS Knowledge Artifact schema and associated clinical data mappings in a UML model in a proprietary format	Informative
HeDSchema.xsi	Contains the CDS Knowledge Artifact schema and associated clinical data mappings in a UML model in a standard format	Informative

TABLE 2 – PROPOSED BALLOT MATERIALS – CDS KNOWLEDGE ARTIFACT IG

1.4 Alignment to HeD Artifact Sharing Use Case

The specific requirements implemented within this guide are focused on the structure, format, and encoding of a CDS Knowledge Artifact. These requirements are directly tied to the HeD Artifact Sharing Use Case (HeD Use Case 1) and as noted in Section 1.4.1 of this guide, a thorough understanding of the use case is expected for implementation.

© 2014 Health Level Seven International. All rights reserved

Full material on the HeD Artifact Sharing Use Case can be found here:

http://wiki.siframework.org/file/view/SIFramework_HeD_UC1_CDSArtifactSharing_v1.0.docx/371583300/SIFramework_HeD_UC1_CDSArtifactSharing_v1.0.docx

1.4.1 Use Case Assumptions and Conditions

It is important for implementers to clearly understand the underlying CDS environmental assumptions defined in Section 5 of the HeD Artifact Sharing Use Case, to ensure that these assumptions align to the implementation environment in which CDS content will be exchanged using a knowledge artifact. Failure to meet any of these assumptions could impact implementation of the knowledge artifact.

1.4.2 Usage Conformance Testing Recommendations

The following text is pre-adopted from the HL7 V2.7.1 Conformance (Chapter 2B, 2.B.7.5). Please refer to the base standard documentation for a full explanation of conformance concepts. Usage is described here as it introduces the revised approach to conditional element handling; upon successful ballot and publication this material will be replaced with a reference to the normative documentation.

1.4.2.1 System Requirements

This implementation guide is not focused on specific EHR and CDS system behaviors that may apply to the CDS Knowledge Artifact, such as the interaction between specific actors within the Use Case who may wish to search or import a CDS knowledge artifact.

Specific system requirements targeted in this implementation guide include the following:

System Requirement	Description
Provides CDS Knowledge Artifact in Structured Format	The implementation guide defines how the knowledge artifact should be structured, NOT how the system actually generates the structure.
Provide metadata about CDS artifact in a standardized structured format	The implementation guide defines how the knowledge artifact metadata is applied to the different knowledge artifact types, and DOES NOT specify how systems should parse and interpret this metadata.

TABLE 3 – CDS SYSTEM REQUIREMENTS COVERED IN THIS GUIDE

1.5 Organization of this Guide

It is important for readers of this implementation guide to understand specific terms, actors, roles, and conventions used in this implementation guide.

1.5.1 Definitions of Terms

Several terms are used throughout this document and a level of detailed technical understanding of healthcare standards is expected. It is extremely critical for the reader to review Appendix C - Definitions, to understand the specific acronyms and terms that are used in this implementation guide. In addition, the reader should be familiar with the terms defined in Appendix A of the HeD Artifact Sharing Use Case – wherever possible, this implementation guide reuses existing terms from that document and seeks to minimize the introduction of any new terms, including those listed in the table below.

Term Name	Description of Term and Role in HeD Harmonized Schema
AHRQ eRecs	The AHRQ eRecs project is a source of recommendations that was used to inform the design of the expression language used within the CDS Knowledge Artifact schema.

© 2014 Health Level Seven International. All rights reserved

Term Name	Description of Term and Role in HeD Harmonized Schema
Allscripts CREF	The Allscripts CREF schema is a foundational set of schemas that was used to inform the design of the design of the expression language used within the schema
CDS Knowledge Artifact	The CDS Knowledge Artifact (as defined in the HeD Artifact Sharing Use Case) is medical knowledge represented in a structured and encoded form to enable computer-based clinical decision support. This implementation guide specifically is focused on the structure and encoding necessary to make the knowledge artifact available as CDS content.
CDS Knowledge Artifact Type	The CDS Knowledge Artifact Type represents the different types of CDS content that may constitute a CDS Knowledge Artifact. As defined in the HeD Artifact Sharing Use Case, the artifact type may consist of artifact specific data, metadata, and the components specific to that type of artifact. This implementation guide specifically supports three initial knowledge artifact types: 1) Event Condition Action (ECA) Rules 2) Order Sets 3) Documentation Templates
CDSC L3	The CDSC (CDS Consortium) L3 schema is foundational to the design and structure of the CDS Knowledge Artifact schema and is used as a source for a large number of the types defined in the schema. This schema has been closely aligned to the CDSC L3 schema.
Component	A component is a grouping of data elements within the CDS Knowledge Artifact. The structure of this implementation guide supports the use of a library of reusable components when developing a knowledge artifact.
Documentation Templates	A structured form for recording information on a patient into a set of pre-defined data slots.
Event Condition Action (ECA) Rules	A CDS knowledge artifact with the general syntax “on event, if condition is true, then do action.”
CDS Knowledge Artifact schema	The CDS Knowledge Artifact schema is the formal XML schema of the harmonization of multiple CDS standards and specifications adopted as the starting point for the CDS Knowledge Artifact structure, such as: • HL7 vMR • CDSC L3 • Allscripts CREF • Arden Syntax • Yale GEM • AHRQ eRecs
HL7 Arden Syntax	The HL7 Arden Syntax standard is a primary source of knowledge that was used to inform the development of the expression language used within the CDS Knowledge Artifact schema.

© 2014 Health Level Seven International. All rights reserved

Term Name	Description of Term and Role in HeD Harmonized Schema
HL7 vMR (Virtual Medical Record)	A Virtual Medical Record (vMR) for Clinical Decision Support (CDS) is a data model for representing clinical data relevant to CDS. The vMR encompasses data about a patient's demographics and clinical history, as well as CDS inferences about the patient (e.g. recommended clinical interventions). The vMR standard plays two different roles in the HeD schema: 1) Expressions in a CDS Knowledge Artifact may refer, through the use of expressions, to an actual vMR record included as an 'external data reference' in the artifact. These are used for the purpose of deciding applicability of a portion of the artifact to a patient population. (For example, deciding the amount of insulin to give a diabetic patient is a function of the patient's blood sugar levels.) 2) Actions in the CDS Knowledge Artifact (referred to as "orders" by many EHR systems) are represented using the same clinical data elements that are used in a virtual medical record. For example, the vMR "substance administration" type is used as the model to specify the medication to give a patient, including dose, frequency, PRN reasons, etc.
Order Sets	A pre-defined and approved group of orders related to a particular clinical condition (e.g., hypertension treatment and monitoring) or stage of care (e.g., hospital admission to Coronary Care Unit). Order sets are used within electronic health record systems as a checklist for physicians when treating a patient with a specific condition. An order set is a structured collection of orders presented to the physician in a computerized physician order entry system (CPOE).
Yale Guidelines Element Model (GEM)	The Yale Guidelines Element Model (GEM) is a supporting schema that is harmonized by the CDS Knowledge Artifact schema and provides additional background material for elements such as Knowledge Resources. GEM was a significant conceptual input to the CDSC L3 schema.

TABLE 4 – KEY TERMS IN THIS GUIDE

1.5.2 Definitions of Actors

This implementation guide is specifically targeted to meeting the requirements of the following roles and actors (each of which are drawn from the HeD Artifact Sharing Use Case)

Actor/Role from HeD Artifact Sharing Use Case	How Implementation Guide Supports This Role
CDS Knowledge Artifact Supplier (including CDS/EHR vendors and CDS content suppliers) An organization/system that creates,collects and/or distributes CDS Knowledge Artifacts.	Provides the structure and encoding format needed to generate CDS content that conforms to the Health eDecisions CDS Knowledge Artifact schema. NOTE: This implementation guide DOES NOT specify HOW the content supplier actually generates the CDS knowledge artifact, only the desired structure and encoding of that artifact.
CDS Knowledge Artifact Integrator (including CDS/EHR vendors and healthcare delivery systems that implement CDS systems) An organization/person/system that imports, adapts, and maps CDS Knowledge artifacts to be embedded within a CDS system.	Provides a standardized format for vendors and implementers of EHR and CDS systems to adopt when creating and/or consuming CDS Content. NOTE: CDS and EHR Vendors (as with all HeD Artifact Sharing actors) are given considerable flexibility when implementing the CDS Knowledge Artifact

© 2014 Health Level Seven International. All rights reserved

TABLE 5 – ROLES AND ACTORS SUPPORTED IN THIS GUIDE

1.5.3 Conventions Used

The conventions defined in this document are specifically drawn from other implementation guides and include common conventions adopted by HL7, IHE, ASTM, and ISO. This implementation guide adopts high-level conformance statements that apply to each complex type, element, and attribute defined within the HeD schema.

1.5.3.1 Use of Cardinality

Cardinality applies specifically to metadata and the data elements associated with the CDS Knowledge Artifact. The specific conventions for cardinality in this implementation guide are as follows:

Cardinality	Explanation of Cardinality
0..0	The element is never present
0..1	The element MAY be omitted and has at most one occurrence
1..1	The element is present once and only once
0..n	The element MAY be omitted or may repeat up to *n* times
1..n	The element MUST appear at least once, and MAY repeat up to n times
0..*	The element MAY be omitted, or it MAY repeat an unlimited number of times
1..*	The element MUST appear at least once, and MAY repeat an unlimited number of times
m..n	The element MUST appear at least *m* times, and at most, *n* times
2..2	The element MUST appear two and only two times
3..3	The element MUST appear three and only three times

TABLE 6 – SUMMARY OF CARDINALITY

1.5.3.2 Use of Versioning

Version control for this implementation guide and the associated schema files is enforced using the Google Code Repository that hosts the Health eDecisions/CDS Knowledge Artifact project. Each of the components included in the associated CDS Knowledge Artifact XML schema files are kept in this repository.

Versioning is of critical importance for this implementation guide due to the large number of XML schemas included in this implementation guide, and wherever necessary, the specific version of the XML schema being referenced in this implementation guide is noted. In all cases, the schema files hosted in the Google Code Repository are to be noted as the source of truth.

1.5.3.3 Use of References

Documentation and terms that appear throughout this document in ***bold italic*** text indicate a specific reference. Documents are referenced to indicate that implementers should refer to that documentation for final conformance language and other levels of guidance. An example is shown below:

For conformance language, please refer to the ***Conformance Implementation Manual*** for further details

Working code examples are also provided in this implementation guide to assist in understanding the CDS Knowledge Artifact schema. While this implementation guide is normative, examples are meant to be informative, and are provided for human readability. In all cases, the formal specification referred to by the example takes precedence.

© 2014 Health Level Seven International. All rights reserved

1.5.3.4 Use of Conformance Language

Conformance language is defined within this implementation guide at a high level, to ensure alignment to the multiple standards/specifications which have been harmonized. The use of conformance language within this document is limited to further constraints or relaxation of constraints on existing standards/specifications. New conformance language that specifically deviates from the underlying standard/profile is avoided wherever possible. Also, in those instances where new metadata is being defined, specific constraints are offered. Implementers should refer to the CDS Knowledge Artifact schema for the source for all conformance statements and rules.

Conformance language is defined throughout this implementation guide using **BOLD CAPS** to denote the conformance criteria to be applied. The conformance language that is used in this implementation guide is drawn from RFC 2219, and the conformance matrix offered for use in this implementation guide is drawn from the HL7 Implementation Guide for CDA® Release 2: IHE Health Story Consolidation, Release 1:

1) **SHALL/MUST**: an absolute requirement for all implementations of the Knowledge Artifact
2) **SHALL NOT**: an absolute prohibition against inclusion for all implementations of the Knowledge Artifact
3) **SHOULD/SHOULD NOT**: A best practice or recommendation to be considered by implementers within the context of their requirements to implement the Knowledge Artifact; there may be valid reasons to ignore an item, but the full implications must be understood and carefully weighed before choosing a different course
4) **MAY**: This is truly optional language for an implementation; can be included or omitted as the implementer decides with no implications

The Consolidated Conformance Verb Matrix included as part of the HL7 Implementation Guide for CDA® Release 2: IHE Health Story Consolidation, Release 1 (shown below) summarizes how the different standards/profiles are used within the implementation guide, and also lists specific recommendations used in this implementation guide:

RFC 2119	HL7	IHE	HITSP
SHALL Absolute requirement of the specification	SHALL Required/Mandatory	R (Required) Element must be present but can be NULL.	R (Required) Data elements must always be sent. A NULL can be sent.
SHALL NOT Absolute prohibition of the specification	SHALL NOT Not Required/Mandatory	-	-

© 2014 Health Level Seven International. All rights reserved

RFC 2119	HL7	IHE	HITSP
SHOULD Recommended There may exist valid reasons in certain circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.	SHOULD Best Practice or Recommendation	R2 (Required if known) The sending application must be able to demonstrate that it can send all required if known elements, unless it does not in fact gather that data. If the information cannot be transmitted, the data element contains a value indicating the reason for omission of the data.	R2 (Required if known) If the sending application has data for the data element, it is REQUIRED to populate the data element. If the value is not known, the data element need not be sent.
SHOULD NOT Not Recommended	SHOULD NOT Not Recommended	-	-
MAY Optional	MAY Accepted/Permitted	O (Optional)	O (Optional)
-	-	C (Conditional) A conditional data element is one that is required, required if known or optional depending upon other conditions.	C (Conditional) Required to be sent when the conditions specified in the HITSP additional specifications column are true.

TABLE 7 – IMPLEMENTATION GUIDE– CONFORMANCE VERB MATRIX

The use of the word "recommendation" is also used throughout this implementation guide, especially in light of the initial level of development being done on the harmonization of CDS standards and specifications into a CDS Knowledge Artifact schema. A recommendation is used to offer implementers flexibility in their CDS environments by recommending an approach to be followed, while not constraining in any way the use of alternative options. Recommendations are primarily used in those areas where the S&I Framework requests further implementation feedback from implementers and pilot sites prior to defining conforming criteria.

Optionality is further defined for implementers for each of the metadata elements that were outlined within the HeD Artifact Sharing Use Case in Section 11, using IHE guidelines:

Guideline Term	Definition
Required	Element must be present and CANNOT BE NULL (no NULL flavors allowed).
Required if Known	The sending system must be able to demonstrate that it can send all required elements within the CDS knowledge artifact, unless it does not gather that specific element or does not plan to include it. If the information cannot be transmitted in the CDS knowledge artifact, the data element contains a value indicating the reason for omission of the data from the knowledge artifact.
Optional	There is no need to include this element unless the implementer so desires.
Conditional	A conditional data element is one that is required, required if known or optional depending upon other conditions that may be present in the CDS environment. Implementers have some latitude to apply conditions to specific metadata or other data elements within the knowledge artifact that do not apply to their environment.

© 2014 Health Level Seven International. All rights reserved

TABLE 8 – DATA ELEMENT – OPTIONALITY LEVELS

1.5.4 Schema Roadmap

The approach used to document the CDS Knowledge Artifact schema is very similar to the approach used in the development of the HL7 Version 3 Domain Analysis Model: Virtual Medical Record for Clinical Decision Support (vMR-CDS), Release 1. The schema has been developed as a set of XSD files that is documented within this implementation guide, together with the datatypes for each of the schema elements.

A Google Code Repository is being used to host the schemas, example files, and documentation associated with this Implementation Guide. The location of this repository is:

http://code.google.com/p/health-e-decisions/

Section 2 CDS Knowledge Artifact Types of this guide covers the different knowledge artifact types supported by the CDS Knowledge Artifact schema. Section 3 of this guide details the different components that may comprise an artifact type, and Section 4 covers the expected semantics of an implementation of the HeD Schema expression language.

The following table depicts the alignment of schema files to CDS Knowledge Artifact Components:

Schema File Name	Supported Components
metadata.xsd	Metadata
base.xsd	Supporting Evidence Supporting Reference
expression.xsd	Expression
action.xsd	Action

TABLE 9 – ALIGNMENT OF SCHEMA FILES TO CDS KNOWLEDGE ARTIFACT COMPONENTS

Each of these components is used in different ways depending on the type of knowledge artifact being constructed. Constraints on the schema appropriate for each artifact type are defined in more detail in the following section.

© 2014 Health Level Seven International. All rights reserved

2 CDS KNOWLEDGE ARTIFACT TYPES

This implementation guide is designed to support each of the three CDS Knowledge Artifact types defined in the HeD Artifact Sharing Use Case. In this section, specific guidance on implementing each of these artifact types is defined, with specific emphasis on:

1) Describing a logical example of what the CDS Knowledge Artifact looks like
2) Defining the specific components of each knowledge artifact type
3) Identifying the required and optional components and their attributes for each knowledge artifact type – known as conformance statements

Each of the knowledge artifact types is structured in a format to allow for flexibility in implementation. This format allows for defining a base set of requirements for a specific artifact type, which may then be extended by implementers.

2.1 Conformance to the Health eDecisions Knowledge Artifact Specification

There are layers of constraints that must be met to conform to the HeD specification:

1) An artifact SHALL conform to the constraints specified in the HeD XML schema files. Thus, an HeD knowledge artifact must be well-formed and valid according to the HeD schema. These constraints are described in Section 5.
2) An artifact SHALL conform to the general constraints for HeD artifacts described in Section 2.2 and Section 4. Many of these constraints are specified in the form of schematron rules that may be used as an optional tool for validation.
3) An artifact SHALL conform to the constraints described for its specific type. These constraints are described in
4) Section 2.3 for ECA rule type artifacts
5) Section 2.4 for order set type artifacts
6) Section 2.5 for documentation template type artifacts
7) Each artifact SHALL also conform to the terminology specifications for its realm. The requirements for the US realm are described in the specification for **HL7 Virtual Medical Record for Clinical Decision Support (vMR-CDS) Templates, Release 1, Version 2.0.**

2.2 HeD Knowledge Artifacts

An HeD Knowledge Artifact is represented in an XML file following the structure specified by its schema and conforming to the constraints specified within this document. The root element of the artifact is knowledgeDocument. A knowledgeDocument element contains a single ECA rule, an order set, or a documentation template. This section provides an overview of the HeD Knowledge Artifact and general constraints on the artifacts. Subsequent sections describe each artifact type in detail and enumerate constraints that are specific to the respective artifact type.

2.2.1 Metadata Elements

Each knowledgeDocument contains a single metadata element. The metadata element supports knowledge management and the implementation of the artifact in a CDS system. The element contains information about the identity of the artifact (including its structure), its

© 2014 Health Level Seven International. All rights reserved

provenance, its lifecycle and status, and its applicability in the care context. The latter is specified in part in the applicability element. This element is used for indexing a knowledge artifact in a repository and as a guide to integrating the artifact into the appropriate system and workflow. This element is not used during the execution of the artifact. During execution, the conditions elements are used to determine if an artifact is applicable for the patient.

2.2.2 Def Elements

The externalData element contains any number of def elements, each of which provides a declarative specification of the data that are needed by this artifact for execution. Each def element maps a name to a data element using an expression of type ClinicalRequest (e.g., the name lastBodyTemperature maps to an ObservationResult object with specified LOINC codes and was the one most recently measured). The def name can be used in other expressions (lastBodyTemperature > 102 degrees Fahrenheit) to refer to this data element. Mappings to clinical data must be specified using the VMR as the data model. This is explained in Section 4.1 Clinical Data Retrieval in HeDS Artifacts. The externalData may also contain any number of parameterDef elements that can be used to specify parameters that may be passed into the artifact when an artifact is called by another artifact or system. For example, an artifact for diabetes control may specify a parameter that allows control over the length of time since an A1c test has occurred.

2.2.3 Expression Elements

The expressions element provides a mechanism to construct modular expressions that can improve reusability and readability of an artifact. That is expression elements defined here may be used elsewhere in the artifact where expressions are needed (e.g., in condition or inside other expressions). Each def element maps a name to an expression (e.g., the name elevatedLDL may be mapped to an expression, value of lastLDL > 100). The expression may be referenced by name in an expression of type ExpressionRef to construct more complex expressions (e.g., elevatedLDL and CHDRiskPercentAt10years > 20).

2.2.4 Trigger Elements

The triggers element allows definition of events (e.g., new serum potassium result available), such that when the event occurs the artifact is executed. A more detailed description of how a trigger element is constructed is provided in Section 3.3 Triggers and Events.

2.2.5 Condition Elements

The conditions elements contain logical criteria that evaluate to true or false and that control further execution of an artifact. The conditionRole element of a condition specifies how the criteria are to be used. In the current version of the schema, the only conditionRole is applicableScenario. When a condition with an applicableScenario role evaluates to true, the actions in the actionGroups element are executed. Conditions usually are based on the data items specified in the externalData element and represent the clinical situation (e.g., patient has diabetes and has not had a hemoglobin A1c element in the past six months).

2.2.6 Behavior Elements

Behaviors control how an artifact is presented and how the users may interact with an artifact. At the level of the knowledgeDocument, no behavior types are included as part of the HeD specification. However, a supplier may define behavior types that are applicable to an artifact.

2.2.7 Action Elements

The actionGroups element contains the “output” actions or the recommendations of the CDS. These may be in the form of messages (such as reminders), or structured clinical acts (e.g., a laboratory test order) that can be implemented via clinical systems such as a computerized provider entry system or a documentation system, or may create new events (e.g., declaration

© 2014 Health Level Seven International. All rights reserved

of a patient state such as failure to a treatment). Clinical actions must be specified using the VMR as the data model, as described in Section 3 CDS Knowledge Artifact Components. The action elements may be nested inside actionGroup elements (e.g., all medication-related actions may be aggregated in one actionGroup element). To enable reuse and modularity, an actionGroup may be incorporated by reference from another CDS artifact (e.g., a ventilator protocol may be defined in an artifact that is reused in different order sets). Elements of type actionGroups can have behaviors associated with them. These behaviors specify, for example, how a user may select from a set of actions in the group, or whether an action is required or optional.

In the next chapter, the components of the HeD artifacts are described in more detail. The remaining sections of this chapter describe the three different types of artifacts currently supported by HeD.

2.3 Event Condition Action (ECA) Rules

2.3.1 Conceptual Overview

As defined in the HeD Artifact Sharing Use Case, an event condition action rule is an artifact with the general syntax "on event, if condition is true, then do action." The event triggers the invocation of the rule. The condition is a logical test that, if satisfied or evaluates "true," causes an action. The action part consists of a set of operations to execute. These actions may in turn cause further events to occur, which may in turn cause other ECA rules to fire.

A typical rule can be represented in the HeD Knowledge Artifact schema as shown in the diagram below:

© 2014 Health Level Seven International. All rights reserved

FIGURE 2 – ECA RULE – CONCEPTUAL OVERVIEW

A system with this rule in its rule repository activates the rule when the triggering events occur. At that point, the conditions are evaluated. Conditions may reference external data and may be composed of expressions defined in the expressions section. If the condition is true, the actions specified in the "action" part of the rule are executed (represented in Section 3.2 Actionsof the implementation guide).

The action groups are the containers and organizers of the actions in an ECA rule. A rule typically has a single action group (top level section), but may have more. Conceptually, a set of actions in a rule could be considered a "mini order set" which is presented to a clinician at certain times and under certain conditions. As such, the actions may be structured hierarchically using action groups and behaviors to specify how the orders should be a shown to a provider, and to place restrictions on how a provider chooses from the available set of orders. It should be noted that this is just a conceptual example, and that not all actions are necessarily orders. For example, an action can be a creation of a new event that triggers another rule, a future encounter, or the creation of a state description of the patient.

The following example illustrates the conceptual structure of the rule:

Hemoglobin A1C Reminder for Patients with Poorly Controlled Diabetes

Adapted From: CDSC L3 Diabetes Mellitus Assessment Rule

Clinical Focus: Diabetes Mellitus

Trigger

7. Start of encounter in primary care clinic

Conditions

8. Patient is an adult, and
9. Patient has diabetes, and
10. Patient has not had a Hemoglobin A1C test in the last three months, and
11. Patient had an abnormal Hemoglobin A1C test sometime in the nine-month period before the last three months

Actions

12. Notify the provider
13. Order Hemoglobin A1C in the next three days

2.3.2 ECA Rule CONFORMANCE REQUIREMENTS

The following sub-sections describe how to use the HeD schema for expressing a rule and specific conformance constraints for ECA rules. All constraints specified are relative to the root knowledgeDocument element.

2.3.2.1 Metadata

Constraint ECA-1: The value attribute of the artifactType element in the metadata element SHALL be specified as "Rule".

```
<sch:pattern name="ECA-1: Artifact type is Rule">
    <sch:rule
context="/hed:knowledgeDocument/hed:metadata/hed:artifactType">
```


© 2014 Health Level Seven International. All rights reserved

```
                <sch:assert test="./@value='Rule'">
                        The value attribute of artifact type must be 'Rule'
                </sch:assert>
        </sch:rule>
</sch:pattern>
```

2.3.2.2 External Data

No constraints specific to ECA rules are specified for the externalData element of a knowledgeDocument.

2.3.2.3 Expressions

No constraints specific to ECA rules are specified for the expressions element of a knowledgeDocument.

2.3.2.4 Triggers

Even though ECA rules require a trigger, triggers intentionally are left as optional in conformance requirements for ECA rules. This is because the triggering events are often specific to a particular implementation based on organizational workflows or policies. Thus, triggers may not always be appropriate to define in a sharable ECA rule Knowledge Artifact and it might be better to add these at the time the rule is integrated into the CDS system.

Thus, no constraints specific to ECA rules are specified for the triggers element of a knowledgeDocument.

2.3.2.5 Conditions

An ECA rule shall always define a condition that determines if the specified actions in the actionGroups element must be executed. The following constraint applies to conditions:

Constraint ECA-2: There SHALL be exactly one condition element specified in the conditions element whose conditionRole is “ApplicableScenario”.

```
<sch:pattern name="ECA-2: One condition of type ApplicableScenario is included">
        <sch:rule context="/hed:knowledgeDocument">
                <sch:assert
test="count(hed:conditions/hed:condition/hed:conditionRole[@value
='ApplicableScenario'])=1">
                        Exactly one condition of type ApplicableScenario must be
present in a rule
                </sch:assert>
        </sch:rule>
</sch:pattern>
```

2.3.2.6 Behaviors

No constraints specific to ECA rules are specified for the behaviors element of a knowledgeDocument.

© 2014 Health Level Seven International. All rights reserved

2.3.2.7 ActionGroups

No constraints specific to ECA rules are specified for the actionGroups element of a knowledgeDocument.

2.4 Order Sets

2.4.1 Conceptual Overview

As defined in the HeD Artifact Sharing Use Case, an order set is a pre-defined and approved group of orders related to a particular clinical condition (e.g., hypertension treatment and monitoring) or stage of care (e.g., hospital admission to Coronary Care Unit). An order set is used as a checklist for the clinician when managing a patient with a specific condition. It is a structured collection of orders (or actions in the HeD schema) relevant to that condition and presented to the clinician in a computerized provider order entry system (CPOE).

The actions in an order set are typically organized hierarchically, as a set of sections, sub-sections, etc., with the actions themselves at the very bottom of the structure. In the HeD Knowledge Artifact schema, we generalize the notion of "section" and call it a "group." Each group and/or subgroup may have behavior indicators associated with it, e.g., the number of actions that can/should/must be selected from the group of actions.

FIGURE 3 – ORDER SET – CONCEPTUAL OVERVIEW

A clinician chooses an order set from a library of order sets in the CPOE system based on metadata associated with the order set, and some information about the patient (typically just the diagnosis). The clinician then selects orders to be placed for a particular patient.

© 2014 Health Level Seven International. All rights reserved

The following example shows the a partial snippet of an order set that is organized according to the conceptual structure shown above:

Heart Failure Admission to Med/Surg (partial order set)

Venue: InPatient

Population: Adult

Section: General

1) ADHERE Risk Model (click here to link to evidence)
2) Heart Failure Survival Score (click here to link to evidence)

Section: Activity (CHOOSE ONE)

1) Ambulate
2) Bed rest

Section: Nursing Orders

Sub-section: Assessments (CHOOSE ANY)

1) Cardiac Monitor
2) Measure blood pressure; orthostatic

Sub-section: Interventions (CHOOSE ANY)

1) Elevate head of bed
2) Urinary catheter initiation/management

Section: Medications

Sub-section: Angiotensin-Converting Enzyme Inhibitors (CHOOSE ONE)

1) For patients with diastolic heart failure who are intolerant to an ACE inhibitor, consider the use of an ARB
2) For patients with systolic heart failure who are hemodynamically stable and are intolerant to an ACE inhibitor due to cough, use an ARB
3) captopril 6.25 milligram orally 3 times a day
4) lisinopril 2.5 milligram orally once a day

Sub-section: Beta-Blockers (CHOOSE ONE)

1) For patients with diastolic heart failure and a previous MI, use a beta-blocker
2) For patients with diastolic heart failure, consider the use of a beta-blocker
3) For patients with systolic heart failure who are hemodynamically stable, use beta-blocker therapy (eg, bisoprolol, carvedilol, metoprolol extended release)
4) bisoprolol 1.25 milligram orally once a day
5) carvedilol 3.125 milligram orally 2 times a day
6) carvedilol phosphate SR 10 mg multiphase 24 hr cap 1 capsule orally once a day
7) metoprolol succinate SR 25 mg 24 hr tab 0.5 tablet orally once a day

© 2014 Health Level Seven International. All rights reserved

The HeD Knowledge Artifact schema can be used to construct an order set. Such an order set may be imported into the CPOE module of an electronic health record system.

2.4.2 Order Set Conformance Requirements

The following sub-sections describe how to use the HeD schema for expressing an order set and specific conformance constraints for order sets. All constraints specified are relative to the root knowledgeDocument element.

2.4.2.1 Metadata

Constraint OS-1: The value attribute of the artifactType element in the metadata element SHALL be specified as "Order Set".

```
<sch:pattern name="OS-1: Artifact type is Order Set">
    <sch:rule
context="/hed:knowledgeDocument/hed:metadata/hed:artifactType">
        <sch:assert test="./@value='Order Set'">
            The value attribute of artifact type must be 'Order Set'
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

2.4.2.2 External Data

External data are likely to be used only if an order set or its constituent actions have conditions or other expressions. Many order sets do not contain these, hence an order set may not include external data.

No constraints specific to order sets are specified for the externalData element of a knowledgeDocument.

2.4.2.3 Expressions

As noted above, order sets often do not contain any expressions. Expressions may be used when the order set or actions have patient-specific conditions.

No constraints specific to order sets are specified for the expressions element of a knowledgeDocument.

2.4.2.4 Triggers

No constraints specific to order sets are specified for the triggers element of a knowledgeDocument.

2.4.2.5 Conditions

The condition with type applicableScenario specifies that the order set **SHOULD** be used only if the condition expression evaluates to true at the point in time when the order set is used in the target system. If the desire is to express the clinical scenarios in which this order set is applicable without requiring a run-time evaluation of the condition, consider using the applicability elements of the metadata instead. This approach will allow the clinician to find an order set by browsing or searching for it in an order set repository.

No constraints specific to order sets are specified for the conditions element of a knowledgeDocument.

© 2014 Health Level Seven International. All rights reserved

2.4.2.6 Behaviors

The current version of the HeD schema does not provide any behaviors applicable at the top level of the order set. Furthermore, none of the behaviors included as part of the HeD schema may be used here. Behaviors that apply at the action-level are described within the following section on actionGroups.

No constraints specific to order sets are specified for the behaviors element of a knowledgeDocument.

2.4.2.7 ActionGroups

The actionGroups element contains the orders (represented as actions) which may be further organized into sub actionGroups. Behaviors may be specified at the level of the groups or the actions to specify the number of actions to be chosen, if an action should be prechecked (meaning it will be ordered, unless the clinician user unchecks that action), and whether an action must be executed. Actions, actionGroups, and behaviors are further described in Section 3.2 Actions.

The following constraints are defined for order sets:

Constraint OS-2: Order sets do not include functionality to modify or cancel an existing order. They also do not fire events. Accordingly, actions of type UpdateAction, RemoveAction FireEventAction, SHALL not be allowed in order sets.

```
<sch:pattern name="OS-2: Some action types are disallowed in order sets">
    <sch:rule context="//hed:simpleAction">
        <sch:assert test="not(@xsi:type ='FireAction' or
                              @xsi:type='UpdateAction' or
                              @xsi:type='RemoveAction')">
            An action of this type is not allowed in an order set
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

Constraint OS-3: Order sets SHALL only incorporate other artifacts via groupReference that are of type ‘Order Set’. A schematron rule is not available for this constraint.

2.5 Documentation Templates

2.5.1 Conceptual Overview

As defined in the HeD Artifact Sharing Use Case, a documentation template is a structured form for recording information on a patient into a set of pre-defined data slots. These templates are used to guide structured data entry within an EHR or other clinical information system. Some types of clinical documents that can be represented via the documentation template artifacts are encounter summaries, procedure notes, patient-reported outcomes, and flowsheets.

A documentation template is a structured collection of what are known variously as documentation concepts, form elements, or observation items. Each documentation concept (the moniker used in the HeD schema) also can be thought of as a question to the user entering the data. Elements within the documentation concept guide and constrain the user’s

© 2014 Health Level Seven International. All rights reserved

responses, for example, a list from which to choose an answer, whether an answer is a number, a date, or some other type, and the cardinality of the answer.

Documentation concepts are contained in an action of type CollectInformationAction. This enables these concepts to be presented to the user conditionally (e.g., to ask questions appropriate to a patient's gender or to ask questions based on other responses), to compute responses for a concept based on previous responses or data from an EHR score (e.g., a risk score), and to bind the responses into expressions that can drive logic elsewhere in the documentation template (e.g., ask questions conditionally as described above). More details on the use of CollectInformationAction and documentation concepts are provided in Section 3.2.1.2 Collecting Information.

The documentation concepts in a template are typically organized hierarchically, into sections and sub-sections with the concepts themselves at the very bottom of the structure. As described in the overview of order sets, in HeD Knowledge Artifact schema these "sections" are called actionGroups. Similar to order sets, actionGroups in documentation templates may have behavior indicators associated with it, e.g., whether a documentation concept must have a response.

The figure below illustrates the conceptual structure of a documentation template.

FIGURE 4 – DOCUMENTATION TEMPLATE – CONCEPTUAL OVERVIEW

The example below shows a documentation template organized according to the conceptual structure of the figure above.

History and Physical Exam for Headache Venue: Office

© 2014 Health Level Seven International. All rights reserved

Population: Adult

Section: Symptoms

1) Chief complaint (text, response required)
2) Duration (time interval)
3) Triggering factors (text)

Section: Physical Exam

Sub-section: Neurological Exam

1) Pupil (pick list: reacting to light, equal)
2) Tremors (boolean)

Sub-section: Vital signs

1) Heart rate (number)
2) Temperature (number)
3) Temperature location (pick list: oral, axillary, surface)

2.5.2 Documentation Template Conformance Requirements

The following sub-sections describe how to use the HeD schema for expressing a documentation template and specific conformance constraints for documentation templates. All constraints specified are relative to the root knowledgeDocument element.

2.5.2.1 Metadata

Constraint DOC-1: The value attribute of the artifactType element in the metadata element SHALL be specified as "Documentation Template".

```
<sch:pattern name="DOC-1: Artifact type is Documentation Template">
    <sch:rule
context="/hed:knowledgeDocument/hed:metadata/hed:artifactType">
        <sch:assert test="./@value='Documentation Template'">
            The value attribute of artifact type must be 'Documentation
Template'
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

2.5.2.2 External Data

External data are likely to be used only if a documentation template has conditions or other expressions. If a documentation template does not contain these, then it may not include external data.

No constraints specific to documentation template are specified for the externalData element of a knowledgeDocument.

© 2014 Health Level Seven International. All rights reserved

2.5.2.3 Expressions

Expressions may be used when the documentation template or the documentation concepts have patient-specific conditions.

No constraints specific to documentation templates are specified for the expressions element of a knowledgeDocument.

2.5.2.4 Triggers

No constraints specific to documentation templates are specified for the triggers element of a knowledgeDocument.

2.5.2.5 Conditions

The condition with type applicableScenario specifies that the documentation template **SHOULD** be used only if the condition expression evaluates to true at the point in time when the order set is used in the target system. As with order sets, if the desire is to express the clinical scenarios in which this documentation template is applicable without requiring a run-time evaluation of the condition, consider using the applicability elements of the metadata instead. This approach will allow the user to find a template by browsing or searching for it in a template repository.

No constraints specific to documentation template are specified for the conditions element of a knowledgeDocument.

2.5.2.6 Behaviors

The current version of the HeD schema does not provide any behaviors applicable at the top level of the documentation template. Furthermore, none of the behaviors included as part of the HeD schema may be used here.

No constraints specific to documentation template are specified for the behaviors element of a knowledgeDocument.

2.5.2.7 ActionGroups

The actionGroups element contains the CollectInformationActions which contain the documentationConcept. The actionGroups element may also contain sub actionGroups or other documentation templates by references.

Behaviors may be specified at the level of the groups or the actions to specify the concepts to be documented. Actions, actionGroups, and behaviors are further described in Section 3.2 Actions.

The following constraints are defined for documentation templates:

Constraint DOC-2: Documentation concepts do not include functionality to cancel an existing order or remove records. Accordingly, an action of type RemoveAction SHALL NOT be included in documentation templates.

```
        <sch:pattern name="DOC-2: Some action types are disallowed in documentation
templates">
                <sch:rule context="//hed:simpleAction">
                        <sch:assert test="not(@xsi:type =@xsi:type='RemoveAction')">
                                An action of this type is not allowed in a documentation
template
                        </sch:assert>
```


© 2014 Health Level Seven International. All rights reserved

```
            </sch:rule>
        </sch:pattern>
```

Constraint DOC-3: Documentation templates do not allow creation or modification of existing orders. Therefore “proposal” type elements from the VMR SHALL NOT be used within the actionSentence of a CreateAction or UpdateAction. A schematron rule is not available for this constraint.

Constraint DOC-4: Documentation templates SHALL only incorporate other artifacts via groupReference that are of type ‘Documentation Template’. A schematron rule is not available for this constraint.

Constraint DOC-5: Pre-check behavior for actions applies for clinical orders. Since documentation concepts do not include orders, precheck behaviors are disallowed. That is a CollectInformationAction SHALL NOT include a behavior of type Precheck.

```
        <sch:pattern name="DOC-5: CollectInformationAction shall not incorporate precheck behavior">
            <sch:rule context="//hed:simpleAction[@xsi:type='CollectInformationAction']/hed:behaviors/hed:behavior">
                <sch:assert test="not(@xsi:type='PrecheckBehavior')">
                    An CollectInformationAction in a documentation template may not include a precheck behavior
                </sch:assert>
            </sch:rule>
        </sch:pattern>
```

2.6 Value Sets and Vocabulary Codes

The following sections explain guiding principles that Value Sets and Terminologies Work Group used in aligning vocabulary codes and value sets to CDS artifact data elements. The guiding principles for aligning coding systems and value sets to data elements are in line with vocabularies and value sets recommended by the CMS Blueprint for eMeasure Specifications. These specifications and full details of the value sets and vocabulary codes briefly described below can be found in the HeD Value Sets and Terminology Implementation Guide.

2.6.1 Value Set Use Case

If a value set is defined for HeD, then the value set must be used in order for end users to be considered conformant with HeD. Therefore, a valid use case should exist before a value set is defined for a data element.

2.6.2 Coded Data versus Free Text

When there is general agreement across stakeholders regarding the semantic meaning of coded concepts (regardless of the actual descriptions) as well as in the importance of exchanging these defined concepts, it is worthwhile to encode a value set. An example is the concept of Frequency for substance administration. There would be unanimous agreement that 1x/day, 2x/day, 3x/day, etc. would make sense as coded values, even though there may be variation in the way these concepts are described (e.g., QD, BID, TID, QID).

© 2014 Health Level Seven International. All rights reserved

On the other hand, if a shared consistency of the concepts that comprise a value set is not important, it may be better to NOT define a value set for that data element. An example is the concept of named departments within a hospital. Due to different departmental structures across healthcare organizations, it may not make sense to try to capture all of the different departmental concepts because of differences in granularity which may result in overlap of concepts, gaps for some organizations, and irrelevant coded values for others, resulting in little benefit from an attempt to determine a common set of agreed-upon representative concepts.

© 2014 Health Level Seven International. All rights reserved

3 CDS KNOWLEDGE ARTIFACT COMPONENTS

This chapter describes the components of a Knowledge Artifact, how they are used, and the constraints that apply to those components.

3.1 Metadata

MET-1: The metadata element of the knowledgeDocument MUST specify the schemaIdentifier to have “urn:hl7-org:v3:knowledgeartifact:r1” as the value of the root attribute.

```
<sch:pattern name="MET-1: Schema identifier">
    <sch:rule
context="/hed:knowledgeDocument/hed:metadata/hed:schemaIdentifier">
        <sch:assert test="./@root='urn:hl7-org:v3:knowledgeartifact:r1'
            The schemaIdentifier root value must be ‘urn:hl7-
org:v3:knowledgeartifact:r1’
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

3.1.1 Clinical Data Models

The HeD Schema specification itself does not reference any specific data model, and so can be used to create artifacts that capture logic expressed against any clinical data model. These data models are specified in the dataModels element of the artifact metadata.

NOTE: To enable sharing of artifacts, this implementation guide uses the vMR exclusively to represent clinical data.

For example, the following data models snippet specifies that the artifact will use the vMR:

```
<dataModels>
    <modelReference>
        <description value="Virtual Medical Record model" />
        <referencedModel value="'urn:hl7-org:v3:vmr:r2"/>
    </modelReference>
</dataModels>
```

An artifact can reference any number of data models, so long as they are all defined within the dataModels element.

In addition to specifying the actual namespace of the data model within the dataModels element, the XML artifact must specify a local name for use in referencing the data model within the logic of the artifact. For example, the following namespace declaration in the root element specifies the local namespace for the vMR data model:

xmlns:vmr="'urn:hl7-org:v3:vmr:r2"

© 2014 Health Level Seven International. All rights reserved

With this namespace prefix defined, the types defined in the referenced schema can now be referenced within expression logic in the artifact. For example, the following snippet uses this local namespace prefix to define an external data request in terms of vMR types:

```
<def name="antithromboticNotPrescribedForDocumentedReason">
        <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                dataType="vmr:ObservationResult" codeProperty="observationFocus.code"
                dateProperty="observationEventTime">
                <description>Patient reason or other reason for not prescribing an
antithrombotic</description>
                <codes xsi:type="List">
                        <element xsi:type="CodeLiteral" code="G8697"
                                codeSystem="2.16.840.1.113883.6.12"
codeSystemName="CPT-4"
                                displayName="" />
                </codes>
        </expression>
</def>
```

This clinical request specifies that the type of data being requested is "vmr:ObservationResult". During semantic verification, this type is resolved to the data model reference specified for the artifact.

For more information on specifying external data requirements for an artifact, please refer to the the Expressions discussion in Section 3.4.14 External Data.

3.2 Actions

Actions are one of the core components of CDS that define the recommended operations to be executed in the specified clinical context. These can include creating orders for medications or diagnostic tests, collecting information, modifying or canceling an existing action, preventing a proposed action from occurring, or creating a new event that can trigger other CDS artifacts.

3.2.1 Types of Actions

Actions are included in an artifact by adding an element simpleAction as a subElement to any element of type ActionGroup. The simpleAction element is of an abstract type called AtomicAction. The following concrete types of AtomicAction are defined in the schema and further described in Section 3.2 Actions:

1) CreateAction
2) RemoveAction
3) UpdateAction
4) FireEventAction
5) CollectInformationAction
6) DeclareResponseAction

© 2014 Health Level Seven International. All rights reserved

3.2.1.1 Working with Clinical Data and Actions

The first three types of actions, respectively, create, remove, or update a clinical data object. These objects are specified as the actionSentence element of type Expression. The actionSentence expression returns a single object that is a subtype of ClinicalStatement from the VMR (Section 3.2.2 Creating Action Sentence Expressions).

ACT-2: The expression specified in an actionSentence element of a CreateAction, UpdateAction, or RemoveAction MUST evaluate to an object of a subtype of ClinicalStatement from the **HL7 vMR Logical Model Release 2, Version 3.0** specification.

In the case of a createAction, the object returned by the actionSentence is a new one, i.e., one not representing existing data or orders from the patient records. The new object may be a proposal to carry out a new clinical action (such as starting a medication, or conducting a procedure). These are represented as vMR proposal statements (Table 10). The new object may be an inference about the patient's clinical state or a patient history element, which would be constructed as an object of the "Event" type from vMR. An actionSentence element is defined using an expression of type ObjectExpression, ObjectRedefine, or ComplexLiteral.

AppointmentProposal
CommunicationProposal
ComplexIVProposal
DietProposal
EnteralFeedingProposal
GoalProposal
ImagingProposal
LaboratoryProposal
ObservationProposal
PCAProposal
ProcedureProposal
RespiratoryCareProposal
SubstanceAdministrationProposal
SubstanceDispensationProposal
SupplyProposal

TABLE 10. PROPOSAL STATEMENT TYPES IN THE VMR

The actionSentence expression in removeAction and updateAction returns existing data or orders about the patient. These actionSentence objects will refer to an object that was specified in the externalData element, using its def element.

3.2.1.2 Collecting Information

The CollectInformationAction type is used to obtain any information from the user. This type of action is most frequently used as part of the documentation template artifact type. An element called documentationConcept of type DocumentationItem specifies the details of the information to be obtained. The DocumentationItem type contains elements to describe the text to be displayed to the user, terminology codes, the data type and cardinality of the response, and the allowed range of values for the response.

The initialValue element of the CollectInformationAction type is used to optionally specify a value for the documentationConcept element that can be modified by the user. Since the initialValue is of type Expression, the value can be a simple literal value, a value computed from other responses entered by the user in the documentation template (e.g., when

© 2014 Health Level Seven International. All rights reserved

computing a risk score), a value derived from data in the patient records (e.g., question about use of anti-depressants might be responded from the medical record), or a combination of the above. An example of a CollectInformationAction with an initialValue expression is shown in Appendix E, FLACC Example. In this example, the value of the pain scale is computed based on user entered values for previous items.

In order to allow logic to be written that accesses the values entered by the user, the HeD Schema allows documentation items to be "bound" to a container. The container must be declared using a DeclareResponseAction. By default, the container is named "Responses", but the schema allows any number of response containers to be created, so long as they are all uniquely named. The responseBinding element can then be used to designate a name for the value entered by the end-user. This name, once defined via a responseBinding element, will then be available as a property of the responses container. An example of binding is shown in Appendix E, FLACC Example. An example of a response container declaration is shown in Appendix E, FLACC Example. The value of the property can then be used in an expression in some other action by referring to the name of the property within the response container. For example and as mentioned above, the value can be used to compute the initial value of another documentationConcept element, or as part of an actionSentence expression in createAction.

3.2.1.3 Creating Events

The FireEvent action can be used to create an event and place it in the working memory of the CDS system. The types of events that can be fired are specified in the enumeration DataEventType. A payload associated with the event is specified as an actionSentence. The event that is fired can be the trigger for another CDS artifact, as described in Section 3.3 Triggers and Events .

3.2.2 Creating Action Sentence Expressions

Action sentences for new objects in the CreateAction and FireEventare created by constructing an object in a specified data model. The CDS Knowledge Artifact schema does not include a model of health data about a patient. However, this implementation guide specifies that HeD knowledge artifacts will use the vMR (instead the dependency is indirect) as the data model for a patient's health data. There are two methods to specify action sentences using the vMR.

The first approach uses an expression of type ObjectExpression. In an ObjectExpression, the action sentence specifies the type of the Object (a concrete subtype of ClinicalStatement from the vMR) in the ObjectType attribute, and constraints on the values of properties of that type in the property elements. Since the constraints are specified as expressions, these can be literal values (5 mg), relative values (e.g., three days after surgery), or ranges (50 to 100 mg). An example of ObjectExpression is shown in Appendix E, RespiratoryOrder.

ACT-3: The objectType attribute of an ObjectExpression specifying an actionSentence must specify it as ClinicalStatement or one of its subtypes from the vMR.

An alternative approach is to use the ComplexLiteral expression. In this case, the vMR XML schema can be leveraged to specify the new object. The ComplexLiteral provides a more compact and simpler method to specify objects, when the properties can be specified as literal values, a common scenario in order sets. This is because the vMR schema does not allow ranges or constraints for many property values (e.g., the date of a Procedure). An example of ComplexLiteral is shown in Appendix E, DopamineComplexIVOrderWithComplexLiteral.

© 2014 Health Level Seven International. All rights reserved

ACT-4: The type of the value element of a ComplexLiteral element used as the source element of an actionSentence of type ObjectRedefine must be ClinicalStatement or one of its subtypes from the vMR.

When an action will be evaluated within a host environment (i.e., not within the CDS engine), an ObjectDescriptor may be used as the source of the ComplexLiteral expression. An ObjectDescriptor identifies an object within the host environment, and one which may or may not be available in the CDS engine. Hence, an ObjectDescriptor expression is not evaluated in the CDS engine. A typical use of ObjectDescriptor may be when an actionSentence is part of a CDS response in a decision-support service. Hence, an ObjectDescriptor must not be used in an artifact that is evaluated in an embedded CDS engine.

3.2.3 Organizing Actions

Elements of the type ActionGroup can be used to organize the actions into logical and visual groupings. An ActionGroup type object incorporates its constituents in an element called subElements. There different types of constituents can be included:

1) An action (sub) group that is specified in place in this artifact (specified in the group element).
2) A reference to an action group. This functionally is a subgroup but resides in an artifact by itself (specified in the groupReference element).
3) Actions that are described in the previous sections (specified in the simpleAction element).

An action group may constitute a visual section of an order set (e.g., Medications section or Diets section) or of a documentation template (e.g., History of Present Illness section, Cardiovascular System Examination section) that are displayed with each other. The title of a visual group is the title or the heading of the section. An action group may also provide logical or functional relationship amongst the actions in a group (e.g., a set of orders which must be ordered all together or not at all). Such relationships are specified as behaviors. These are further described in Section 3.2.5.

Group references, specified by using the groupReference element, enable the construction of modular, reusable knowledge artifacts. A reusable group should be defined as a self-contained artifact in its own Knowledge Document. This group can then be used in another artifact by using the groupReference element and specifying the identifier from the metadata of the referenced order set. For example, a DVT Prophylaxis group may be defined as an order set artifact. This order set artifact may be included in order sets for clinical scenarios which call for the use of DVT prophylaxis. An example of groupReference is shown in Appendix E, HeartFailureAdmissionToMedSurgOrder. Similarly, a Smoking History documentation template artifact can be reused in a variety of documentation templates for visit notes based on different presenting conditions.

3.2.4 Conditional Actions

Actions may have conditional expressions associated with them such that the action is executed only if given condition is true. Similarly, a group may have conditions associated with it, so that actions contained in it and its subgroups are executed only if the specified condition is true. This can be used to show orders selectively in an order set or to show appropriate documentation concepts in a documentation template (e.g., do not show prostate exam questions for female patients). The role type of these conditions is ApplicableScenario. An example of applicableScenario is shown in Appendix E, HeartFailureAdmissionToMedSurgOrderSet.

© 2014 Health Level Seven International. All rights reserved

Constraint ACT-1: An action or an action group MUST have at the most one condition with the condition role ApplicableScenario.

```
<sch:pattern name="ACT-1a: Actions have at most one condition with ApplicableScenario role">
    <sch:rule context="//hed:simpleAction/hed:conditions">
        <sch:assert test="count(hed:condition/hed:conditionRole[@value ='ApplicableScenario'])&lt;=1">
            Exactly one condition of type ApplicableScenario MUST be present in an action
        </sch:assert>
    </sch:rule>
</sch:pattern>


<sch:pattern name="ACT-1b: Action groups have at most one condition with ApplicableScenario role">
    <sch:rule context="//hed:actionGroup/hed:conditions">
        <sch:assert test="count(hed:condition/hed:conditionRole[@value ='ApplicableScenario'])&lt;=1">
            Exactly one condition of type ApplicableScenario MUST be present in an action group
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

3.2.5 Behaviors

Behaviors may be specified for knowledgeDocument, groups and for actions. However, the current version of the HeD schema does not specify particular behaviors at the level of the knowledgeDocument. In fact, the behaviors defined in the HeD specification shall not be used within the behaviors element on the knowledgeDocument. However, CDS Knowledge Artifact suppliers MAY define behaviors that can be used at the knowledgeDocument level.

The following constraints are specified for behaviors at the knowledgeDocument level:

Constraint ECA-3: The behaviors element SHALL NOT contain a behavior element of the types GroupSelectionBehavior, PrecheckBehavior, RequiredBehavior, or GroupOrganizationBehavior.

```
<sch:pattern name="BHV-9: None of the HeD predefined behaviors are used at the knowledgeDocument level">
    <sch:rule context="/hed:knowledgeDocument/hed:behaviors/hed:behavior">
        <sch:assert test="not(@xsi:type ='GroupSelectionBehavior' or


@xsi:type='PrecheckBehavior' or
```


© 2014 Health Level Seven International. All rights reserved

```
        @xsi:type='RequiredBehavior' or

        @xsi:type='GroupOrganizationBehavior')">
                                A behavior of this type is not allowed at the level of the
knowledgeDocument
                        </sch:assert>
                </sch:rule>
        </sch:pattern>
```

There are two types of behaviors which apply exclusively to groups: GroupSelectionBehavior and GroupOrganizationBehavior. The GroupSelectionBehavior specifies the number of items in the group that should be chosen by the recipient of the CDS. The selection options are listed in the enumeration GroupSelectionBehaviorType (e.g., all or none, exactly one, at most one).

The GroupOrganizationBehavior provides hints to the end user system to aid with display of the group's subElements. The behavior is specified by selecting from the enumeration GroupOrganizationBehaviorType. A VisualGroup indicates a group of items displayed together, perhaps as a subsection with a title and description. A LogicalGroup indicates a group of items that are logically related, but do not need to be separated visually from other items. A SentenceGroup indicates group of items which share a common orderable (such as "aspirin"), but differ in the details of their administration (e.g., dose level, frequency, route of administration). The end user system may choose to separate out the common orderable items, and show the order details underneath, or could decide to show each item as a separate orderable in and of itself. An example of GroupSelectionBehavior is shown in Appendix E, RespiratoryOrder. An example of GroupOrganizationBehavior is shown in Appendix E, HeartFailureAdmissionToMedSurgOrderSet.

The following constraints are specified for group behaviors:

Constraint BHV-1: A behavior of type GroupSelectionBehavior MUST be specified as a behavior of an actionGroup element only. It MUST NOT be used with a simpleAction element.

```
        <sch:pattern name="BHV-1: GroupSelectionBehavior can only be used with action
groups">
                <sch:rule context="//hed:behavior[@xsi:type='GroupSelectionBehavior']">
                        <sch:assert test="name(../..)='actionGroup'">
                                GroupSelectionBehavior MUST be specified under action
groups only
                        </sch:assert>
                </sch:rule>
        </sch:pattern>
```


© 2014 Health Level Seven International. All rights reserved

Constraint BHV-2: A behavior of type GroupSelectionBehavior MUST be specified as a behavior of an actionGroup element only. It MUST NOT be used with a simpleAction element.

```
<sch:pattern name="BHV-2: GroupOrganizationBehavior can only be used with action groups">
    <sch:rule context="//hed:behavior[@xsi:type='GroupOrganizationBehavior']">
        <sch:assert test="name(../..)='actionGroup'">
            GroupOrganizationBehavior MUST be specified under action groups only
        </sch:assert>
    </sch:rule>
</sch:pattern>
```

Constraint BHV-4: An actionGroup element with a behavior of type GroupOrganizationBehavior and value of VisualGroup SHOULD have a title specified.

```
<sch:pattern name="BHV-4: An action group with GroupOrganizationBehavior of VisualGroup SHOULD have a non-empty title">
    <sch:rule context="//hed:behavior[@xsi:type='GroupOrganizationBehavior' and hed:value='VisualGroup']">
        <sch:report test="not(../../hed:title/@value)">
            An action group with GroupOrganizationBehavior of VisualGroup MUST have a non-empty title
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Constraint BHV-5: Any actions specified as a subElement in an actionGroup with behavior of type GroupOrganizationBehavior and value of SentenceGroup MUST NOT specify a behavior of type RequiredBehavior.

```
<sch:pattern name="BHV-5: Sub-elements of an action group with Group Organization Behavior of SentenceGroup MUST NOT specify Required Behavior.">
    <sch:rule context="//hed:behavior[@xsi:type='GroupOrganizationBehavior' and @value='SentenceGroup']">
        <sch:assert test="count(../../hed:subElements/hed:simpleAction/hed:behaviors/hed:behavior[@xsi:type='RequiredBehavior'])=0">
            Sub-elements of an action group with Group Organization Behavior of SentenceGroup MUST NOT specify Required Behavior
```


© 2014 Health Level Seven International. All rights reserved

```
                    </sch:assert>
            </sch:rule>
        </sch:pattern>
```

Constraint BHV-6: An actionGroup element with a behavior of type GroupOrganizationBehavior and value of SentenceGroup and also having a behavior of type GroupSelectionBehavior MUST specify the value of the latter as AtMostOne or ExactlyOne.

```
<sch:pattern name="BHV-6: An action group with Group Organization Behavior of
SentenceGroup MUST specify a GroupSelectionBehavior of AtMostOne or ExactlyOne">
            <sch:rule
context="//hed:actionGroup/hed:behaviors/hed:behavior[@xsi:type='GroupOrganizationBeha
vior' and @value='SentenceGroup']">
                    <sch:assert test="if
(../hed:behavior[@xsi:type='GroupSelectionBehavior']) then
../hed:behavior[@value='AtMostOne' or @value='ExactlyOne'] else 'true'">
                        An action group with Group Organization Behavior of
SentenceGroup MUST specify a GroupSelectionBehavior of AtMostOne or ExactlyOne
                    </sch:assert>
            </sch:rule>
        </sch:pattern>
```

Constraint BHV-7: The subElement of an actionGroup element with a behavior of type GroupOrganizatioBehavior and value of SentenceGroup MUST only contain simpleAction elements.

```
<sch:pattern name="BHV-7: Sub-elements of an action group with Group Organization
Behavior of SentenceGroup MUST be simple actions only">
            <sch:rule
context="//hed:actionGroup/hed:behaviors/hed:behavior[@xsi:type='GroupOrganizationBeha
vior' and @value='SentenceGroup']">
                    <sch:assert test="count(../../hed:subElements/hed:simpleAction) =
count(../../hed:subElements/*)">
                        Sub-elements of an action group with Group Organization
Behavior of SentenceGroup MUST be simple actions only
                    </sch:assert>
            </sch:rule>
        </sch:pattern>
```


© 2014 Health Level Seven International. All rights reserved

Constraint BHV-8: The subElement of an actionGroup element with a behavior of type GroupSelectionBehavior and value of AllOrNone or ExactlyOne or AtMostOne MUST NOT contain any actions or groups in its subElements that have a behavior of type RequiredBehavior and value of Must.

```
<sch:pattern name="BHV-8: Group Selection Behavior and sub-elements whose Required Behavior is Must">
        <sch:rule context="//hed:behavior[@xsi:type='GroupSelectionBehavior' and (@value='AllOrNone' or @value='AtMostOne' or @value='ExactlyOne')]">
            <sch:assert test="count(../../hed:subElements/*/hed:behaviors/hed:behavior[@xsi:type='RequiredBehavior' and @value='Must'])=0">
                An action group with Group Selection Behavior of AllOrNone, ExactlyOne, AtMostOne MUST NOT have any sub-elements whose Required Behavior is Must
            </sch:assert>
        </sch:rule>
    </sch:pattern>
```

The RequiredBehavior is typically used at the action level, but may be used at the group level. RequiredBehavior specifies whether a given item or a group of items is optional, must be executed, or must be executed unless documentation is provided saying why it was not ordered. Such behavior assumes that a whole group of actions may be chosen as a single unit. An example of RequiredBehavior is shown in Appendix E, HeartFailureAdmissionToMedSurgOrderSet.

The PrecheckBehavior type is used exclusively with Actions. This type of behavior indicates if an action should be checked or selection for execution when presented to the CDS recipient. This is a means for the CDS system to make it convenient for the user to execute that action or order, by saving the user the effort required to selection that action. This approach is used typically in order sets for orders that are either placed commonly or that the organization prefers (such as those that might be based on evidence). An example of PrecheckBehavior is shown in Appendix E, HeartFailureAdmissionToMedSurgOrderSet.

Constraint BHV-3: A behavior of type PrecheckBehavior MUST be specified as a behavior of a simpleAction element only. It MUST NOT be used with an actionGroup element.

```
    <sch:pattern name="BHV-3: PrecheckBehavior can only be used with actions">
        <sch:rule context="//hed:behavior[@xsi:type='PrecheckBehavior']">
            <sch:assert test="name(../..)='simpleAction'">
                PrecheckBehavior MUST be specified under actions only
            </sch:assert>
        </sch:rule>
    </sch:pattern>
```


© 2014 Health Level Seven International. All rights reserved

3.3 Triggers and Events

An event specifies the signal that triggers the invocation of an artifact. The knowledge artifact schema defines two types of triggering events:

- DataEvent – Specifies the data that, when changed, should trigger the event.
- PeriodicEvent – Specifies that the event is triggered on a periodic schedule.

Implementers MAY create additional Event components that extend the existing components defined in this implementation guide. See Section 3.5 Extending the Schema on how the schema can be extended.

3.3.1 DataEvent

The DataEvent trigger allows the artifact to define the types of data that, when changed, should trigger the event. For a DataEvent, the expression element of the trigger is expected to be of type ExpressionDef, and is expected to reference an expression defined in the externalData section of the artifact.

The referenced expression is expected to be a ClinicalRequest which defines the type of the triggering data. In addition to defining the type of the data, the ClinicalRequest is expected to define the triggering event, (Added, Modified, Removed, Accessed, or AccessEnded), as well as an optional relative time offset to allow for delayed triggering of the rule. For more information, refer to the documentation for the ClinicalRequest type in the expression section.

When an ExpressionDef element in the externalData section defines a trigger event, then the data for that element is supplied by the triggering event. Furthermore, if the artifact was executed due to a different triggering event, then the data specified in a given ExpressionDef element that was not associated with the triggering event is not queried for and is not available in the execution context.

3.3.2 PeriodicEvent

The PeriodicEvent trigger allows the knowledge artifact to be triggered on a schedule. The expression element of the trigger definition is expected to be a PeriodicInterval expression defining the period at which the knowledge artifact should be triggered.

3.4 Expressions

The HeD Schema expression component allows clinical decision support logic to be represented at various points within the artifact. The intent of this component is to ensure that expression logic within an artifact can be unambiguously specified so that it can be shared effectively.

Toward this end, the expressions within an HeD Schema artifact are represented at the syntax tree level. This form simplifies language processing tasks such as semantic verification and translation, while also allowing a human readable format that removes any potential ambiguity that would be introduced by a more prose-like representation.

This section provides several specific examples of logical expressions within HeD example artifacts to help guide content authors and implementers in expressing and understanding clinical decision support logic.

© 2014 Health Level Seven International. All rights reserved

3.4.1 Basic Values (Scalars)

Most expressions will at some point involve basic values such as integers and strings. For example, comparisons of patient age, or encounter time. These values are referred to as literals when they appear within an HeD expression, and there are several expression types defined that allow them to be represented.

The simplest expression for literals within HeD is the *Literal* expression. This expression can be used to select values of all the basic (scalar) types. For example:

```
<expression xsi:type="Literal" valueType="xsi:int" value="6"/>
<expression xsi:type="Literal" valueType="xsi:decimal" value="8.2"/>
<!-- NOTE: The date format for this literal is defined by the XSD standard -->
<expression xsi:type="Literal" valueType="xsi:date" value="2010-10-10"/>
<expression xsi:type="Literal" valueType="xsi:string" value="Patient is on
antithrombotic."/>
```

In addition to the generic literal, the HeD expression schema defines literal expressions specifically for each supported type, including the most common ISO 21090 data types. For example:

```
<expression xsi:type="IntegerLiteral" value="20"/>
<expression xsi:type="StringLiteral" value="Patient is less than 20 years old."/>
<!-- NOTE: The date format for this literal is defined by the ISO-21090 standard -->
<expression xsi:type="TimestampIntervalLiteral" low="20120101" high="20121231"/>
<expression xsi:type="PhysicalQuantityLiteral" value="12" unit="mg"/>
<expression xsi:type="CodeLiteral"
        code="G8697"
        codeSystem="2.16.840.1.113883.6.12"
        codeSystemName="CPT-4"
        displayName="" />
```

The following table gives a complete listing of the Literal expressions available within the HeD Specification:

Expression	Description
Literal	Generic literal for scalar types.
AddressLiteral	Returns a value of type AD with the given attributes.
BooleanLiteral	Returns a value of type BL with the given attributes.
CodeLiteral	Returns a value of type CD with the given attributes.
CodedOrdinalLiteral	Returns a value of type CO with the given attributes.

© 2014 Health Level Seven International. All rights reserved

Expression	Description
SimpleCodeLiteral	Returns a value of type CS with the given attributes.
EntityNameLiteral	Returns a value of type EN with the given attributes.
IdentifierLiteral	Returns a value of type II with the given attributes.
IntegerLiteral	Returns a value of type INT with the given attributes.
IntegerIntervalLiteral	Returns a value of type IVL_INT with the given attributes.
PeriodLiteral	Returns a value of type PIVL_TS with the given attributes.
PhysicalQuantityLiteral	Returns a value of type PQ with the given attributes.
QuantityIntervalLiteral	Returns a value of type IVL_QTY with the given attributes.
RealIntervalLiteral	Returns a value of type IVL_REAL with the given attributes.
RealLiteral	Returns a value of type Real with the given attributes.
StringLiteral	Returns a value of type ST with the given attributes.
TimestampLiteral	Returns a value of type TS with the given attributes.
TimestampIntervalLiteral	Returns a value of type IVL_TS with the given attributes.
UrlLiteral	Returns a value of type TEL with the given attributes.

3.4.2 Comparison Operators

The HeD Schema expression language defines a standard set of comparison operators for use with scalar values. Each comparison operator takes two arguments of the same type, and returns a boolean indicating the result of the comparison. Note that for comparison operators, if either or both operands evaluate to null, the result of the comparison is *unknown*, not false.

The following examples illustrate the use of comparison operators:

```
<condition xsi:type="Equal">
        <operand xsi:type="Property" path="Status"/>
        <operand xsi:type="Literal" valueType="xsi:string" value="Active"/>
</condition>

<when xsi:type="LessOrEqual">
        <operand xsi:type="Property" path="demographics.age"/>
        <operand xsi:type="IntegerLiteral" value="20"/>
```


© 2014 Health Level Seven International. All rights reserved

```
</when>
```

The following table lists the comparison operators available in the HeD Schema expression language:

Expression	Description
Equal	Returns true if the operands are equal.
NotEqual	Returns true if the operands are not equal.
Less	Returns true if the first operand is less than the second operand.
LessOrEqual	Returns true if the first operand is less than or equal to the second operand.
Greater	Returns true if the first operand is greater than the second operand.
GreaterOrEqual	Returns true if the first operand is greater than or equal to the second operand.

3.4.3 Logical Operators

The HeD Schema defines logical operators that can be used to combine the results of logical expressions. And and Or can be used to combine any number of results, and Not can be used to invert the result of any expression.

Note that these operators are defined with 3-valued logic semantics, allowing the operators to deal consistently with missing information.

For example, in the following expression:

```
<condition xsi:type="And">
        <operand xsi:type="LessOrEqual">
                <operand xsi:type="Property" path="demographics.age"/>
                <operand xsi:type="IntegerLiteral" value="20"/>
        </operand>
        <operand xsi:type="Equal">
                <operand xsi:type="Property" path="status"/>
                <operand xsi:type="Literal" valueType="xsi:string" value="Active"/>
        </operand>
</condition>
```

If the patient's age is not known, but the status is present and not equal to Active, the result of the And operation will be *false*. In contrast, if the patient's age is not known, but the status *is* equal to Active, the result of the And operation is *unknown*. Only in the case where the patient's Age is known to be less than or equal to 20, and the status is known to be Active will the And operation evaluate to *true*.

The following table lists the logical operators available in the HeD Schema expression language:

© 2014 Health Level Seven International. All rights reserved

Expression	Description
And	Returns the logical conjunction of its operands.
Or	Returns the logical disjunction of its operands.
Not	Returns the logical negation of its operand.

3.4.4 Nullological Operators

The HeD Schema expression language defines several nullological operators that are useful for dealing with potential missing information. These are *Null, IsNull*, *IfNull*, and *Coalesce*.

The following table lists the logical operators available in the HeD Schema expression language:

Expression	Description
Null	Returns a typed null.
IsNull	Returns true if the argument is *null*, false otherwise.
IfNull	Returns the first argument if it is not null, otherwise, returns the second argument.
Coalesce	Returns the first non-null argument, null if there are no non-null arguments.

3.4.5 Conditional Operators

The HeD Schema expression language defines several conditional expressions that can be used to return different values based on a condition, or set of conditions. These are the *Conditional* expression, and the *Case* expression.

The conditional expression allows a simple condition to be used to decide between one expression or another. For example:

```
<expression xsi:type="Conditional">
        <condition xsi:type="LessOrEqual">
                <operand xsi:type="Property" path="demographics.age"/>
                <operand xsi:type="IntegerLiteral" value="20"/>
        </condition>
        <then xsi:type="StringLiteral" value="Patient is 20 years old or less."/>
        <else xsi:type="StringLiteral" value="Patient is over 20 years old."/>
</expression>
```

The above expression will evaluate to the string *"Patient is 20 years old or less."* if the patient age property is less than or equal to 20. Otherwise, the expression will evaluate to *"Patient is over 20 years old."*

The case expression has two varieties, one that is equivalent to repeated conditionals, and one that allows a specific comparand to be identified and compared with each item to determine a result.

The following example illustrates the multi-condition variety:

```
<expression xsi:type="Case">
        <caseItem>
```


© 2014 Health Level Seven International. All rights reserved

```
        <when xsi:type="LessOrEqual">
                <operand xsi:type="Property" path="demographics.age"/>
                <operand xsi:type="IntegerLiteral" value="20"/>
        </when>
                <then xsi:type="StringLiteral" value="Patient is 20 years old or less."/>
        </caseItem>
        <caseItem>
                <when xsi:type="LessOrEqual">
                        <operand xsi:type="Property" path="demographics.age"/>
                        <operand xsi:type="IntegerLiteral" value="40"/>
                </when>
                <then xsi:type="StringLiteral" value="Patient is over 20, but not more than 40
years old."/>
        </caseItem>
        <else xsi:type="StringLiteral" value="Patient is over 40 years old."/>
</expression>
```

The following example illustrates a case expression using a comparand:

```
<expression xsi:type="Case">
        <comparand xsi:type="Property" path="demographics.age"/>
        <caseItem>
                <when xsi:type="IntegerLiteral" value="10"/>
                <then xsi:type="StringLiteral" value="Patient is 10 years old."/>
        </caseItem>
        <caseItem>
                <when xsi:type="IntegerLiteral" value="20"/>
                <then xsi:type="StringLiteral" value="Patient is 20."/>
        </caseItem>
        <else xsi:type="StringLiteral" value="Patient is neither 10 nor 20 years old."/>
</expression>
```

The following table lists the conditional operators available in the HeD Schema expression language:

Expression	Description
Conditional	Allows for conditional evaluation between two expressions.
Case	Allows for multiple conditional expressions, or a comparand with multiple cases.

© 2014 Health Level Seven International. All rights reserved

3.4.6 Arithmetic Operators

The HeD Schema expression language provides a complete set of arithmetic operators to allow for manipulation of integer and real values within artifacts. In general, these operators have the expected semantics for arithmetic operators.

Note that if an operand evaluates to null, the result of the operation is defined to be null. This provides consistent semantics when dealing with missing information.

The following examples illustrate the use of some common arithmetic operators:

```
<expression xsi:type="Add">
        <operand xsi:type="IntegerLiteral" value="2"/>
        <operand xsi:type="IntegerLiteral" value="2"/>
</expression>

<expression xsi:type="Multiply">
        <operand xsi:type="IntegerLiteral" value="6"/>
        <operand xsi:type="IntegerLiteral" value="9"/>
</expression>

<expression xsi:type="TruncatedDivide">
        <operand xsi:type="IntegerLiteral" value="63"/>
        <operand xsi:type="IntegerLiteral" value="2"/>
</expression>
```

The following table lists the arithmetic operators available in the HeD Schema expression language:

Expression	Description
Add	Performs numeric addition of its arguments.
Subtract	Performs numeric subtraction of its arguments.
Multiply	Performs numeric multiplication of its arguments.
Divide	Performs numeric division of its arguments.
TruncatedDivide	Performs integer division of its arguments.
Modulo	Computes the remainder of the division of its arguments.
Ceiling	Returns the first integer greater than or equal to its argument.
Floor	Returns the first integer less than or equal to its argument.
Truncateb589b589	Returns the integer component of its argument.
Abs	Returns the absolute value of its argument.
Negate	Returns the negative value of its argument.
Round	Returns the nearest numeric value to its argument, optionally specified to a number of decimal places for rounding.
Ln	Computes the natural logarithm of its argument.

© 2014 Health Level Seven International. All rights reserved

Expression	Description
Log	Computes the logarithm of its first argument, using the second argument as the base.
Power	Raises the first argument to the power given by the second argument.
Succ	Returns the successor of its argument.
Pred	Returns the predecessor of its argument.
MinValue	Returns the minimum representable value for a type.
MaxValue	Returns the maximum representable value for a type.

3.4.7 String Operators

The HeD Schema expression language defines a set of string operators to allow for manipulation of string values within artifact definitions.

Indexes within strings are defined to be 1-based.

Note that except as noted within the documentation for each operator, if any argument evaluates to null, the result of the operation is also defined to be null.

The following examples illustrate some common string manipulation operators:

```
<expression xsi:type="Pos">
        <pattern xsi:type="StringLiteral" value="abc"/>
        <string xsi:type="StringLiteral" value="abcdefg"/>
</expression>

<expression xsi:type="Substring">
        <stringToSub xsi:type="StringLiteral" value="abcdefg"/>
        <startIndex xsi:type="IntegerLiteral" value="1"/>
        <length xsi:type="IntegerLiteral" value="3"/>
</expression>

<expression xsi:type="Indexer">
        <operand xsi:type="StringLiteral" value="abcdefg"></operand>
        <index xsi:type="IntegerLiteral" value="1"/>
</expression>
```

The following table lists the string operators available in the HeD Schema expression language:

Expression	Description
Concat	Returns the concatenation of its arguments.
Combine	Combines a list of strings, optionally separating them with the given separator.
Split	Splits a string into a list of strings along a given separator.

© 2014 Health Level Seven International. All rights reserved

Expression	Description
Length	Returns the length of its argument.
Upper	Returns the upper case representation of its argument.
Lower	Returns the lower case representation of its argument.
Indexer	Returns the nth element of its argument.
Pos	Returns the starting position of a given pattern within a string.
Substring	Returns a substring of its argument.

3.4.8 Date and Time Operators

The HeD Schema expression language defines several operators for manipulating date and time values within HeD artifacts. These operators are defined using a common granularity type that allows the various granularities (e.g. day, month, week, hour, minute, second) of time to be manipulated.

Except as noted within the documentation for each operator, if any argument evaluates to null, the result of the operation is also defined to be null.

The following example constructs an interval of dates beginning 6 months before today, and ending today:

```
<dateRange xsi:type="Interval">
        <begin xsi:type="DateAdd">
                <date xsi:type="Today"/>
                <granularity xsi:type="Literal" valueType="DateGranularity"
value="Month"/>
                <numberOfPeriods xsi:type="Literal" valueType="xsi:int" value="-6"/>
        </begin>
        <end xsi:type="Today"/>
</dateRange>
```

The following table lists the date and time operators available in the HeD Schema expression language:

Expression	Description
DateAdd	Adds a given number of periods to its argument.
DateDiff	Computes the number of periods between a starting and ending date.
DatePart	Returns a specified component of its argument.
Today	Returns the date (with no time component) of the start timestamp associated with the evaluation request.
Now	Returns the date and time of the start timestamp associated with the evaluation request.
Date	Constructs a date from its arguments.
DateOf	Returns the date (with no time component) of the argument.
TimeOf	Returns the time (with the date set to the minimum representable date) of the argument.

© 2014 Health Level Seven International. All rights reserved

3.4.9 List Values

The HeD Schema expression language allows for the expression and manipulation of lists of values of any type. The most basic list operation is the list selector:

```
<source xsi:type="List">
        <element xsi:type="Value" valueType="xsi:int" value="1"/>
        <element xsi:type="Value" valueType="xsi:int" value="2"/>
        <element xsi:type="Value" valueType="xsi:int" value="3"/>
        <element xsi:type="Value" valueType="xsi:int" value="4"/>
        <element xsi:type="Value" valueType="xsi:int" value="5"/>
</source>
```

The above expression creates a list of integers with the elements *1, 2, 3, 4* and *5*.

Basic list operations include testing for membership, indexing, and content, as the following examples illustrate:

```
<!-- Returns true because the operand has no elements -->
<expression xsi:type="IsEmpty">
        <operand xsi:type="List"/>
</expression>

<!-- Returns true because the operand has at least one element -->
<expression xsi:type="IsNotEmpty">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
</expression>

<!-- Returns the second element, the integer 5 -->
<expression xsi:type="Indexer">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="4"/>
                <element xsi:type="IntegerLiteral" value="5"/>
                <element xsi:type="IntegerLiteral" value="6"/>
        </operand>
        <index xsi:type="IntegerLiteral" value="2"/>
```


© 2014 Health Level Seven International. All rights reserved

```
</expression>

<!-- Returns the index of the integer 5, 2 -->
<expression xsi:type="IndexOf">
        <source xsi:type="List">
                <element xsi:type="IntegerLiteral" value="4"/>
                <element xsi:type="IntegerLiteral" value="5"/>
                <element xsi:type="IntegerLiteral" value="6"/>
        </source>
        <element xsi:type="IntegerLiteral" value="5"/>
</expression>

<!-- Returns true because the list contains the integer 2 -->
<expression xsi:type="Contains">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
        <operand xsi:type="IntegerLiteral" value="2"/>
</expression>
```

The language also supports comparison of lists, including equality and inclusion determination (subset/superset):

```
<!-- Returns true because the lists are equal by value. -->
<expression xsi:type="Equal">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
```


© 2014 Health Level Seven International. All rights reserved

```
</expression>

<!-- Returns true because the first list includes the second. -->
<expression xsi:type="Includes">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
                <element xsi:type="IntegerLiteral" value="4"/>
                <element xsi:type="IntegerLiteral" value="5"/>
        </operand>
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
                <element xsi:type="IntegerLiteral" value="4"/>
        </operand>
</expression>
```

Supported operations on single lists include filtering, sorting, and computation:

```
<!-- Returns a list with only the first element -->
<expression xsi:type="Filter">
        <source xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </source>
        <condition xsi:type="Equal">
                <operand xsi:type="Current"/>
                <operand xsi:type="IntegerLiteral" value="1"/>
        </condition>
</expression>

<!-- Returns the list sorted by value ascending -->
<expression xsi:type="Sort">
        <source xsi:type="List">
                <element xsi:type="IntegerLiteral" value="3"/>
```


© 2014 Health Level Seven International. All rights reserved

```
            <element xsi:type="IntegerLiteral" value="1"/>
            <element xsi:type="IntegerLiteral" value="2"/>
            <element xsi:type="IntegerLiteral" value="4"/>
            <element xsi:type="IntegerLiteral" value="5"/>
        </source>
</expression>

<!-- Returns a list with each element set to the computed element (current * 2) -->
<expression xsi:type="ForEach">
        <source xsi:type="List">
            <element xsi:type="IntegerLiteral" value="1"/>
            <element xsi:type="IntegerLiteral" value="2"/>
            <element xsi:type="IntegerLiteral" value="3"/>
        </source>
        <element xsi:type="Multiply">
            <operand xsi:type="Current"/>
            <operand xsi:type="IntegerLiteral" value="2"/>
        </element>
</expression>
```

For multiple lists, the language supports combining through union and intersection, as well as computing the difference:

```
<!-- Returns a list with all elements (integers 1 through 9) -->
<expression xsi:type="Union">
        <operand xsi:type="List">
            <element xsi:type="IntegerLiteral" value="1"/>
            <element xsi:type="IntegerLiteral" value="2"/>
            <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
        <operand xsi:type="List">
            <element xsi:type="IntegerLiteral" value="4"/>
            <element xsi:type="IntegerLiteral" value="5"/>
            <element xsi:type="IntegerLiteral" value="6"/>
        </operand>
        <operand xsi:type="List">
```


© 2014 Health Level Seven International. All rights reserved

```
                <element xsi:type="IntegerLiteral" value="7"/>
                <element xsi:type="IntegerLiteral" value="8"/>
                <element xsi:type="IntegerLiteral" value="9"/>
        </operand>
</expression>

<!-- Returns a list with only the common elements (the integer 3) -->
<expression xsi:type="Intersect">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
                <element xsi:type="IntegerLiteral" value="4"/>
        </operand>
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="3"/>
                <element xsi:type="IntegerLiteral" value="4"/>
                <element xsi:type="IntegerLiteral" value="5"/>
        </operand>
</expression>

<!-- Returns a list without the elements of the second (the integer 1) -->
<expression xsi:type="Difference">
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
        </operand>
        <operand xsi:type="List">
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
```


© 2014 Health Level Seven International. All rights reserved

```
                <element xsi:type="IntegerLiteral" value="4"/>
        </operand>
</expression>
```

The use of the scope attribute allows for more complex expressions such as correlated subqueries:

```
<def name="Medications">
        <!-- Returns medications in the year 2013 with a code in the value set 12345 -->
        <expression xsi:type="ClinicalRequest"
                        cardinality="Multiple"
                        dataType="vmr:SubstanceAdministrationEvent"
                        codeProperty="substanceAdministrationGeneralPurpose"
                        dateProperty="administrationTimeInterval.low">
                <codes xsi:type="ValueSet" authority="VSAC" id="12345"/>
                <dateRange xsi:type="TimestampIntervalLiteral" highClosed="false">
                        <low value="20130101"/>
                        <high value="20140101"/>
                </dateRange>
        </expression>
</def>
<def name="Encounters">
        <!-- Returns encounters in the year 2013 with a code in the value set 56789 -->
        <expression xsi:type="ClinicalRequest"
                        cardinality="Multiple"
                        dataType="vmr:EncounterEvent"
                        codeProperty="encounterType"
                        dateProperty="encounterEventTime.low">
                <codes xsi:type="ValueSet" authority="VSAC" id="56789"/>
                <dateRange xsi:type="TimestampIntervalLiteral" highClosed="false">
                        <low value="20130101"/>
                        <high value="20140101"/>
                </dateRange>
        </expression>
</def>
<def name="CorrelatedSubquery">
        <!-- Returns medications that were administered during an encounter -->
```


© 2014 Health Level Seven International. All rights reserved

```
        <expression xsi:type="Filter" scope="M">
                <source xsi:type="ExpressionRef" name="Medications"/>
                <condition xsi:type="IsNotEmpty">
                        <operand xsi:type="Filter" scope="E">
                                <source xsi:type="ExpressionRef" name="Encounters"/>
                                <condition xsi:type="IncludedIn">
                                        <operand xsi:type="Property" scope="M"
                                                path="administrationTimeInterval"/>
                                        <operand xsi:type="Property" scope="E"
                                                path="encounterEventTime"/>
                                </condition>
                        </operand>
                </condition>
        </expression>
</def>
```

And the language supports a flattening operator, *Expand* to construct a single list from a list of lists:

```
<!-- Returns a single list containing the nested elements -->
<expression xsi:type="Expand">
        <operand xsi:type="List">
                <element xsi:type="List">
                        <element xsi:type="IntegerLiteral" value="1"/>
                        <element xsi:type="IntegerLiteral" value="2"/>
                        <element xsi:type="IntegerLiteral" value="3"/>
                </element>
                <element xsi:type="List">
                        <element xsi:type="IntegerLiteral" value="4"/>
                        <element xsi:type="IntegerLiteral" value="5"/>
                        <element xsi:type="IntegerLiteral" value="6"/>
                </element>
                <element xsi:type="List">
                        <element xsi:type="IntegerLiteral" value="7"/>
                        <element xsi:type="IntegerLiteral" value="8"/>
                        <element xsi:type="IntegerLiteral" value="9"/>
                </element>
```


© 2014 Health Level Seven International. All rights reserved

```
        </operand>
</expression>
```

The following table provides a complete listing of the list operators available in the HeD Schema expression language:

Expression	Description
List	Constructs a list from its arguments.
IsEmpty	Returns true if its argument contains no elements.
IsNotEmpty	Returns true if its argument contains any elements.
Equal	Returns true if its arguments have the same number of elements, and for each element considered in order, the elements are equal.
NotEqual	Returns true if its arguments are not equal.
Union	Returns a list containing all the elements of its arguments.
Difference	Returns a list containing only the elements in the first list that are not in the second list.
Intersect	Returns a list containing only the elements that are in all of its arguments.
Filter	Returns a list containing only the elements for which the given condition evaluates to true.
IndexOf	Returns the 1-based index of an element within the list, or 0 if the element is not present.
Indexer	Returns the element at the given 1-based index in the list.
In	Returns true if the given element is in a given list.
Contains	Returns true if the given list contains a given element.
Includes	
IncludedIn	
ProperIncludes	Returns true if every element in the first argument is in the second argument, and the second argument is strictly larger than the first.
ProperIncludedInb963	Returns true if the first argument contains every element in the second argument, and the first argument is strictly larger than the second.
Sort	Returns a list with the same elements, sorted by the given sort criteria.
ForEach	Returns a list whose elements are determined by evaluated an expression for each element in its argument.
Expand	
Distinct	Returns a list that contains the unique elements within its argument.
Current	Returns the contents of the current scope.

3.4.10 Aggregate Operators

For computing aggregate quantities, the HeD Schema expression language defines several aggregate operators. These operators perform computations on lists of values, either on the elements of the list directly, or on a specific property of each element in the list.

For example, the following invocation computes the sum of a list of integers:

© 2014 Health Level Seven International. All rights reserved

```
<expression xsi:type="Sum">
        <source xsi:type="List">
                <element xsi:type="IntegerLiteral" value="1"/>
                <element xsi:type="IntegerLiteral" value="2"/>
                <element xsi:type="IntegerLiteral" value="3"/>
                <element xsi:type="IntegerLiteral" value="4"/>
                <element xsi:type="IntegerLiteral" value="5"/>
        </source>
</expression>
```

Whereas the following example computes the sum of the *dose* property of the elements in the list returned by evaluating the *Medications* expression:

```
<expression xsi:type="Sum" path="dose">
        <source xsi:type="ExpressionRef" name="Medications"/>
</expression>
```

As another example, the following expression computes the average of 10 times the dose of the medications in the given list:

```
<expression xsi:type="Avg" path="dose">
        <source xsi:type="ForEach">
                <source xsi:type="ExpressionRef" name="Medications"/>
                <element xsi:type="Multiply">
                        <operand xsi:type="Property" path="dose"/>
                        <operand xsi:type="RealLiteral" value="10.0"/>
                </element>
        </source>
</expression>
```

Unless noted in the documentation for each operator, aggregate operators deal with missing information by excluding elements which have no value before performing the aggregation. In addition, an aggregate operation performed over an empty list is defined to return null, except as noted in the documentation for each operator (e.g. Count).

The following table lists the aggregate operators available in the HeD Schema expression language:

Expression	Description
Count	Returns the number of non-null elements in the source.
Sum	Computes the sum of non-null elements in the source.

© 2014 Health Level Seven International. All rights reserved

Expression	Description
Min	Returns the minimum element in the source.
Max	Returns the max element in the source.
Avg	Returns the average of the elements in the source.
Median	Returns the median of the elements in the source.
Mode	Returns the mode of the elements in the source.
Variance	Returns the statistical variance of the elements in the source.
PopulationVariance	Returns the population variance of the elements in the source.
StdDev	Returns the standard deviation of the elements in the source.
PopulationStdDev	Returns the population standard deviation of the elements in the source.
AllTrue	Returns true if all the elements in source are true.
AnyTrue	Returns true if any element in source is true.

3.4.11 Interval Values

The HeD Schema expression language defines a complete set of operators for use in defining and manipulating interval values.

Constructing an interval is performed with the *Interval* expression:

```
<expression xsi:type="Interval" endOpen="true">
        <begin xsi:type="Literal" valueType="xsi:Date" value="2010-10-10"/>
        <end xsi:type="Literal" valueType="xsi:Date" value="2010-10-11"/>
</expression>
```

This expression returns an interval from October 10th, 2010, inclusive, to October 11th, 2010, exclusive.

Basic operations on intervals includes determining length, accessing interval properties, and determining interval boundaries:

```
<!-- Returns 5, the interval contains 5 points. -->
<expression xsi:type="Length">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
</expression>
<!-- Returns 1, the beginning of the closed interval -->
<expression xsi:type="Begin">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
</expression>
<!-- Returns 2, the beginning of the open interval -->
<expression xsi:type="Begin">
        <operand xsi:type="IntegerIntervalLiteral" low="1" lowClosed="false" high="5"/>
</expression>
```


© 2014 Health Level Seven International. All rights reserved

The language supports complete operations involving comparisons of intervals, including equality, membership testing, and inclusion testing:

```
<!-- Returns true, because the intervals represent the same final range of points -->
<expression xsi:type="Equal">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"
                lowClosed="true" highClosed="true"/>
        <operand xsi:type="IntegerIntervalLiteral" low="0" high="6"
                lowClosed="false" highClosed="false"/>
</expression>
<!-- Returns true, the interval contains the point 2 -->
<expression xsi:type="Contains">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
        <operand xsi:type="IntegerLiteral" value="2"/>
</expression>
<!-- Returns true, the point 2 is in the interval -->
<expression xsi:type="In">
        <operand xsi:type="IntegerLiteral" value="2"/>
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
</expression>
<!-- Returns true, the first interval includes the second -->
<expression xsi:type="Includes">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
        <operand xsi:type="IntegerIntervalLiteral" low="2" high="4"/>
</expression>
<!-- Returns true, the first interval is properly included in the second -->
<expression xsi:type="ProperIncludedIn">
        <operand xsi:type="IntegerIntervalLiteral" low="2" high="4"/>
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
</expression>
```

In addition, the language supports operators for combining and manipulating intervals:

```
<!-- Returns the interval [1..7] -->
<expression xsi:type="Union">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
        <operand xsi:type="IntegerIntervalLiteral" low="3" high="7"/>
</expression>
```


© 2014 Health Level Seven International. All rights reserved

```
<!-- Returns the interval [3..5] -->
<expression xsi:type="Intersect">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
        <operand xsi:type="IntegerIntervalLiteral" low="3" high="7"/>
</expression>
<!-- Returns the interval [1..2] -->
<expression xsi:type="Difference">
        <operand xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
        <operand xsi:type="IntegerIntervalLiteral" low="3" high="7"/>
</expression>
<!-- Returns the intervals [1..9], [11..15] -->
<expression xsi:type="Collapse">
        <operand xsi:type="List">
                <element xsi:type="IntegerIntervalLiteral" low="1" high="5"/>
                <element xsi:type="IntegerIntervalLiteral" low="3" high="7"/>
                <element xsi:type="IntegerIntervalLiteral" low="5" high="9"/>
                <element xsi:type="IntegerIntervalLiteral" low="11" high="15"/>
                <element xsi:type="IntegerIntervalLiteral" low="13" high="14"/>
        </operand>
</expression>
```

The following table provides a complete listing of the interval operators available in the HeD Schema expression language:

Expression	Description
Equal	Returns true if the arguments are the same interval.
NotEqual	Returns true if the arguments are not the same interval.
Contains	Returns true if the interval contains the given point or interval.
In	
Includes	
IncludedIn	
ProperIncludes	
ProperIncludedIn	
Before	Returns true if the first interval ends before the second one starts.
After	Returns true if the first interval starts after the second one ends.
Meets	Returns true if the first interval ends on the starting point of the second, or if the first interval starts on the ending point of the second.
Overlaps	Returns true if the first interval overlaps the second.

© 2014 Health Level Seven International. All rights reserved

Expression	Description
OverlapsBefore	Returns true if the first interval starts before and overlaps the second.
OverlapsAfter	Returns true if the first interval ends after and overlaps the second.
Union	Returns the interval that results from combining the arguments.
Intersect	Returns the interval that results from the intersection of the arguments.
Difference	Returns the interval that results from subtracting the second interval from the first.
Length	Returns the length of the interval.
Begin	Returns the starting point of the interval.
End	Returns the ending point of the interval.
Begins	Returns true if the first interval begins the second.
Ends	Returns true if the first interval ends the second.

3.4.12 Structured Values

Structured values in the HeD Schema expression language are values with sets of named properties, each of which may have a value of any type. Structured values are most commonly used to represent clinical information such as encounters, problems, and procedures.

There are several operators that provide the ability to construct and manipulate structured values. The following example illustrates the creation of a simple structured value:

```
<actionSentence xsi:type="ObjectExpression"
        objectType="vmr:SubstanceAdministrationProposal">
        <description>Prescribe aspirin or other antithrombotic</description>
        <property name="substance.substanceCode">
                <value xsi:type="CodeLiteral"
code="2.16.840.1.113883.3.464.1003.196.12.1211"
codeSystem="National Committee for Quality Assurance"
displayName="Select a medication from this value set." />
        </property>
</actionSentence>
```

In addition to constructing a value, a new structured value may be created by modifying the properties of an existing value:

```
<value xsi:type="ObjectRedefine">
        <source xsi:type="ExpressionRef" name="PRNReasonRelatedObservation"/>
        <property name="observationValue.physicalQuantity"
xsi:type="PropertyExpression">
```


© 2014 Health Level Seven International. All rights reserved

```
                <value xsi:type="QuantityIntervalLiteral" lowIsInclusive="true"
highIsInclusive="true">
                        <low xsi:type="PQ" value="200" unit="Milligram/Deciliter"/>
                        <high xsi:type="PQ" value="249" unit="Miligram/Deciliter"/>
                </value>
        </property>
</value>
```

The example above creates a new observation result based on the value of the *PRNReasonRelatedObservation* but with a new value for the *physicalQuantity* property. Note that this is also an expression evaluation, so there is no change in the value returned by *PRNReasonRelatedObservation*; subsequent evaluations of that expression will return the same original value.

To access properties of a structured value, use the *Property* expression. A property expression has a *path* attribute, and an optional *source* element, and a *value* element. The source element returns the structured value to be accessed. In some contexts, such as within a *Filter* expression, the source is implicit. If used outside such a context, a source must be provided.

The path attribute specifies a property path relative to that structured value. The property expression returns the value of the property specified by the property path. Property paths are allowed to include qualifiers to indicate that subproperties should be traversed. For example:

```
<expression xsi:type="Property" path="demographics.age.value">
        <source xsi:type="ExpressionRef" name="Patient" />
</expression>
```

The above property expression accesses the *value* property of the *age* property of the *demographics* property of the structured value returned by the *Patient* expression.

The following table lists the structured value operators available in the HeD Schema expression language:

Expression	Description
ObjectExpression	Constructs a new structured value.
ObjectRedefine	Constructs a new structured value based on the values of an existing structured value, with specific properties given new values in the result.
Property	Returns the value of a specific property of a structured value.

3.4.13 Reusing Expressions

The HeD Schema expression language provides a mechanism for reusing expressions by declaring a named expression. This construct is similar to a function call with no parameters in a traditional imperative language.

The *ExpressionDef* type is used to define a named expression that can then be referenced within any HeD expression within the artifact:

```
<def name="PatientAge">
```


© 2014 Health Level Seven International. All rights reserved

```
        <expression xsi:type="Property" path="demographics.age.value">
                <source xsi:type="ExpressionRef" name="Patient" />
        </expression>
</def>
```

This example establishes the named expression *PatientAge*, which results in the age value of the patient; itself the result of evaluating the named expression *Patient*.

Note that circular expression references are not allowed, but that named expressions can be defined in any order, so long as the actual references do not result in a cycle.

The following table lists the expression definition components available in the HeD Schema expression language:

Expression	Description
ExpressionDef	Defines a named expression that can be referenced by other expressions.
ExpressionRef	Returns the result of evaluating a named expression.

3.4.14 External Data

All access to external data within the HeD Schema expression language is performed through the use of *Request* expressions.

The base request expression defines the cardinality and data type of the request. These attributes determine the type of elements to be returned, as well as the whether the result will be a single value, or a list of values.

The type of the elements to be returned is specified with the *dataType* attribute of the *ClinicalRequest*, and must refer to the name of a type within a known data model specified in the *dataModels* element of the artifact metadata. For more information on specifying the data model, please refer to the Clinical Data Models section of the Metadata discussion in Section 3.1.1 Clinical Data Models .

Note that *RequestBase* is an abstract type. This is intended to allow different types of requests to be introduced through extension. However, for the purposes of this implementation guide, the *ClinicalRequest* is the only request type that is defined. Furthermore, the HeD Schema places restrictions on where request expressions are allowed to appear within an artifact to improve readability of artifact definitions. The e*xternalData* element allows a list of *ExpressionDef* elements, each of which must contain one and only one *ClinicalRequest*. In addition, clinical requests may only appear within this external data element of the artifact. Further manipulation of these results must be performed elsewhere in the artifact, either in the *expressions* element (which allows any number of additional expressions to be defined), or directly within the logic of the artifact as appropriate.

In addition to the basic attributes defined on the base request, the *ClinicalRequest* introduces the ability to specify optional criteria for the request. The available criteria are intentionally restricted to the set of codes involved, and the date range involved. If these criteria are omitted, the request is interpreted to mean all data of that type.

NOTE: There is an implicit patient context assumed within HeD artifacts. Among other things, this implies that the relationships between clinical data (such as the patient and their associated encounters) are supplied by the implementation environment. This is an intentional

© 2014 Health Level Seven International. All rights reserved

simplifying assumption to avoid having to define those relationships explicitly within the artifact.

The following example illustrates a simple singleton request with no criteria:

```
<def name="Patient">
        <expression xsi:type="ClinicalRequest" cardinality="Single"
                dataType="vmr:EvaluatedPerson" isInitial="true" />
</def>
```

This definition (which must appear in the *externalData* section of the artifact) establishes the named expression *Patient* to return the singleton value of type *EvaluatedPerson*. Throughout the artifact, the patient information can now be accessed by referencing this expression. For example, the following definition (which must appear in the *expressions* section of the artifact) establishes the name *PatientAge* to refer specifically to the age property of the patient:

```
<def name="PatientAge">
        <expression xsi:type="Property" path="demographics.age.value">
                <source xsi:type="ExpressionRef" name="Patient" />
        </expression>
</def>
```

The following example illustrates a simple multiple cardinality request:

```
<def name="antithromboticNotPrescribedForDocumentedReason">
        <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                dataType="vmr:ObservationResult" codeProperty="observationFocus.code"
                dateProperty="observationEventTime">
                <description>Patient reason or other reason for not prescribing an
antithrombotic</description>
                <codes xsi:type="List">
                        <element xsi:type="CodeLiteral" code="G8697"
                                codeSystem="2.16.840.1.113883.6.12"
codeSystemName="CPT-4"
                                displayName="" />
                </codes>
        </expression>
</def>
```


© 2014 Health Level Seven International. All rights reserved

The above example defines *antithromboticNotPrescribedForDocumentedReason* to refer to all clinical data elements for the patient that are of type *ObservationResult* and that have an *observationFocus* of CPT-4:G8697. Because there is no date range criteria, this observation can be present at any time in the patient's record.

As a final example, the following definition establishes *onAntiThrombotic* to refer to all clinical data elements for the patient that are of type *SubstanceAdministrationEvent*, had a substance code within the given value set, and that were administered within the past year.

```
<def name="onAntiThrombotic">
        <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                dataType="vmr:SubstanceAdministrationEvent"
                codeProperty="substanceAdministrationGeneralPurpose.code"
                dateProperty="administrationTimeInterval.low" useValueSets="true">
                <description>Patient prescribed antithrombotic within the past
year</description>
                <codes xsi:type="ValueSet" id="2.16.840.1.113883.3.464.1003.196.12.1211"
                        authority="National Committee for Quality Assurance" />
                <dateRange xsi:type="Interval">
                        <begin xsi:type="DateAdd">
                                <date xsi:type="Today" />
                                <granularity xsi:type="Literal" valueType="DateGranularity"
                                        value="Month" />
                                <numberOfPeriods xsi:type="Literal" valueType="xsi:int"
                                        value="-12" />
                        </begin>
                        <end xsi:type="Today" />
                </dateRange>
        </expression>
</def>
```

In addition to specifying external data, the HeD Schema expression language defines several operators for referencing and working with terminology sets.

The following table lists the expressions relevant to defining external data and value sets in the HeD Schema expression language:

Expression	Description
ClinicalRequest	Defines clinical data that will be used within the artifact.
ValueSet	Returns the list of codes for a value set.
InValueSet	Tests a code for membership in a value set.

© 2014 Health Level Seven International. All rights reserved

Expression	Description
Subsumes	Tests two codes for subsumption (i.e. whether the concept represented by one code is subsumed by the concept represented by another).
SetSubsumes	Computes the intersection of two sets of codes using the subsumption relationship.

3.4.15 Parameters

In addition to external data, the HeD Schema expression language provides a mechanism for defining parameters to an artifact. An artifact can define any number of parameters, each of which has a name, and a defined type, as well as an optional default value.

Parameter values, if any, are expected to be provided as part of the evaluation request, and can be accessed with a *ParameterRef* expression in any expression throughout the artifact.

The following example illustrates a parameter definition:

```
<parameterDef name="MonthsThreshold" xsi:type="xsi:int">
        <default xsi:type="IntegerLiteral" value="6"/>
</parameterDef>
```

And this example illustrates the use of this parameter within an external data definition:

```
<expressionDef name="DiabetesDiagnoses">
        <!-- Get Diabetes diagnoses within the last @MonthsThreshold months -->
        <expression xsi:type="ClinicalRequest" dataType="vmr:Problem"
cardinality="Multiple"
isInitial="true">
                <codes xsi:type="ValueSet" id="DiabetesDiagnosisCodes"/>
                <dateRange xsi:type="Interval">
                        <begin xsi:type="DateAdd">
                                <date xsi:type="Today"/>
                                <granularity xsi:type="Literal"
valueType="DateGranularity" value="Month"/>
                                <numberOfPeriods xsi:type="Negate">
                                        <operand xsi:type="ParameterRef"
name="MonthsThreshold"/>
                                </numberOfPeriods>
                        </begin>
                        <end xsi:type="Today"/>
                </dateRange>
        </expression>
</expressionDef>
```


© 2014 Health Level Seven International. All rights reserved

The following table lists the expressions relevant to parameters in the HeDS expression language:

Expression	Description
ParameterDef	Defines a parameter to the artifact.
ParameterRef	Returns the value of a parameter.

3.5 Extending the Schema

The Knowledge Artifact Schema is designed to be extensible by an artifact developer, if the current standard specification does not meet the needs of the developer. For example, the artifact developer may want to incorporate features that are specific to a particular EHR platform such as layout of actions in an order set. Such extensions will not interoperate with other EHR platforms.

The types of extensions that are allowed to the schema are intended to facilitate integration with specific implementation environments but still maintain interoperability of the artifact with other environments. In order to guarantee interoperability, the artifact semantics cannot reside in the extensions. The extension can only add information that aids the other endpoint in integrating the artifact (e.g., EHR-specific items) but not add to the meaning of the artifact itself. That is, the artifact, fully stripped of its extensions must still retain the same semantics in an interoperability use case since not all receiving endpoints may understand such extensions. Thus, a consumer of the artifact who does not know how to interpret the extensions must still be able to consume and use the artifact, ignoring the elements represented by the extensions. In order for this to occur, the extensions MUST NOT redefine the semantics of the standard schema. The extensions MUST NOT remove or violate any of the HeDS conformance requirements specified in this document.Further, it is the responsibility of the extension developer to communicate the structure and semantics of the extensions to the artifact consumer, if they would like the consumer to be able to use the extensions.

We recommend that the developer of extensions provide feedback to the CDS Working Group on the extensions they have created so that such extensions may be incorporated in future revisions of the schema. Additionally, artifact developers and consumers also are encouraged to communicate any shortcomings of the standard to the CDS Working Group so that they may be considered for inclusion in future revisions.

We describe below the mechanisms that may be used to extend the schema.

3.5.1 Extending Types

An artifact developer may create new complexTypes by extending the complexTypes in the standard schema, using the approaches specified in XML schema definition. For example, the CreateAction complexType can be extended to a new type called CreateActionForAcmeEHR. The latter type adds a field that can allow order items within an order set to be shown in specified colors, a feature supported in the Acme EHR.

3.5.2 Extending Enumerations

The standard schema provides a number of enumerations. These are in the enum sub-folder of the schema distribution. The enumerations can be extended by adding new items. However, existing items in an enumeration cannot be removed. New items are added to the enumeration by adding the values to the appropriate schema file in the ext sub-folder of the schema. For

© 2014 Health Level Seven International. All rights reserved

example, to add a new item, say “Revoked”, to the enumeration ArtifactLifeCycleEvent, edit the file artifactlifecycleeventext.xsd in the ext folder. Remove the restriction of the type ArtifactLifeCycleEventTypeExt on the ArtifactLifeCycleEventTypeCore. Now, to add the enumeration entry “Revoked”:

```
<xs:enumeration value="Restored"/>
```

3.5.3 Other Extensions and Modifications

The following types of extensions are out of scope of this guide:

1. Using the schema to represent other types of knowledge artifacts beyond order sets, ECA rules, and documentation templates.
2. Using data models other than the vMR.
3. Extending the vMR objects
4. Extending the datatypes
5. Extending the expression language to include new operators and functionality

© 2014 Health Level Seven International. All rights reserved

4 LANGUAGE IMPLEMENTATION

This Section contains more detailed information relating to the intended semantics of the HeD Schema expression language. These topics are specifically relevant for readers interested in building translation, semantic validation, or evaluation applications for the HeD Schema expression language.

4.1 Clinical Data Retrieval in HeDS Artifacts

This section discusses the problem of clinical data retrieval in the clinical decision support space in general, and how the problem is resolved in the HeDS specification.

4.1.1 Defining Clinical Data

The problem of determining what data needs to be involved in the evaluation of any given artifact if that artifact contains arbitrary queries against the data model, is equivalent to the problem of query containment from database theory. This problem is known to be undecidable for arbitrary queries of the relational algebra, but is also shown to be both decidable and equivalent to the problem of query evaluation for the restricted class of conjunctive queries (Foundations of Databases, Abiteboul, Hull, Vianu).

In the Clinical Decision Support space, this problem is further complicated by the problem of terminology mapping. The meaning of a particular clinical statement within a patient's data is represented with a vocabulary consisting of codes which determine the kind of statement being represented. For example, a diagnosis clinical statement may be classified using the ICD-9 vocabulary, identifying the specific diagnosis represented.

In order for Clinical Decision Support artifacts to operate correctly, the meaning of each clinical statement, as identified by the vocabularies involved, must be preserved. However, this meaning is often represented in different vocabularies in different systems. A mapping between the vocabularies is therefore required in order to facilitate expression and evaluation of the artifact.

In addition, patient data is represented in differing schemas across various patient data sources, and must therefore be mapped structurally into the patient data model used by an artifact.

These problems collectively constitute what is referred to as the "curly braces problem" in the Arden space. This problem arises because of the difficulty in defining the structural and semantic aspects of the data involved.

The solution to this problem proposed by the HeDS specification is to create a well-defined and relatively simple interface between the clinical data provided by patient data sources, and the usage of that data within the artifact.

First, all clinical data within a HeDS knowledge artifact is represented using the HL7 Virtual Medical Record (VMR). This allows content to be authored without regard to the specific data models used by various patient data sources.

Second, all references to clinical data within a HeDS knowledge artifact are represented using a specific type of expression that only allows a well-defined set of clinically relevant criteria to be used to reference the data. The purpose of this restriction is two-fold: First, it allows the data required for evaluation to be determined solely by inspection of the artifact. And second, it allows for easy and reliable implementation of the interface between the evaluation engine and the patient data source, because the criteria used to request information from the patient data source are simple and well-defined.

© 2014 Health Level Seven International. All rights reserved

Third, by using standard terminologies within this data interface, the HeDS specification can guarantee that any given clinical statement referenced in an artifact has the same meaning as the data that is provided to the artifact from the patient data source. At a high level, this is the terminology problem; ensuring that the vocabularies used within the artifact are accurately mapped to the vocabularies used by the patient data source.

These three motivating factors inform the design of the Request expression used within the HeDS specification.

4.1.2 Conformance Levels

Although HeDS uses the VMR, there are many possibilities for variance in the way that VMR data is provided. This problem leads to the potential for artifacts to reference properties within the model that may or not be provided within a given specific instance of patient data expressed in the VMR schema. To address this potential problem, the request expressions within a HeDS artifact specify not only the type of the data (meaning the specific model type being requested), but optionally a template identifier that further constrains the data that is expected in a given request. If a template identifier is provided, then the request expression is expected to return only data that matches the constraints in the given template.

To help communicate validity of an artifact for a specific use, the HeDS specification defines two conformance levels related to this use of templates:

4.1.2.1 Strict Conformance

A HeDS artifact can be said to be strictly conforming if all references to clinical statement model properties (elements and attributes of VMR model types) within the artifact are explicitly constrained by the templates used in the requests.

4.1.2.2 Loose Conformance

A HeDS artifact can be said to be loosely conforming if the artifact references properties that are not explicitly constrained by the templates used in the requests. This is not to say that the artifact is necessarily invalid, just that the structure of the clinical data provided to the request may or may not contain the elements referenced by properties within the artifact.

4.1.3 Artifact Data Requirements

Because of the way data access is modeled within HeDS, the data requirements of a particular artifact can be clearly and accurately defined by inspecting only the ClinicalRequest expressions defined within the artifact. The following table broadly describes the data defined by each Request:

Item	Description
Clinical Data Type	The type of clinical data to be retrieved.
Codes	The set of codes defining the clinical data. Only clinical data matching codes in the set will be retrieved. If no codes are specified, clinical data with any code will be retrieved.
Date Range	The date range for clinical data. Only data within the specified date range will be retrieved. If no date range is specified, clinical data of any date will be retrieved.
Timing	The IsInitial attribute specifies whether or not the request is part of the initial data requirements for the artifact.
Triggering	If specified, this indicates whether the request is defining the triggering criteria for the artifact.

© 2014 Health Level Seven International. All rights reserved

These criteria are designed to allow the implementation environment to communicate the data requirements for an artifact, or group of artifacts, to a consumer to allow the consumer to gather all and only the relevant clinical information for transport to the evaluation environment. This supports the near-real-time clinical decision support scenario where the evaluation environment is potentially separate from the medical records system environment.

To support further reducing the overall size of data required to be transported, the following steps can be taken to combine request descriptors that deal with the same type of clinical data.

First, create a request context for each unique type of request using the request data type and the triggering context for each initial request.

Next, for each request, add the codes to the matching request context (by data type and triggering context), recording the associated date range, if any, for each code. Note that the empty set of codes should be represented as the single code "ALL" for the purposes of this method. As date ranges are recorded, they must be merged so that for each code in each request context, no two date range intervals overlap or meet.

Once the date ranges for each code within each unique request context are determined, the unique set of date ranges for all codes is calculated, accumulating the set of associated codes. Each unique date range for the context then results in a final descriptor.

This process produces a set of clinical data descriptors with the following structure:

Property	Description
Clinical Data Type	The type of clinical data required.
Triggering Context	The triggering context, if any.
Codes	The set of applicable codes, possibly empty (meaning all codes).
Date Range	The applicable date range, possibly empty (meaning all dates).

Collectively, these descriptors then represent the minimum initial data requirements for the artifact, with any overlapping requests for the same type of data collapsed into a single request descriptor.

In addition to being used to describe the initial data requirements, this same process can be used to collapse additional data requests that are evaluated as part of further evaluation of the artifact.

4.2 Expression Language Conceptual Model

In order to completely specify the semantics of the expression logic defined within HeDS, the intended execution model for expressions must be clearly defined. The following sections discuss the conceptual components of the expression language, and how these components are defined to operate.

4.2.1 Data Model

The data model for the HeDS expression language provides the overall structure and definition for the types of operations and capabilities that can be represented within the language. Note that the schema itself is layered into a core expression schema, and a more specific, clinical expression schema. The expression schema deals with defining the core

© 2014 Health Level Seven International. All rights reserved

operations that are available without respect to any specific model. The clinical expression schema then extends those operations to include references to clinical data.

Note that although the expression language deals with various categories of types, these are only conceptually defined within the expression language schema. There is no expectation within the core expression language that any particular data model be used, only that whatever concrete data model is actually used can be concretely mapped to the type categories defined within HeDS. Because these type categories are extremely broad, this allows the HeDS expression language component to be used with a large class of concrete data models without modifying the underlying specification.

4.2.1.1 Values

A *value* within the HeDS expression language represents some piece of data. All values are of some *type*, which designates what operations can be performed on the value. There are four categories of types within the HeDS expression language:

1) Scalar types – Types representing simple values such as strings, integers, dates, and decimals.
2) Structured types – Types representing composite values consisting of sets of named properties, each of which has a declared type, and may or may not have a current value of that type.
3) Collection types – Types representing lists of values of some declared type.
4) Interval types – Types representing an interval of some declared type, called the *point* type.

4.2.1.2 Scalar Types

Scalar types allow for the representation of simple, atomic types, such as integers and strings. For example, the value **5** is a value of type *Integer*, meaning that it can be used in operations that require integer-valued input such as addition or comparison.

The expression language itself does not define any scalar types. The clinical expression layer introduces expressions for dealing with the subset of ISO 21090 data types defined as part of the HeDS specification. These are the same types used by the VMR, so literal expressions are defined at the clinical expression layer to facilitate expressing values of these data types.

4.2.1.3 Structured Types

Structured types allow for the representation of composite types. Typically, these types correspond to the model types defined in the clinical data model used for the artifact. Structured types are defined as containing a set of named properties, each of which are of some type, and may have a value of that type.

As with scalar types, the core expression layer does not define any structured types, it only provides facilities for constructing values of structured types and for operating on structured values.

4.2.1.4 Collection Types

Collection types allow for the representation of lists and sets of values of any type. All the values within a collection are expected to be of the same type.

Collections may be empty, and are defined to be 1-based for indexing purposes.

© 2014 Health Level Seven International. All rights reserved

4.2.1.5 Interval Types

Interval types allow for the representation of intervals of some type. For example, an interval of integers allows the expression of the interval 1 to 5. Intervals can be open or closed at the beginning and/or end of the interval, and the beginning or end of the interval can be unspecified.

The core expression layer does not define any interval types, it only provides facilities for constructing values of interval types, and for operating on intervals.

4.2.2 Language Elements

The expression language specified as part of HeDS is defined as an Abstract Syntax Tree. Whereas a traditional language would have syntax and require lexical analysis and parsing, the HeDS specification allows expressions to be represented directly as trees. This removes potential ambiguities such as operator order precedence, and makes analysis and processing of the expressions in the language much easier.

Concretely, this is accomplished by defining the language elements as types in an XML schema. Each language element is represented by a type in the XML schema. For example, the following element represents an integer literal expression:

```
<expression xsi:type="Literal" valueType="xs:int" value="6"/>
```

Arguments to operations are represented naturally using the hierarchical structure of the XML document. For example, the following fragment represents an expression for adding the integer values 2 and 2:

```
<expression xsi:type="Add">
        <operand xsi:type="IntegerLiteral" value="2"/>
        <operand xsi:type="IntegerLiteral" value="2"/>
</expression>
```

This structure allows expressions of arbitrary complexity to be built up using the language elements defined in the schema. Essentially, the language consists of only two kinds of elements: 1) Expressions, and 2) Expression Definitions.

Each expression returns a value of some type, and an expression definition allows a given expression to be defined with an identifier so that it can be referenced in other expressions.

These expressions and expression definitions are then used throughout the HeDS specification wherever logic needs to be applied within an artifact.

4.2.3 Semantic Validation

Semantic Validation of an expression within the HeDS expression language is the process of verifying that the meaning of the expression is valid. This involves determining the type of each expression, and verifying that the arguments to each operation have the correct type.

This process proceeds as follows:

The graph of the expression being validated is traversed and the type of each node is determined. If the node has children (operands) the type of each child is determined in order to determine the type of the node. The following table defines the categories of nodes and the process for determining the type of each category:

© 2014 Health Level Seven International. All rights reserved

Node Category	Type Determination
Literal	The type of the node is the type of the literal being represented.
Property	The type of the node is the declared type of the property being referenced.
ParameterRef	The type of the node is the parameterType of the parameter being referenced.
ExpressionRef	The type of the node is the type of the expression being referenced.
ClinicalRequest	The type of the node is a list of the type of the data being requested.
ValueSet	The type of the node is a list of codes.
Operator	Generally, the type of the node is determined by resolving the type of each operand, and then using that signature to determine the resulting type of the operator.

During validation, the implementation must maintain a stack of symbols that track the type of the object currently in scope. This allows the type of context-sensitive operators such as Current and Property to be determined. Refer to the Execution Model section for a description of the evaluation-time stack.

Details for the specifics of type determination for each operator are provided with the documentation for those operators.

4.2.4 Execution Model

All logic in the HeDS expression language is represented as *expressions*. The language is pure functional, meaning no operations are allowed to have side effects of any kind. An expression may consist of any number of other expressions and operations, so long as they are all combined according to the semantic rules for each operation as described in the Semantic Validation section.

Because the language is pure functional, every expression and operator is defined to return the same value on every evaluation within the same artifact evaluation. In particular this means:

1) All clinical data returned by request expressions within the artifact must return the same set on every evaluation. An implementation would likely use a snapshot of the required clinical data in order to achieve this behavior.
2) Invocations of non-deterministic operations such as Now() and Today() are defined to return the timestamp associated with the evaluation request, rather than the clock of the engine performing the evaluation.

Once an expression has been semantically validated, its return type is known. This means that the expression is guaranteed to return either a value of that type, or a *null*, indicating the evaluation did not result in a value.

In general, operations are defined to result in null if any of their arguments are null. For example, the result of evaluating 2 + null is null. In this way, missing information results in an unknown result. There are exceptions to this rule, notably the logical operators, and the null-handling operators. The behavior for these operators (and others that do not follow this rule) are described in detail in the documentation for each operator.

© 2014 Health Level Seven International. All rights reserved

Evaluation takes place within an execution model that provides access to the data and parameters provided to the evaluation. Data is provided to the evaluation as a set of lists of structured values representing a patient's clinical information. In order to be represented in this data set, a given structured value must be a *cacheable* item. A cacheable item must have the following:

Property	Description
Identifier	A property, or set of properties, that uniquely identify the item.
Codes	A code, or list of codes that identify the associated clinical codes for the item.
Date	A date time defining the clinically relevant date and/or time of the item.

Evaluation consists of two phases, a *pre-processing* phase, and an *evaluation* phase. The pre-processing phase is used to determine the initial data requirements for a rule. During this phase any request expressions in the rule are analyzed to determine what data must be provided to the evaluation in order to successfully complete a rule evaluation. All requests with IsInitial set to true, as well as any trigger requests, are considered and a set of data descriptors is produced using the method described in the Artifact Data Requirements section. This means in particular that the Codes and DateRange expressions in each of the initial and triggering requests must be compile-time evaluable. This means that these expressions may not reference any clinical information, though they are allowed to reference parameter values.

During the evaluation phase, the result of the expression is determined. Conceptually, evaluation proceeds as follows:

The graph of the expression being evaluated is traversed and the result of each node is calculated. If the node has children (operands), the result of each child is evaluated before the result of the node can be determined. The following table describes the general categories of nodes and the process of evaluation for each:

Node Category	Evaluation
Literal	The result of the node is the value of the literal represented.
Operation	The result of the node is the result of the operation described by the node given the results of the operand nodes of the expression.
ClinicalRequest	The result of the node is the result of retrieving the data represented by the request, i.e. a list of structured values of the type defined in the request representing the patient information being retrieved.
ExpressionRef	The result of the node is the result of evaluating the referenced expression.
ParameterRef	The result of the node is the value of the referenced parameter.

During evaluation, the implementation must maintain a stack that is used to represent the value that is currently in context. Certain operations within the expression language are defined with a scope, and these operations use the stack to represent this scope. The following table details these operations:

Operation	Stack Effect

© 2014 Health Level Seven International. All rights reserved

Operation	Stack Effect
ObjectRedefine	The *source* operand is pushed on to the stack prior to evaluating the *property* expressions. The stack is popped before the result is returned.
Filter	For each item in the *source* operand, the item is pushed on to the stack, the *condition* expression is evaluated, and the item is popped off of the stack.
ForEach	For each item in the *source* operand, the item is pushed on to the stack, the *element* expression is evaluated, and the item is popped off of the stack.

The *scope* attribute of these operators provides an optional name for the item being pushed on to the stack. This name can be used within the **Current** and **Property** expressions to determine which element on the stack is being accessed. If no scope is provided, the top of the stack is assumed.

Details for the evaluation behavior of each specific operator are provided as part of the documentation for each operator.

© 2014 Health Level Seven International. All rights reserved

5 SCHEMA REFERENCES

Namespace Summary

urn:hl7-org:cdsdt:r2	Page
Targeting Schemas (1): datatypes.xsd **Targeting Components:** elements (29 local), complexTypes (30), simpleTypes (20)	78

urn:hl7-org:knowledgeartifact:r1	Page
Targeting Schemas (48): action.xsd, actor.xsd, artifactlifecycleeventtype.xsd, artifactlifecycleeventtypecore.xsd, artifactlifecycleeventtypeext.xsd, artifactstatustype.xsd, artifactstatustypecore.xsd, artifactstatustypeext.xsd, artifacttype.xsd, artifacttypecore.xsd, artifacttypeext.xsd, base.xsd, behavior.xsd, behaviortype.xsd, behaviortypecore.xsd, behaviortypeext.xsd, cardinalitytype.xsd, catalogitem.xsd, clinicalexpression.xsd, condition.xsd, conditionroletype.xsd, conditionroletypecore.xsd, conditionroletypeext.xsd, contributortype.xsd, contributortypecore.xsd, contributortypeext.xsd, coveragetype.xsd, coveragetypecore.xsd, coveragetypeext.xsd, dataeventtype.xsd, dataeventtypecore.xsd, dataeventtypeext.xsd, eventtype.xsd, eventtypecore.xsd, eventtypeext.xsd, expression.xsd, knowledgedocument.xsd, literalexpression.xsd, metadata.xsd, rangeconstrainttype.xsd, rangeconstrainttypecore.xsd, rangeconstrainttypeext.xsd, resourcerelationshiptype.xsd, resourcerelationshiptypecore.xsd, resourcerelationshiptypeext.xsd, valuetype.xsd, valuetypecore.xsd, valuetypeext.xsd **Targeting Components:** elements (2 global + 212 local), complexTypes (189), simpleTypes (48)	149

Schema Summary		Page
action.xsd	This file defines the Action types used to describe guidance within a knowledge artifact.	407
actor.xsd	This file defines the Actor type.	407
artifactlifecycleeventtype.xsd	This file defines the ArtifactLifeCycleEventType enumeration.	407
artifactlifecycleeventtypecore.xsd	This file defines the core values available within the ArtifactLifeCycleEventType enumeration.	408
artifactlifecycleeventtypeext.xsd	This file allows organizations to extend the enumeration ArtifactLifeCycleEventType.	408
artifactstatustype.xsd	This file defines the ArtifactStatusType enumeration.	408
artifactstatustypecore.xsd	This file defines the core values available within the ArtifactStatusType enumeration.	409
artifactstatustypeext.xsd	This file allows organizations to extend the enumeration ArtifactStatusType.	409
artifacttype.xsd	This file defines the ArtifactType enumeration.	409
artifacttypecore.xsd	This file defines the core values available within the ArtifactType enumeration.	410
artifacttypeext.xsd	This file allows organizations to extend the enumeration ArtifactType.	410
base.xsd	This file defines the base types used in defining knowledge artifacts.	411
behavior.xsd	This file defines the behaviors used within knowledge documents.	411
behaviortype.xsd	This file defines the BehaviorType enumeration.	411
behaviortypecore.xsd	This file defines the core values available within the BehaviorType enumeration.	412
behaviortypeext.xsd	This file allows organizations to extend the enumeration BehaviorType.	412

cardinalitytype.xsd	This file defines the Cardinality enumeration.	*412*
catalogitem.xsd	This file defines catalog items used to describe questions and responses for the documentation template knowledge artifact type.	*413*
clinicalexpression.xsd	This file defines the RequestBase extensions that introduce clinically relevant dependencies such as terminology, relevant time, and value set considerations.	*413*
condition.xsd	This file defines the condition components used in knowledge artifacts.	*413*
conditionroletype.xsd	This file defines the ConditionRoleType enumeration.	*414*
conditionroletypecore.xsd	This file defines the core values available within the ConditionRoleType enumeration.	*414*
conditionroletypeext.xsd	This file allows organizations to extend the enumeration ConditionRoleType.	*414*
contributortype.xsd	This file defines the ContributorType enumeration.	*415*
contributortypecore.xsd	This file defines the core values available within the ContributorType enumeration.	*415*
contributortypeext.xsd	This file allows organizations to extend the enumeration ContributorType.	*415*
coveragetype.xsd	This file defines the CoverageType enumeration.	*416*
coveragetypecore.xsd	This file defines the core values available within the CoverageType enumeration.	*416*
coveragetypeext.xsd	This file allows organizations to extend the enumeration CoverageType.	*416*
dataeventtype.xsd	This file defines the DataEventType enumeration.	*417*
dataeventtypecore.xsd	This file defines the core values available within the DataEventType enumeration.	*417*
dataeventtypeext.xsd	This file allows organizations to extend the enumeration DataEventType.	*418*
datatypes.xsd	Specifies data types used.	*418*
eventtype.xsd	This file defines the EventType enumeration.	*418*
eventtypecore.xsd	This file defines the core values available within the EventType enumeration.	*419*
eventtypeext.xsd	This file allows organizations to extend the enumeration EventType.	*419*
expression.xsd	This file defines the core expression components used throughout knowledge artifacts.	*419*
knowledgedocument.xsd	This file defines the root knowledge document type and element.	*420*
literalexpression.xsd	This file defines additional expressions that provide syntactic short-hands for literals for each of the base data types.	*420*
metadata.xsd	This file defines the meta data components used within a knowledge artifact.	*420*
rangeconstrainttype.xsd	This file defines the RangeConstraintType enumeration.	*421*
rangeconstrainttypecore.xsd	This file defines the core values available within the RangeCosntraintType enumeration.	*421*
rangeconstrainttypeext.xsd	This file allows organizations to extend the enumeration RangeConstraintType.	*421*
resourcerelationshiptype.xsd	This file defines the ResourceRelationshipType enumeration.	*422*
resourcerelationshiptypecore.xsd	This file defines the core values available within the ResourceRelationshipType enumeration.	*422*
resourcerelationshiptypeext.xsd	This file allows organizations to extend the enumeration ResourceRelationshipType.	*422*
valuetype.xsd	This file defines the ValueType enumeration.	*423*
valuetypecore.xsd	This file defines the core values available within the ValueType enumeration.	*423*
valuetypeext.xsd	This file allows organizations to extend the enumeration ValueType.	*423*

Namespace "urn:hl7-org:cdsdt:r2"

Targeting Schemas (1):
datatypes.xsd

Targeting Components:
elements (29 local), complexTypes (30), simpleTypes (20)

Complex Type Summary		Page
dt:AD	Mailing and home or office addresses.	*80*
dt:ADXP	A part with a type-tag signifying its role in the address.	*82*
dt:ANY	Defines the basic properties of every data value.	*84*
dt:BL	BL stands for the values of two-valued logic.	*84*
dt:CD	A CD is a reference to a concept defined in an external code system, terminology, or ontology.	*85*
dt:CO	Represents data where coded values are associated with a specific order.	*89*
dt:CS	Coded data in its simplest form, where only the code is not predetermined.	*91*
dt:ED	Data that is primarily intended for human interpretation or for further machine processing outside the scope of this specification.	*92*
dt:EN	A name for a person, organization, place or thing.	*97*
dt:ENXP	A part with a type code signifying the role of the part in the whole entity name, and qualifier codes for more detail about the name part type.	*98*
dt:HXIT	Information about the history of this value: period of validity and a reference to an identified event that established this value as valid.	*99*
dt:II	An identifier that uniquely identifies a thing or object.	*100*
dt:INT	Integer numbers (-1,0,1,2, 100, 3398129, etc.) are precise numbers that are results of counting and enumerating.	*102*
dt:IVL	A set of consecutive values of an ordered base datatype.	*103*
dt:IVL_CO	A set of consecutive values of an ordered base datatype.	*103*
dt:IVL_INT	A set of consecutive values of an ordered base datatype.	*105*
dt:IVL_PQ	A set of consecutive values of an ordered base datatype.	*107*
dt:IVL_QTY	A set of consecutive values of an ordered base datatype.	*109*
dt:IVL_REAL	A set of consecutive values of an ordered base datatype.	*111*
dt:IVL_TS	A set of consecutive values of an ordered base datatype.	*113*
dt:PIVL_TS	An interval of time that recurs periodically.	*115*
dt:PQ	A dimensioned quantity expressing the result of measuring.	*118*
dt:QSET	Abstract; specializes ANY Parameter: T : QTY An unordered set of distinct values which are quantities.	*120*
dt:QTY	The quantity datatype is an abstract generalization for all datatypes whose domain values has an order relation (less-or-equal) and where difference is defined in all of the datatype's totally ordered value subsets.	*121*
dt:REAL	Fractional numbers.	*121*
dt:RTO	A quantity constructed as the quotient of a numerator quantity divided by a denominator quantity.	*122*
dt:ST	The character string datatype stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) or direct display.	*124*
dt:TEL	A locatable resource that is identified by a URI, such as a web page, a telephone number (voice, fax or some other resource mediated by telecommunication equipment), an e-mail address, or any other locatable resource that can be specified by a URL.	*124*
dt:TS	A quantity specifying a point on the axis of natural time.	*126*

dt:XP	A part of a name or address.	*127*

Simple Type Summary		*Page*
dt:AddressPartType	Specifies whether an address part names the street, city, country, postal code, post box, etc.	*128*
dt:CalendarCycle		*131*
dt:Code	A code representing the string data.	*132*
dt:Compression	The compression algorithm, specified in the HL7 CompressionAlgorithm code system.	*133*
dt:Decimal	A number that is not restricted to an integer, and may contain fractional values between two integers.	*134*
dt:EntityNamePartQualifier	The qualifier is a set of codes each of which specifies a certain subcategory of the name part in addition to the main name part type.	*134*
dt:EntityNamePartType	Indicates whether the name part is a given name, family name, prefix, suffix, etc.	*136*
dt:EntityNameUse	A set of codes advising a system or user which name in a set of names to select for a given purpose.	*137*
dt:IntegrityCheckAlgorithm	The algorithm used to compute the integrityCheck value.	*139*
dt:PostalAddressUse	A set of codes advising a system or user which address in a set of like addresses to select for a given purpose.	*140*
dt:set_EntityNamePartQualifier		*142*
dt:set_EntityNameUse		*142*
dt:set_PostalAddressUse		*143*
dt:set_TelecommunicationAddressUse		*143*
dt:set_TelecommunicationCapability		*144*
dt:TelecommunicationAddressUse	One or more codes advising a system or user which telecommunication address in a set of like addresses to select for a given telecommunication need.	*144*
dt:TelecommunicationCapability	One or more codes advising a system or user what telecommunication capabilities are known to be associated with the telecommunication address.	*146*
dt:TimeStamp	Represents a timestamp such as 20101127235417.123+0930	*146*
dt:Uid	A unique identifier string is a character string which identifies an object in a globally unique and timeless manner.	*147*
dt:Uri	Universal Resource Identifier	*148*

complexType "dt:AD"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   use = list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "PHYS" | "PST" | "TMP" | "ABC" |
         "IDE" | "SYL" | "SRCH" | "SNDX" | "PHON")
   >
   Content: dt:part+
</...>
```

Content Model Elements (1):

dt:part (in address) [82]

Annotation

Mailing and home or office addresses.

AD is primarily used to communicate data that will allow printing mail labels, or that will allow a person to physically visit that address. The postal address datatype is not supposed to be a container for additional information that might be useful for finding geographic locations (e.g., GPS coordinates) or for performing epidemiological studies. Such additional information should be captured by other, more appropriate data structures.

Addresses are essentially sequences of address parts, but add a "use" code and a valid time range for information about if and when the address can be used for a given purpose.

Type Definition Detail

Type Derivation Tree

dt:HXIT [99] (extension)

└ *dt:ANY* [84] (extension)

└ **dt:AD**

XML Source (w/o annotations (3))

```
<xs:complexType name="AD">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="part" type="ADXP"/>
      </xs:sequence>
      <xs:attribute name="use" type="set_PostalAddressUse" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

use

Type: dt:set_PostalAddressUse [143]
Use: optional

A set of codes advising a system or user which address in a set of like addresses to select for a given purpose.
An address without specific use code might be a default address useful for any purpose, but an address with a specific use code would be preferred for that respective purpose.
If populated, the values contained in this attribute SHALL be taken from the HL7 PostalAddressUse code system.

Attribute Value

```
list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "PHYS" | "PST" | "TMP" | "ABC" | "IDE" |
"SYL" | "SRCH" | "SNDX" | "PHON")
```

Content Element Detail (all declarations; defined within this component only; 1/1)

dt:part

Type: dt:ADXP [82], empty content

A sequence of address parts, such as street or post office Box, city, postal code, country, etc.

XML Representation Summary

```
<dt:part
    type  = ("AL" | "ADL" | "UNID" | "UNIT" | "DAL" | "DINST" | "DINSTA" | "DINSTQ" | "DMOD" | "DMODID" | "SAL" | "BNR"
            | "BNN" | "BNS" | "STR" | "STB" | "STTYP" | "DIR" | "INT" | "CAR" | "CEN" | "CNT" | "CPA" | "CTY" | "DEL" |
            "POB" | "PRE" | "STA" | "ZIP" | "DPID")
    value = xs:string
/>
```

complexType "dt:ADXP"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value  =  xs:string
    type   =  ("AL" | "ADL" | "UNID" | "UNIT" | "DAL" | "DINST" | "DINSTA" | "DINSTQ" | "DMOD" | "DMODID"
              | "SAL" | "BNR" | "BNN" | "BNS" | "STR" | "STB" | "STTYP" | "DIR" | "INT" | "CAR" | "CEN" |
              "CNT" | "CPA" | "CTY" | "DEL" | "POB" | "PRE" | "STA" | "ZIP" | "DPID")
/>
```

Annotation

A part with a type-tag signifying its role in the address. Typical parts that exist in about every address are street, house number, or post box, postal code, city, country but other roles may be defined regionally, nationally, or on an enterprise level (e.g. in military addresses).

Type Definition Detail

Type Derivation Tree

dt:XP [127] (extension)

└ **dt:ADXP**

XML Source (w/o annotations (2))

```
<xs:complexType name="ADXP">
  <xs:complexContent>
    <xs:extension base="XP">
      <xs:sequence/>
      <xs:attribute name="type" type="AddressPartType" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/2)

type

Type: dt:AddressPartType [128]
Use: required

Whether an address part names the street, city, country, postal code, post box, address line 1, etc.
The value of this attribute SHALL be taken from the HL7 AddressPartType code system.

Attribute Value

enumeration of xs:string

Enumeration:

- "AL" – Address Line: An address line is for either an additional locator, a delivery address or a street address. An address generally has only a delivery address line or a street address line, but not both.
- "ADL" – Additional Locator : This can be a unit designator, such as apartment number, suite number, or floor. There may be several unit designators in an address (e.g., "3rd floor, Appt. 342"). This can also be a designator pointing away from the location, rather than specifying a smaller location within some larger one (e.g., Dutch "t.o." means "opposite to" for house boats located across the street facing houses)
- "UNID" – Unit Identifier : The number or name of a specific unit contained within a building or complex, as assigned by that building or complex
- "UNIT" – Unit Designator: Indicates the type of specific unit contained within a building or complex. E.g. Apartment, Floor
- "DAL" – Delivery Address Line: A delivery address line is frequently used instead of breaking out delivery mode, delivery installation, etc. An address generally has only a delivery address line or a street address line, but not both.
- "DINST" – Delivery Installation Type: Indicates the type of delivery installation (the facility to which the mail will be delivered prior to final shipping via the delivery mode.) Example: post office, letter carrier depot, community mail center, station, etc.
- "DINSTA" – Delivery Installation Area: The location of the delivery installation, usually a town or city, and is only required if the area is different from the municipality. Area to which mail delivery service is provided from any postal facility or service such as an individual letter carrier, rural route, or postal route.
- "DINSTQ" – Delivery Installation Qualifier: A number, letter or name identifying a delivery installation. E.g., for Station A, the delivery installation qualifier would be 'A'.
- "DMOD" – Delivery Mode: Indicates the type of service offered, method of delivery. For example: post office box, rural route, general delivery, etc.
- "DMODID" – Delivery Mode Identifier: Represents the routing information such as a letter carrier route number. It is the identifying number of the designator (the box number or rural route number).
- "SAL" – Street Address Line: A street address line is frequently used instead of breaking out build number, street name, street type, etc. An address generally has only a delivery address line or a street address line, but not both.
- "BNR" – Building Number: The number of a building, house or lot alongside the street. Also known as "primary street number". This does not number the street but rather the building.
- "BNN" – Building Number Numeric: The numeric portion of a building number
- "BNS" – Building Number Suffix: Any alphabetic character, fraction or other text that may appear after the numeric portion of a building number
- "STR" – Street Name: The name of the street, including the type
- "STB" – Street Name Base: The base name of a roadway or artery recognized by a municipality (excluding street type and direction)
- "STTYP" – Street Type: The designation given to the street. (e.g. Street, Avenue, Crescent, etc.)
- "DIR" – Direction (e.g., N, S, W, E)
- "INT" – Intersection: An intersection denotes that the actual address is located at or close to the intersection of two or more streets
- "CAR" – Care Of: The name of the party who will take receipt at the specified address, and will take on responsibility for ensuring delivery to the target recipient
- "CEN" – Census Tract: A geographic sub-unit delineated for demographic purposes.
- "CNT" – Country
- "CPA" – County or Parish: A sub-unit of a state or province. (49 of the United States of America use the term "county;" Louisiana uses the term "parish".)
- "CTY" – Municipality: The name of the city, town, village, or other community or delivery center
- "DEL" – Delimiter: Delimiters are printed without framing white space. If no value component is provided, the delimiter appears as a line break.
- "POB" – Post Box: A numbered box located in a post station.
- "PRE" – Precinct: A subsection of a municipality
- "STA" – State or Province: A sub-unit of a country with limited sovereignty in a federally organized country.

"ZIP" – Postal Code: A postal code designating a region defined by the postal service.

"DPID" – Delivery Point Identifier : A value that uniquely identifies the postal address.

complexType "dt:ANY"

Namespace: urn:hl7-org:cdsdt:r2
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*

XML Representation Summary

<...>

Annotation

Defines the basic properties of every data value. This is conceptually an abstract type, meaning that no proper value can be just a data value without belonging to any concrete type. Every public concrete type is a specialization of this general abstract DataValue type.

However exceptional values (nullFlavored values) may be of type ANY, except for the exceptional values that imply the nullFlavor INV, since this requires a type to be meaningful. Note that not all nullFlavors may be used with the type ANY.

This class is maintained here despite the lack of attributes to maintain compatibility with the ISO 21090 data structure.

We have also made it abstract to be consistent with the lack of support for nullFlavors.

Type Definition Detail

Type Derivation Tree

dt:HXIT [99] (extension)

└ ***dt:ANY***

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="ANY">
  <xs:complexContent>
    <xs:extension base="HXIT">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

complexType "dt:BL"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = xs:boolean
/>
```

Annotation

BL stands for the values of two-valued logic. A BL value can be either true or false.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:BL
```

XML Source (w/o annotations (2))

```
<xs:complexType name="BL">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
      <xs:attribute name="value" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

  Type: xs:boolean, predefined
  Use: required

  The value of the BL.

complexType "dt:CD"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 6 attributes, 3 elements

XML Representation Summary

```
<...
   code              = xs:string
   codeSystem        = xs:string
   codeSystemName    = xs:string
   codeSystemVersion = xs:string
   valueSet          = xs:string
   valueSetVersion   = xs:string
   >
   Content: dt:displayName?, dt:originalText?, dt:translation*
</...>
```

Content Model Elements (3):

dt:displayName [88], dt:originalText [88], dt:translation [89]

Annotation

A CD is a reference to a concept defined in an external code system, terminology, or ontology.

A CD may also contain an original text or phrase that served as the basis of the coding.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:CD
```

XML Source (w/o annotations (10))

```
<xs:complexType name="CD">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence>
        <xs:element minOccurs="0" name="displayName" type="ST"/>
        <xs:element minOccurs="0" name="originalText" type="ST"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="translation" type="CD"/>
      </xs:sequence>
      <xs:attribute name="code" type="Code" use="optional"/>
      <xs:attribute name="codeSystem" type="Uid" use="optional"/>
      <xs:attribute name="codeSystemName" type="xs:string" use="optional"/>
      <xs:attribute name="codeSystemVersion" type="xs:string" use="optional"/>
      <xs:attribute name="valueSet" type="Uid" use="optional"/>
      <xs:attribute name="valueSetVersion" type="xs:string" use="optional"/>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 6/6)

code

Type: dt:Code [132]
Use: optional

The plain code symbol defined by the code system, or an expression in a syntax defined by the code system which describes the concept.
Code SHALL be an exact match to a plain code symbol or expression defined by the code system. If the code system defines a code or expression that includes whitespace, the code SHALL include the whitespace. An expression can only be used where the codeSystem either defines an expression syntax, or there is a generally accepted syntax for the codeSystem. A code system may be defined that only defines an expression syntax with bindings to other code Systems for the elements of the expression.
It is at the discretion of the interpreting system whether to check for an expression instead of a simple code and evaluate the expression instead of treating the expression as a code. In some cases, it may be unclear or ambiguous whether the code represents a single symbol or an expression. This usually arises where the code system defines an expression language and then defines pre-coordinated concepts with symbols which match their expression, e.g. UCUM. In other cases, it is safe to treat the expression as a symbol. There is no guarantee that this is always safe: the definitions of the codeSystem should always be consulted to determine how to handle potential expressions.

Attribute Value

xs:string

codeSystem

Type: dt:Uid [147]
Use: optional

The code system that defines the code, or if no code was found, the codeSystem in which no code was found.
Code systems SHALL be referred to by a UID, which allows unambiguous reference to standard code systems and other local codesystems. Where either ISO or HL7 have assigned UID to code Systems, then these UIDs SHALL be used. Otherwise implementations SHALL use an appropriate ISO Object Identifier (OID) or UUID to construct a globally unique local coding system identifier.

Attribute Value

xs:string

codeSystemName

Type: xs:string, predefined
Use: optional

The common name of the coding system.

The code system name has no computational value. codeSystemName can never modify the meaning of codeSystem and cannot exist without codeSystem.

Information Processing Entities claiming direct or indirect conformance SHALL NOT functionally rely on codeSystemName. In addition, they MAY choose not to implement codeSystemName; but SHALL NOT reject instances because codeSystemName is present.

Note: The purpose of a code system name is to assist an unaided human interpreter of a code value to interpret codeSystem.

codeSystemVersion

Type: xs:string, predefined
Use: optional

If applicable, a version descriptor defined specifically for the given code system.

valueSet

Type: dt:Uid [147]
Use: optional

The value set that applied when this CD was created.

Attribute Value

xs:string

valueSetVersion

Type: xs:string, predefined
Use: optional

The version of the valueSet in which the code was found.

Content Element Detail (all declarations; defined within this component only; 3/3)

dt:displayName

Type: dt:ST [124], empty content

A name, title, or representation for the code or expression as it exists in the code system.
If populated, the displayName SHALL be a valid human readable representation of the concept as defined by the code system at the time of data entry. The displayName SHALL conform to any rules defined by the codingSystem; if the codeSystem does not define a human representation for the code or expression, then none can be provided. displayName is included both as a courtesy to an unaided human interpreter of a code value and as a documentation of the name used to display the concept to the user. The display name has no functional meaning; it SHALL never exist without a code; and it SHALL never modify the meaning of the code. A display name may not be present if the code is an expression for which no display name has been assigned or can be derived. Information Processing Entities claiming direct or indirect conformance MAY choose not to implement displayName but SHALL NOT reject instances because displayName is present.
Display names SHALL not alter the meaning of the code value. Therefore, display names SHOULD NOT be presented to the user on a receiving application system without ascertaining that the display name adequately represents the concept referred to by the code value. Communication SHALL NOT simply rely on the display name. The display name's main purpose is to support implementation debugging.

XML Representation Summary

```
<dt:displayName
    value = xs:string
/>
```

dt:originalText

Type: dt:ST [124], empty content

The text as seen and/or selected by the user who entered the data which represents the intended meaning of the user.

Note: Local implementations may influence what is required to represent that original text.

Original text can be used in a structured user interface to capture what the user saw as a representation of the code on the data input screen, or in a situation where the user dictates or directly enters text, it is the text entered or uttered by the user.

It is valid to use the CD datatype to store only the text that the user entered or uttered. In this situation, original text will exist without a code. In a situation where the code is assigned sometime after the text was entered, originalText is the text or phrase used as the basis for assigning the code.

The original text SHALL be an excerpt of the relevant information in the original sources, rather than a pointer or exact reproduction. Thus the original text SHALL be represented in plain text form. In specific circumstances, when clearly descirbed

the context of use, the originalText may be a reference to some other text artefact for which the resolution scope is clearly described.

Values of type CD MAY have a original text despite not having a code. Any CD value with no code signifies a coding exception. In this case, originalText is a name or description of the concept that was not coded.

XML Representation Summary

```
<dt:originalText
    value = xs:string
/>
```

dt:translation

Type: dt:CD [85], complex content

Translation of the base code / codeSystem to other codeSystems.

XML Representation Summary

```
<dt:translation
    code              = xs:string
    codeSystem        = xs:string
    codeSystemName    = xs:string
    codeSystemVersion = xs:string
    valueSet          = xs:string
    valueSetVersion   = xs:string
    >
    Content: dt:displayName?, dt:originalText?, dt:translation*
</dt:translation>
```

complexType "dt:CO"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
    value = xs:double
    >
    Content: dt:code?
</...>
```

Content Model Elements (1):

dt:code [90]

Annotation

Represents data where coded values are associated with a specific order.

Note: CO may be used for things that model rankings and scores, e.g. likert scales, 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. CO may also be used in the context of an ordered code system. In this case, it may not be appropriate or even possible to use the value attribute, but CO may still be used so that models that make use of such code systems may introduce model elements

that involve statements about the order of the terms in a domain.

The relative order of values in a code system need not be independently obvious in the literal representation of the CO. It these circumstances, is expected that an application will look up the ordering of these values from some definition of the code system.

Some of the code systems will directly assign numerical value to the concepts that are suitable for some mathemetical operations.

Though it would generally make sense, applications SHOULD not assume that the translations of the code, if provided, will have the same ordering as the CO. Translations SHALL not be considered when the ordering of the code system is determined.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:CO
```

XML Source (w/o annotations (3))

```
<xs:complexType name="CO">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence>
        <xs:element minOccurs="0" name="code" type="CD"/>
      </xs:sequence>
      <xs:attribute name="value" type="Decimal" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: dt:Decimal [134]
Use: optional

A numerical value associated with the coded ordinal value.
The value may be constrained to an integer in some contexts of use. If code is nonNull, value SHALL only be nonNull if the code system explicitly assigns a value to the concept.

Attribute Value

xs:double

Content Element Detail (all declarations; defined within this component only; 1/1)

dt:code

Type: dt:CD [85], complex content

A code representing the definition of the ordinal item

XML Representation Summary

```
<dt:code
    code              = xs:string
    codeSystem        = xs:string
    codeSystemName    = xs:string
    codeSystemVersion = xs:string
    valueSet          = xs:string
    valueSetVersion   = xs:string
    >
    Content: dt:displayName?, dt:originalText?, dt:translation*
</dt:code>
```

complexType "dt:CS"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    code = xs:string
/>
```

Annotation

Coded data in its simplest form, where only the code is not predetermined.

The code system and code system version are implied and fixed by the context in which the CS value occurs.

Due to its highly restricted functionality, CS SHALL only be used for simple structural attributes with highly controlled and stable terminologies where:
- all codes come from a single code system
- codes are not reused if their concept is deprecated
- the publication and extensibility properties of the code system are well described and understood

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:CS
```

XML Source (w/o annotations (2))

```
<xs:complexType name="CS">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
      <xs:attribute name="code" type="xs:string" use="required"/>
```

```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

code

Type: `xs:string`, predefined
Use: required

The plain code symbol defined by the code system. If the code value is empty or null, then there is no code in the code system that represents the concept.
Code SHALL only contain characters that are either a letter, a digit, or one of '.', '-', '_' or ':'. Code systems that are used with CS SHALL NOT define code symbols or expression syntaxes that contain whitespace or any other characters not in this list.

complexType "dt:ED"

Namespace: `urn:hl7-org:cdsdt:r2`
Includes: definitions of 6 attributes, 5 elements

XML Representation Summary

```
<...
   value                  = xs:string
   mediaType              = xs:string : "text/plain"
   charset                = xs:string
   language               = xs:string
   compression            = ("DF" | "GZ" | "ZL" | "Z" | "BZ" | "Z7")
   integrityCheckAlgorithm = ("SHA1" | "SHA256")
   >
   Content: dt:data?, dt:xml?, dt:reference?, dt:integrityCheck?, dt:description?
</...>
```

Content Model Elements (5):

dt:data [95], dt:description [96], dt:integrityCheck [96], dt:reference [95], dt:xml [95]

Annotation

Data that is primarily intended for human interpretation or for further machine processing outside the scope of this specification. This includes unformatted or formatted written language, multimedia data, or structured information as defined by a different standard (e.g., XML-signatures.)
Encapsulated data can be present in two forms, inline or by reference. The content is the same whether it is located inline or remote. Inline data is communicated or moved as part of the encapsulated data value, whereas by-reference data may reside at a different location: a URL/URI that provides reference to the information required to locate the data. Inline data may be provided in one of 3 different ways:

1) as a plain sequence of characters (value)
2) as a binary (a sequence of bytes) (data
3) as xml content (xml)

Content SHALL be provided if the ED has no nullFlavor. The content may be provided in-line (using only one of value, data or xml), or it may be provided as a reference.Content may be provided in-line and a reference also may be given; in these cases, it

is expected that the content of the reference will be exactly the same as the in-line content. Information Processing Entities are not required to check this, but may regard it as an error condition if the content does not match

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:ED
```

XML Source (w/o annotations (12))

```
<xs:complexType name="ED">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence>
        <xs:element minOccurs="0" name="data" type="xs:base64Binary"/>
        <xs:element minOccurs="0" name="xml" type="xs:anyType"/>
        <xs:element minOccurs="0" name="reference" type="TEL"/>
        <xs:element minOccurs="0" name="integrityCheck" type="xs:base64Binary"/>
        <xs:element minOccurs="0" name="description" type="ST"/>
      </xs:sequence>
      <xs:attribute name="value" type="xs:string" use="optional"/>
      <xs:attribute default="text/plain" name="mediaType" type="Code" use="optional"/>
      <xs:attribute name="charset" type="Code" use="optional"/>
      <xs:attribute name="language" type="Code" use="optional"/>
      <xs:attribute name="compression" type="Compression" use="optional"/>
      <xs:attribute name="integrityCheckAlgorithm" type="IntegrityCheckAlgorithm" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 6/6)

value

Type: `xs:string`, predefined
Use: optional

A simple sequence of characters that contains the content.

If value is used, the mediatype is fixed to text/plain and the charset must be consistent with the String Character Set. Refer to section 6.7.5 for more details

mediaType

Type: `dt:Code [132]`
Use: optional

Identifies the type of the encapsulated data and can be used to determine a method to interpret or render the content.

The IANA defined domain of media types is established by the IETF RFCs 2045 and 2046. mediaType has a default value of

text/plain and cannot be null. If the media type is different to text/plain, the <i>mediaType</i> attribute SHALL be populated.

If the content is compressed using a specified compression algorithm, the mediaType SHALL refer the mediaType of the uncompressed data, whether the data is accessed by reference or not.

Attribute Value

```
xs:string
```

Default: `"text/plain"`

charset

Type: `dt:Code [132]`
Use: optional

An Internet Assigned Numbers Authority (IANA) Charset Registered character set and character encoding for character-based encoding types<b>. </b>

Whenever the content of the ED is character type data in any form, the charset property needs to be known. If the content is provided directly in the value attribute, then the charset SHALL be a known character set consistent with the String Character Set. Refer to section 6.7.5 for more details. If the content is provided as a reference, and the access method does not provide the charset for the content (such as by a mime header), then the charset SHALL be conveyed as part of the ED

Attribute Value

```
xs:string
```

language

Type: `dt:Code [132]`
Use: optional

The human language of the content. Valid codes are taken from the IETF RFC 3066. If this attribute is null, the language may be inferred from elsewhere, either from the context or from unicode language tags, for example.

Conformance profiles SHOULD define defaulting rules for language for a given usage environment of this specification.

Note: While language attribute usually alters the interpretation of the text, the language attribute does not alter the meaning of the characters in the text.

Attribute Value

```
xs:string
```

compression

Type: `dt:Compression [133]`
Use: optional

The compression algorithm, if any, used on the raw byte data.

If the attribute is null, the data is not compressed. Compression only applies to the binary form of the content.

If populated, the value of this attribute SHALL be taken from the HL7 CompressionAlgorithm code system.

Some compression formats allow multiple archive files to be embedded within a single compressed volume. Applications SHALL ensure that the decompressed form of the data conforms to the stated media type.

Attribute Value

```
enumeration of xs:string
```

Enumeration:

"DF" – Deflate : The deflate compressed data format as specified in IETF RFC 1951.

"GZ" – GZIP : A compressed data format that is compatible with the widely used GZIP utility as specified in IETF RFC 1952(uses the deflate algorithm).

"ZL" – ZLIB : A compressed data format that also uses the deflate algorithm. Specified as IETF RFC 1950.

"Z" – Compress : Original UNIX compress algorithm and file format using the LZC algorithm (a variant of LZW). Patent encumbered and less efficient than deflate.

"BZ" – BZIP : bzip-2 compression format. See [http://www.bzip.org/] for more information.

"Z7" – Z7 : 7z compression file format. See [http://www.7-zip.org/7z.html] for more information.

integrityCheckAlgorithm

Type: dt:IntegrityCheckAlgorithm [139]
Use: optional

The algorithm used to compute the integrityCheck value.
If populated, the value of this attribute SHALL be taken from the HL7 IntegrityCheckAlgorithm code system.

Attribute Value

```
enumeration of xs:string
```

Enumeration:

"SHA1" – Secure Hash Algorithm - 1 : This algorithm is defined in FIPS PUB 180-1: Secure Hash Standard. As of April 17, 1995

"SHA256" – Secure Hash Algorithm - 256 : This algorithm is defined in FIPS PUB 180-2: Secure Hash Standard

Content Element Detail (all declarations; defined within this component only; 5/5)

dt:data

Type: xs:base64Binary, predefined, simple content

A simple sequence of byte values that contains the content. (Base64 Encoded String).

XML Representation Summary

```
<dt:data>
    Content:        { xs:base64Binary }
</dt:data>
```

dt:xml

Type: xs:anyType, any content

The content represented in plain XML form.

A direct representation is provided for XML. This is because this specification includes an XML serialization of the data, and this xml attribute is handled specially in the serialisation form. The xml data is not different in any semantic sense to the same data if represented in the value or data attributes.

XML Representation Summary

```
<dt:xml>
    ...
</dt:xml>
```

dt:reference

Type: dt:TEL [124], empty content

A URL the target of which provides the binary content.

The semantic value of an encapsulated data value is the same, regardless whether the content is present as inline content or just by reference. However, an encapsulated data value without inline content behaves differently, since any attempt to examine the content requires the data to be downloaded from the reference. An encapsulated data value may have both inline content and a reference.

If data is provded in the value, data or xml attributes, the reference SHALL point to the same data. It is an error if the data resolved through the reference does not match either the integrity check, data as provided, or data that had earlier been retrieved through the reference and then cached. The mediatype of the ED SHALL match the type returned by accessing the reference.

The reference may contain a usablePeriod to indicate that the data may only be available for a limited period of time. Whether the reference is limited by a usablePeriod or not, the content of the reference SHALL be fixed for all time. Any application using the reference SHALL always receive the same data, or an error. The reference cannot be reused to send a different version of the same data, or different data

XML Representation Summary

```
<dt:reference
    capabilities = list of ("voice" | "fax" | "data" | "tty" | "sms")
    use          = list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" | "MC" | "PG")
    value        = xs:anyURI
/>
```

dt:integrityCheck

Type: `xs:base64Binary`, predefined, simple content

A checksum calculated over the binary data

The purpose of this property, when communicated with a reference is for anyone to validate later whether the reference still resolved to the same content that the reference resolved to when the encapsulated data value with reference was created. If the attribute is null, there is no integrityCheck.
It is an error if the data resolved through the reference does not match the integrity check.
The integrity check is calculated according to the integrityCheckAlgorithm. By default, the Secure Hash Algorithm-1 (SHA-1) shall be used. The integrity check is binary encoded according to the rules of the integrity check algorithm.
The integrity check is calculated over the raw binary data that is contained in the data component, or that is accessible through the reference. No transformations are made before the integrity check is calculated. If the data is compressed, the Integrity Check is calculated over the compressed data.

XML Representation Summary

```
<dt:integrityCheck>
    Content:        { xs:base64Binary }
</dt:integrityCheck>
```

dt:description

Type: `dt:ST [124]`, empty content

An alternative description of the media where the media is not able to be rendered.

E.g. Short text description of an image or sound clip, etc. This attribute is not intended to be a complete substitute for the original. For complete substitutes, use the "translation" property.

The intent of this property is to allow compliance with disability requirements such as those expressed in American's with Disability Act (also known as "Section 508"), where there is a requirement to provide a short text description of included media in some form that can be read by a screen reader. This is similar to a very short thumbnail with mediaType = text/plain.

XML Representation Summary

```
<dt:description
    value = xs:string
/>
```

complexType "dt:EN"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   use = list of ("ABC" | "IDE" | "SYL" | "C" | "OR" | "T" | "I" | "P" | "ANON" | "A" | "R" | "OLD" |
         "DN" | "M" | "PHON" | "SRCH")
   >
   Content: dt:part+
</...>
```

Content Model Elements (1):

dt:part (in name defined in Person complexType) [98]

Annotation

A name for a person, organization, place or thing.

Examples: Jim Bob Walton, Jr., Health Level Seven, Inc., Lake Tahoe, etc. An entity name may be as simple as a character string or may consist of several entity name parts, such as, Jim, Bob, Walton, and Jr., Health Level Seven, and Inc.

Entity names are essentially sequences of entity name parts, but add a "use" code.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:EN
```

XML Source (w/o annotations (3))

```
<xs:complexType name="EN">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="part" type="ENXP"/>
      </xs:sequence>
      <xs:attribute name="use" type="set_EntityNameUse" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

use

Type: dt:set_EntityNameUse [142]
Use: optional

A set of codes advising a system or user which name in a set of names to select for a given purpose.
A name without specific use code might be a default name useful for any purpose, but a name with a specific use code would be preferred for that respective purpose. Names SHOULD not be collected without at least one use code, but names MAY exist without use code, particularly for legacy data.
If populated, the values contained in this attribute SHALL be taken from the HL7 EntityNameUse2 code system.

Attribute Value

```
list of ("ABC" | "IDE" | "SYL" | "C" | "OR" | "T" | "I" | "P" | "ANON" | "A" | "R" | "OLD" | "DN" | "M" |
"PHON" | "SRCH")
```

Content Element Detail (all declarations; defined within this component only; 1/1)

dt:part

Type: dt:ENXP [98], empty content

A sequence of name parts, such as given name or family name, prefix, suffix, etc.

XML Representation Summary

```
<dt:part
    qualifier = list of ("LS" | "AC" | "NB" | "PR" | "HON" | "BR" | "AD" | "SP" | "MID" | "CL" | "IN" | "PFX" | "SFX")
    type      = ("FAM" | "GIV" | "TITLE" | "DEL")
    value     = xs:string
/>
```

complexType "dt:ENXP"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
    value      = xs:string
    type       = ("FAM" | "GIV" | "TITLE" | "DEL")
    qualifier  = list of ("LS" | "AC" | "NB" | "PR" | "HON" | "BR" | "AD" | "SP" | "MID" | "CL" | "IN" |
                 "PFX" | "SFX")
/>
```

Annotation

A part with a type code signifying the role of the part in the whole entity name, and qualifier codes for more detail about the name part type. (Typical name parts for person names are given names, and family names, titles, etc.).

Type Definition Detail

Type Derivation Tree

dt:XP [127] (extension)

└ **dt:ENXP**

XML Source (w/o annotations (3))

```
<xs:complexType name="ENXP">
  <xs:complexContent>
    <xs:extension base="XP">
      <xs:sequence/>
      <xs:attribute name="type" type="EntityNamePartType" use="required"/>
      <xs:attribute name="qualifier" type="set_EntityNamePartQualifier" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/3)

type

Type: dt:EntityNamePartType [136]
Use: required

Indicates whether the name part is a given name, family name, prefix, suffix, etc.
The value of this attribute SHALL be taken from the HL7 EntityNamePartType2 code system.

Attribute Value

enumeration of xs:string

Enumeration:

- "FAM" – Family : Family name, this is the name that links to the genealogy. In some cultures (e.g. Eritrea) the family name of a son is the first name of his father
- "GIV" – Given: Given name.
  Note: don't call it "first name" since this given names do not always come first
- "TITLE" – Title : Part of the name that is acquired as a title due to academic, legal, employment or nobility status etc.
  Note: Title name parts include name parts that come after the name such as qualifications
- "DEL" – Delimiter : A delimiter has no meaning other than being literally printed in this name representation. A delimiter has no implicit leading and trailing white space

qualifier

Type: dt:set_EntityNamePartQualifier [142]
Use: optional

The qualifier is a set of codes each of which specifies a certain subcategory of the name part in addition to the main name part type. For example, a given name may be flagged as a nickname (CL), a family name may be a name acquired by marriage (SP) or a name from birth (BR).
If populated, the values contained in this attribute SHALL be taken from the HL7 EntityNamePartQualifier2 code system.

Attribute Value

list of ("LS" | "AC" | "NB" | "PR" | "HON" | "BR" | "AD" | "SP" | "MID" | "CL" | "IN" | "PFX" | "SFX")

complexType "dt:HXIT"

Namespace: urn:hl7-org:cdsdt:r2
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*

XML Representation Summary

```
<.../>
```

Annotation

Information about the history of this value: period of validity and a reference to an identified event that established this value as valid.

Because of the way that the types are defined, a number of attributes of the datatypes have values with a type derived from HXIT. In these cases the HXIT attributes are constrained to null. The only case where the HXIT attributes are allowed within a datatype is on items in a collection (DSET, LIST, BAG, HIST).
The use of these attributes is generally subject to further constraints in the specifications that make use of these types.

This class is maintained here despite the lack of attributes to maintain compatibility with the ISO 21090 data structure.

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="HXIT">
  <xs:sequence/>
</xs:complexType>
```

complexType "dt:II"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 3 attributes

XML Representation Summary

```
<...
   root           = xs:string
   extension      = xs:string
   identifierName = xs:string
/>
```

Annotation

An identifier that uniquely identifies a thing or object.

Examples are object identifier for HL7 RIM objects, medical record number, order id, service catalog item id, Vehicle Identification Number (VIN), etc. Instance identifiers are usually defined based on ISO object identifiers.

An identifier allows someone to select one record, object or thing from a set of candidates. Usually an identifier alone without any context is not usable. Identifiers are distinguished from concept descriptors as concept descriptors never identify an individual thing, although there may sometimes be an individual record or object that represents the concept.

Information Processing Entities claiming direct or indirect conformance SHALL never assume that receiving applications can infer the identity of issuing authority or the type of the identifier from the identifier or components thereof.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:II
```

XML Source (w/o annotations (4))

```
<xs:complexType name="II">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
      <xs:attribute name="root" type="Uid" use="required"/>
      <xs:attribute name="extension" type="xs:string" use="optional"/>
      <xs:attribute name="identifierName" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 3/3)

root

Type: dt:Uid [147]
Use: required

A unique identifier that guarantees the global uniqueness of the instance identifier.
If root is populated, and there is no extension, then the root is a globally unique identifier in its own right. In the presence of a non-null extension, the root is the unique identifier for the "namespace" of the identifier in the extension. Note that this does NOT necessarily correlate with the organization that manages the issuing of the identifiers. A given organization may manage multiple identifier namespaces, and control over a given namespace may transfer from organization to organization over time while the root remains the same.
This field can be either a DCE UUID, an Object Identifier (OID), or a special identifier taken from lists that may be published by ISO or HL7.
Comparison of root values is always case sensitive. UUID's SHALL be represented in upper case, so UUID case should always be preserved.
The root SHALL not be used to carry semantic meaning - all it does is ensure global computational uniqueness.

Attribute Value

xs:string

extension

Type: xs:string, predefined
Use: optional

A character string as a unique identifier within the scope of the identifier root.
The root and extension scheme means that the concatenation of root and extension SHALL be a globally unique identifier for the item that this II value identifies.
Some identifier schemes define certain style options to their code values. For example, the U.S. Social Security Number (SSN) is normally written with dashes that group the digits into a pattern "123-12-1234". However, the dashes are not meaningful and a SSN can also be represented as "123121234" without the dashes. In the case where identifier schemes provide for multiple representations, HL7 or ISO may make a ruling about which is the preferred form and document that ruling where that respective external identifier scheme is recognized.
If no extension attribute is provided in a non-null II, then the root is the complete unique identifier.

identifierName

Type: xs:string, predefined
Use: optional

A human readable description for this identifier.

complexType "dt:INT"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = xs:integer
/>
```

Annotation

Integer numbers (-1,0,1,2, 100, 3398129, etc.) are precise numbers that are results of counting and enumerating. Integer numbers are discrete, the set of integers is infinite but countable. No arbitrary limit is imposed on the range of integer numbers.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:INT
```

XML Source (w/o annotations (2))

```
<xs:complexType name="INT">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence/>
      <xs:attribute name="value" type="xs:integer" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: xs:integer, predefined
Use: required

The value of the INT. Note that this specification imposes no limitations on the size of integer, but most implementations will map this to a 32 or 64 bit integer.

complexType "dt:IVL"

Namespace: urn:hl7-org:cdsdt:r2

XML Representation Summary

<...>

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds

This class is maintained here despite the lack of attributes to maintain compatibility with the ISO 21090 data structure.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL
```

XML Source (w/o annotations (1))

```
<xs:complexType name="IVL">
  <xs:complexContent>
    <xs:extension base="QSET">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

complexType "dt:IVL_CO"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
    lowClosed   =  xs:boolean
    highClosed  =  xs:boolean
    >
    Content: dt:low?, dt:high?
</...>
```

Content Model Elements (2):

dt:high (defined in dt:IVL_CO complexType) [105], dt:low (defined in dt:IVL_CO complexType) [105]

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL [103] (extension)
              └ dt:IVL_CO
```

XML Source (w/o annotations (5))

```
<xs:complexType name="IVL_CO">
  <xs:complexContent>
    <xs:extension base="IVL">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="CO"/>
        <xs:element minOccurs="0" name="high" type="CO"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: xs:boolean, predefined
Use: optional

This attribute is called lowIsClosed in the ISO 21090 specification and lowClosed in the HL7 Data Types R2 specification.

Whether low is included in the IVL (is closed) or excluded from the IVL (is open).

highClosed

Type: xs:boolean, predefined
Use: optional

This attribute is called highIsClosed in the ISO 21090 specification and highClosed in the HL7 Data Types R2 specification.

Whether high is included in the IVL (is closed) or excluded from the IVL (is open).

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:low

Type: dt:CO [89], complex content

This is the low limit. If the low limit is not known, it may be null.
The low limit SHALL NOT be positive infinity.

XML Representation Summary

```
<dt:low
    value = xs:double
    >
    Content: dt:code?
</dt:low>
```

dt:high

Type: dt:CO [89], complex content

This is the high limit. If the high limit is not known, it may be null.
The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

XML Representation Summary

```
<dt:high
    value = xs:double
    >
    Content: dt:code?
</dt:high>
```

complexType "dt:IVL_INT"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
   lowClosed   = xs:boolean
   highClosed  = xs:boolean
   >
   Content: dt:low?, dt:high?
</...>
```

Content Model Elements (2):

dt:high (defined in dt:IVL_INT complexType) [107], dt:low (defined in dt:IVL_INT complexType) [107]

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL [103] (extension)
              └ dt:IVL_INT
```

XML Source (w/o annotations (5))

```
<xs:complexType name="IVL_INT">
  <xs:complexContent>
    <xs:extension base="IVL">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="INT"/>
        <xs:element minOccurs="0" name="high" type="INT"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called lowIsClosed in the ISO 21090 specification and lowClosed in the HL7 Data Types R2 specification.

Whether low is included in the IVL (is closed) or excluded from the IVL (is open).

highClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called highIsClosed in the ISO 21090 specification and highClosed in the HL7 Data Types R2 specification.

Whether high is included in the IVL (is closed) or excluded from the IVL (is open).

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:low

Type: `dt:INT [102]`, empty content

This is the low limit. If the low limit is not known, it may be null.
The low limit SHALL NOT be positive infinity.

XML Representation Summary

```
<dt:low
    value = xs:integer
/>
```

dt:high

Type: `dt:INT [102]`, empty content

This is the high limit. If the high limit is not known, it may be null.
The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

XML Representation Summary

```
<dt:high
    value = xs:integer
/>
```

complexType "dt:IVL_PQ"

Namespace: `urn:hl7-org:cdsdt:r2`
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
   lowClosed   = xs:boolean
   highClosed  = xs:boolean
   >
   Content: dt:low?, dt:high?
</...>
```

Content Model Elements (2):

dt:high (defined in dt:IVL_PQ complexType) [109], dt:low (defined in dt:IVL_PQ complexType) [109]

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL [103] (extension)
              └ dt:IVL_PQ
```

XML Source (w/o annotations (5))

```
<xs:complexType name="IVL_PQ">
  <xs:complexContent>
    <xs:extension base="IVL">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="PQ"/>
        <xs:element minOccurs="0" name="high" type="PQ"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called lowIsClosed in the ISO 21090 specification and lowClosed in the HL7 Data Types R2 specification.

Whether low is included in the IVL (is closed) or excluded from the IVL (is open).

highClosed

Type: `xs:boolean`, predefined
Use: optional

Whether high is included in the IVL (is closed) or excluded from the IVL (is open).

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:low

Type: `dt:PQ [118]`, empty content

This is the low limit. If the low limit is not known, it may be null.
The low limit SHALL NOT be positive infinity.

XML Representation Summary

```
<dt:low
    unit  = xs:string
    value = xs:double
/>
```

dt:high

Type: `dt:PQ [118]`, empty content

This is the high limit. If the high limit is not known, it may be null.
The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

XML Representation Summary

```
<dt:high
    unit  = xs:string
    value = xs:double
/>
```

complexType "dt:IVL_QTY"

Namespace: `urn:hl7-org:cdsdt:r2`
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
   lowClosed   = xs:boolean
   highClosed  = xs:boolean
   >
   Content: dt:low?, dt:high?
</...>
```

Content Model Elements (2):

dt:high (defined in dt:IVL_QTY complexType) [111], dt:low (defined in dt:IVL_QTY complexType) [111]

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL [103] (extension)
              └ dt:IVL_QTY
```

XML Source (w/o annotations (5))

```
<xs:complexType name="IVL_QTY">
  <xs:complexContent>
    <xs:extension base="IVL">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="QTY"/>
        <xs:element minOccurs="0" name="high" type="QTY"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called lowIsClosed in the ISO 21090 specification and lowClosed in the HL7 Data Types R2 specification.

Whether low is included in the IVL (is closed) or excluded from the IVL (is open).

highClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called highIsClosed in the ISO 21090 specification and highClosed in the HL7 Data Types R2 specification.

Whether high is included in the IVL (is closed) or excluded from the IVL (is open).

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:low

Type: *dt:QTY* [121], empty content

This is the low limit. If the low limit is not known, it may be null.
The low limit SHALL NOT be positive infinity.

XML Representation Summary

```
<dt:low/>
```

dt:high

Type: *dt:QTY* [121], empty content

This is the high limit. If the high limit is not known, it may be null.
The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

XML Representation Summary

```
<dt:high/>
```

complexType "dt:IVL_REAL"

Namespace: `urn:hl7-org:cdsdt:r2`
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
   lowClosed   = xs:boolean
   highClosed  = xs:boolean
   >
   Content: dt:low?, dt:high?
</...>
```

Content Model Elements (2):

dt:high (defined in dt:IVL_REAL complexType) [113], dt:low (defined in dt:IVL_REAL complexType) [113]

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL [103] (extension)
              └ dt:IVL_REAL
```

XML Source (w/o annotations (5))

```
<xs:complexType name="IVL_REAL">
  <xs:complexContent>
    <xs:extension base="IVL">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="REAL"/>
        <xs:element minOccurs="0" name="high" type="REAL"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called lowIsClosed in the ISO 21090 specification and lowClosed in the HL7 Data Types R2 specification.

Whether low is included in the IVL (is closed) or excluded from the IVL (is open).

highClosed

Type: `xs:boolean`, predefined
Use: optional

This attribute is called highIsClosed in the ISO 21090 specification and highClosed in the HL7 Data Types R2 specification.

Whether high is included in the IVL (is closed) or excluded from the IVL (is open).

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:low

Type: `dt:REAL [121]`, empty content

This is the low limit. If the low limit is not known, it may be null.
The low limit SHALL NOT be positive infinity.

XML Representation Summary

```
<dt:low
    value = xs:double
/>
```

dt:high

Type: `dt:REAL [121]`, empty content

This is the high limit. If the high limit is not known, it may be null.
The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

XML Representation Summary

```
<dt:high
    value = xs:double
/>
```

complexType "dt:IVL_TS"

Namespace: `urn:hl7-org:cdsdt:r2`
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
   lowClosed   = xs:boolean
   highClosed  = xs:boolean
   >
   Content: dt:low?, dt:high?
</...>
```

Content Model Elements (2):

dt:high (defined in dt:IVL_TS complexType) [115], dt:low (defined in dt:IVL_TS complexType) [115]

Annotation

A set of consecutive values of an ordered base datatype.

Any ordered type can be the basis of an IVL; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the IVL must be elements of a totally ordered subset of the partially ordered datatype. For example, PQ is considered ordered. However the ordering of PQs is only partial; a total order is only defined among comparable quantities (quantities of the same physical dimension). While IVLs between 2 and 4 meter exists, there is no IVL between 2 meters and 4 seconds.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QSET [120] (extension)
          └ dt:IVL [103] (extension)
              └ dt:IVL_TS
```

XML Source (w/o annotations (5))

```
<xs:complexType name="IVL_TS">
  <xs:complexContent>
    <xs:extension base="IVL">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="TS"/>
        <xs:element minOccurs="0" name="high" type="TS"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: xs:boolean, predefined
Use: optional

This attribute is called lowIsClosed in the ISO 21090 specification and lowClosed in the HL7 Data Types R2 specification.

Whether low is included in the IVL (is closed) or excluded from the IVL (is open).

highClosed

Type: xs:boolean, predefined
Use: optional

This attribute is called highIsClosed in the ISO 21090 specification and highClosed in the HL7 Data Types R2 specification.

Whether high is included in the IVL (is closed) or excluded from the IVL (is open).

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:low

Type: dt:TS [126], empty content

This is the low limit. If the low limit is not known, it may be null.
The low limit SHALL NOT be positive infinity.

XML Representation Summary

```
<dt:low
    value = xs:string
/>
```

dt:high

Type: dt:TS [126], empty content

This is the high limit. If the high limit is not known, it may be null.
The high limit SHALL NOT be negative infinity, and SHALL be higher than the low limit if one exists.

XML Representation Summary

```
<dt:high
    value = xs:string
/>
```

complexType "dt:PIVL_TS"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 2 attributes, 4 elements

XML Representation Summary

```
<...
    alignment   = ("CY" | "MY" | "CM" | "CW" | "WM" | "WY" | "DM" | "CD" | "DY" | "DW" | "HD" | "CH" |
                  "NH" | "CN" | "SN" | "CS")
    isFlexible  = xs:boolean
    >
    Content: dt:phase?, dt:period?, dt:frequency?, dt:count?
</...>
```

Content Model Elements (4):

dt:count [118], dt:frequency [118], dt:period [117], dt:phase [117]

Annotation

An interval of time that recurs periodically. PIVL has two properties, phase and period/frequency. phase specifies the "interval prototype" that is repeated on the period/frequency.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:PIVL_TS
```

XML Source (w/o annotations (6))

```
<xs:complexType name="PIVL_TS">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence>
        <xs:element minOccurs="0" name="phase" type="IVL_TS"/>
        <xs:element minOccurs="0" name="period" type="PQ"/>
        <xs:element minOccurs="0" name="frequency" type="RTO"/>
        <xs:element minOccurs="0" name="count" type="INT"/>
      </xs:sequence>
      <xs:attribute name="alignment" type="CalendarCycle" use="optional"/>
      <xs:attribute name="isFlexible" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

alignment

Type: dt:CalendarCycle [131]
Use: optional

Attribute Value

```
enumeration of xs:string
```

Enumeration:

"CY" – year
"MY" – month of the year
"CM" – month (continuous)
"CW" – week (continuous)
"WM" – week of the month
"WY" – week of the year
"DM" – day of the month
"CD" – day (continuous)
"DY" – day of the year
"DW" – day of the week (begins with monday)
"HD" – hour of the day
"CH" – hour (continuous)
"NH" – minute of the hour
"CN" – minute (continuous)
"SN" – second of the minute
"CS" – second (continuous)

isFlexible

Type: xs:boolean, predefined
Use: optional

Indicates whether the exact timing is up to the party executing the schedule e.g., to distinguish "every 8 hours" from "3 times a day".

Note: this is sometimes referred to as "institution specified timing".

Content Element Detail (all declarations; defined within this component only; 4/4)

dt:phase

Type: dt:IVL_TS [113], complex content

A prototype of the repeating interval, specifying the duration of each occurrence and anchors the PIVL sequence at a certain point in time. phase also marks the anchor point in time for the entire series of periodically recurring intervals. If count is null or nullFlavored, the recurrence of a PIVL has no beginning or ending, but is infinite in both future and past.

The width of the phase SHALL be less than or equal to the period

XML Representation Summary

```
<dt:phase
    highClosed = xs:boolean
    lowClosed  = xs:boolean
    >
    Content: dt:low?, dt:high?
</dt:phase>
```

dt:period

Type: dt:PQ [118], empty content

A time duration specified as a reciprocal measure of the frequency at which the PIVL repeats.

XML Representation Summary

```
<dt:period
    unit  = xs:string
    value = xs:double
/>
```

dt:frequency

Type: dt:RTO [122], complex content

The number of times the PIVL repeats (numerator) within a specified time-period (denominator). The numerator is an integer, and the denominator is a PQ.TIME.

Only one of period and frequency should be specified. The form chosen should be the form that most naturally conveys the idea to humans. i.e. Every 10 mins (period) or twice a day (frequency).

XML Representation Summary

```
<dt:frequency>
    Content: dt:numerator, dt:denominator
</dt:frequency>
```

dt:count

Type: dt:INT [102], empty content

The number of times the period repeats in total. If count is null, then the period repeats indefinitely both before and after the anchor implicit in the phase.

XML Representation Summary

```
<dt:count
    value = xs:integer
/>
```

complexType "dt:PQ"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
    value = xs:double
    unit  = xs:string
/>
```

Annotation

A dimensioned quantity expressing the result of measuring.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:PQ
```

XML Source (w/o annotations (3))

```
<xs:complexType name="PQ">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence/>
      <xs:attribute name="value" type="Decimal" use="required"/>
      <xs:attribute name="unit" type="Code" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

value

Type: dt:Decimal [134]
Use: required

The number which is multiplied by the unit to make the PQ.

Attribute Value

xs:double

unit

Type: dt:Code [132]
Use: required

The unit of measure specified in the Unified Code for Units of Measure (UCUM).
UCUM defines two forms of expression, case sensitive and case insensitive. PQ uses the case sensitive codes. The codeSystem OID for the case sensitive form is 2.16.840.1.113883.6.8. The default value for unit is the UCUM code "1" (unity).
Equality of physical quantities does not require the values and units to be equal independently. Value and unit is only how we represent physical quantities. For example, 1 m equals 100 cm. Although the units are different and the values are different, the physical quantities are equal. Therefore one should never expect a particular unit for a physical quantity but instead allow for automated conversion between different comparable units.
The unit SHALL come from UCUM, which only specifies unambiguous measurement units. Sometimes it is not clear how some measurements in healthcare map to UCUM codes.
Note: The general pattern for a measurement is value unit of Thing. In this scheme, the PQ represents the value and the unit, and the Thing is described by some coded concept that is linked to the PQ by the context of use. This maps obviously to some measurements, such as Patient Body Temperature of 37 Celsius, and 250 mg/day of Salicylate.
However for some measurements that arise in healthcare, the scheme is not so obvious. Two classic examples are 5 Drinks of Beer, and 3 Acetaminophen tablets. At first glance it is tempting to classify these measurements like this: 5 drinks of Beer and 3 Acetaminophen tablets. The problem with this is that UCUM does not support units of "beer", "tablets" or "scoops".
The reason for this is that neither tablets or scoops are proper units. What kind of tablets? How big is the glass? In these kinds of cases, the concept that appears to be a unit needs to further specified before interoperability is established. If a correct amount is required, then it is generally appropriate to specify an exact measurement with an appropriate UCUM unit. If this is not possible, then the concept is not part of the measurement. UCUM provides a unit called unity for use in these cases. The proper way to

understand these measurements as 3 1 Acetaminophen tablets, where 1 is the UCUM unit for unity, and the Thing has a qualifier. The context of use will need to provide the extra qualifying information.

Attribute Value

`xs:string`

complexType "dt:QSET"

Namespace: urn:hl7-org:cdsdt:r2
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*

XML Representation Summary

<...>

Annotation

Abstract; specializes ANY

Parameter: T : QTY

An unordered set of distinct values which are quantities.

Any ordered type can be the basis of an QSET; it does not matter whether the base type is discrete or continuous. If the base datatype is only partially ordered, all elements of the QSET must be elements of a totally ordered subset of the partially ordered datatype (for example, PQ is only ordered when the units are consistent. Every value in a QSET(PQ) must have the same canonical unit).

QSET is an abstract type. A working QSET is specified as an expression tree built using a combination of operator (QSI, QSD, QSU, QSP) and component types (QSC, QSS and IVL; and, for TS, PIVL and EIVL).

QSETs SHALL not contain null or nullFlavored values as members of the set.

This class is maintained here despite the lack of attributes to maintain compatibility with the ISO 21090 data structure.

Type Definition Detail

Type Derivation Tree

dt:HXIT [99] (extension)
└ *dt:ANY* [84] (extension)
└ ***dt:QSET***

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="QSET">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

complexType "dt:QTY"

Namespace: urn:hl7-org:cdsdt:r2
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*

XML Representation Summary

```
<.../>
```

Annotation

The quantity datatype is an abstract generalization for all datatypes whose domain values has an order relation (less-or-equal) and where difference is defined in all of the datatype's totally ordered value subsets.

The quantity type abstraction is needed in defining certain other types, such as the interval, and probability distributions.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY
```

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="QTY">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

complexType "dt:REAL"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = xs:double
/>
```

Annotation

Fractional numbers. Typically used whenever quantities are measured, estimated, or computed from other real numbers. The typical representation is decimal, where the number of significant decimal digits is known as the precision.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:REAL
```

XML Source (w/o annotations (2))

```
<xs:complexType name="REAL">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence/>
      <xs:attribute name="value" type="Decimal" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: dt:Decimal [134]
Use: required

The value of the REAL.

Attribute Value

xs:double

complexType "dt:RTO"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: dt:numerator, dt:denominator
</...>
```

Content Model Elements (2):

dt:denominator [123], dt:numerator [123]

Annotation

A quantity constructed as the quotient of a numerator quantity divided by a denominator quantity.
Common factors in the numerator and denominator are not automatically cancelled out.
The RTO datatype supports titers (e.g., "1:128") and other quantities produced by laboratories that truly represent ratios. Ratios

are not simply "structured numerics", particularly blood pressure measurements (e.g. "120/60") are not ratios.
Notes:
1. Ratios are different from rational numbers, i.e., in ratios common factors in the numerator and denominator never cancel out. A ratio of two real or integer numbers is not automatically reduced to a real number. This datatype is not defined to generally represent rational numbers. It is used only if common factors in numerator and denominator are not supposed to cancel out. This is only rarely the case. For observation values, ratios occur almost exclusively with titers. In most other cases, REAL should be used instead of the RTO.
2. Since many implementation technologies expect generics to be collections, or only have one parameter, RTO is not implemented as a generic in this specification. Constraints at the point where the RTO is used will define which form of QTY are used.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:RTO
```

XML Source (w/o annotations (3))

```
<xs:complexType name="RTO">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence>
        <xs:element name="numerator" type="QTY"/>
        <xs:element name="denominator" type="QTY"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

dt:numerator

Type: dt:QTY [121], empty content

The quantity that is being divided in the ratio

XML Representation Summary

```
<dt:numerator/>
```

dt:denominator

Type: dt:QTY [121], empty content

The quantity that divides the numerator in the ratio.
The denominator SHALL not be zero.

XML Representation Summary

```
<dt:denominator/>
```

complexType "dt:ST"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = xs:string
/>
```

Annotation

The character string datatype stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) or direct display. Used for names, symbols, presentation and formal expressions.

A ST SHALL have at least one character or else be null.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:ST
```

XML Source (w/o annotations (2))

```
<xs:complexType name="ST">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

  Type: xs:string, predefined
  Use: required

  The actual content of the string.

complexType "dt:TEL"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definitions of 3 attributes

XML Representation Summary

```
<...
    value        = xs:anyURI
    use          = list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" |
                   "MC" | "PG")
    capabilities = list of ("voice" | "fax" | "data" | "tty" | "sms")
/>
```

Annotation

A locatable resource that is identified by a URI, such as a web page, a telephone number (voice, fax or some other resource mediated by telecommunication equipment), an e-mail address, or any other locatable resource that can be specified by a URL.

The address is specified as a Universal Resource Locator (URL) qualified by time specification and use codes that help in deciding which address to use for a given time and purpose.

The value attribute is constrained to be a uniform resource locator specified according to IETF RFCs 1738 and 2806 when used in this datatype.

Note: The intent of this datatype is to be a locator, not an identifier; this datatype is used to refer to a locatable resource using a URL, and knowing the URL allows one to locate the object. However some use cases have arisen where a URI is used to refer to a locatable resource. Though this datatype allows for URIs to be used, the resource identified SHOULD always be locatable. A common use of locatable URIs is to refer to SOAP attachments.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:TEL
```

XML Source (w/o annotations (4))

```
<xs:complexType name="TEL">
  <xs:complexContent>
    <xs:extension base="ANY">
      <xs:sequence/>
      <xs:attribute name="value" type="xs:anyURI" use="required"/>
      <xs:attribute name="use" type="set_TelecommunicationAddressUse" use="optional"/>
      <xs:attribute name="capabilities" type="set_TelecommunicationCapability" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 3/3)

- value

  Type: xs:anyURI, predefined
  Use: required

A uniform resource identifier specified according to IETF RFC 2396.
The URI specifies the protocol and the contact point defined by that protocol for the resource.
Examples: Notable uses of the telecommunication address datatype are for telephone and telefax numbers, e-mail addresses, Hypertext references, FTP references, etc.

use

Type: dt:set_TelecommunicationAddressUse [143]
Use: optional

One or more codes advising system or user which telecommunication address in a set of like addresses to select for a given telecommunication need.
The telecommunication use code is not a complete classification for equipment types or locations. Its main purpose is to suggest or discourage the use of a particular telecommunication address. There are no easily defined rules that govern the selection of a telecommunication address. Conformance statements may clarify what rules may apply or how additional rules are applied.
If populated, the values contained in this attribute SHALL be taken from the HL7 TelecommunicationAddressUse code system

Attribute Value

```
list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" | "MC" | "PG")
```

capabilities

Type: dt:set_TelecommunicationCapability [144]
Use: optional

One or more codes advising a system or user what telecommunication capabilities are known to be associated with the telecommunication address.
If populated, the values contained in this attribute SHALL be taken from the HL7 TelecommunicationCapability code system

Attribute Value

```
list of ("voice" | "fax" | "data" | "tty" | "sms")
```

complexType "dt:TS"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = xs:string
/>
```

Annotation

A quantity specifying a point on the axis of natural time. A point in time is most often represented as a calendar expression.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:QTY [121] (extension)
          └ dt:TS
```

XML Source (w/o annotations (2))

```
<xs:complexType name="TS">
  <xs:complexContent>
    <xs:extension base="QTY">
      <xs:sequence/>
      <xs:attribute name="value" type="TimeStamp" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: dt:TimeStamp [146]
Use: required

The value of the TS. value is a string with the format "YYYY[MM[DD[HH[MM[SS[.U[U[U[U]]]]]]]]][+|-ZZzz]" that conforms to the constrained ISO 8601 defined in ISO 8824 (ASN.1) under clause 32 (generalized time). The format should be used to the degree of precision that is appropriate.

Attribute Value

xs:string

Pattern: [1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))(((0[1-9])|([1-2][0-9])|(3[0-1]))((([0-1][0-9])|(2[0-3]))(([0-5][0-9])(([0-5][0-9])(\.[0-9]{1,4})?)?)?)?)?([+\-](0[0-9]|1[0-3])([0-5][0-9]))?

complexType "dt:XP"

Namespace: urn:hl7-org:cdsdt:r2
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = xs:string
/>
```

Annotation

A part of a name or address. Each part is a character string.

XML Source (w/o annotations (2))

```
<xs:complexType name="XP">
  <xs:sequence/>
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: xs:string, predefined
Use: required

The actual string value of the part.

simpleType "dt:AddressPartType"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

- "AL" – Address Line: An address line is for either an additional locator, a delivery address or a street address. An address generally has only a delivery address line or a street address line, but not both.
- "ADL" – Additional Locator : This can be a unit designator, such as apartment number, suite number, or floor. There may be several unit designators in an address (e.g., "3rd floor, Appt. 342"). This can also be a designator pointing away from the location, rather than specifying a smaller location within some larger one (e.g., Dutch "t.o." means "opposite to" for house boats located across the street facing houses)
- "UNID" – Unit Identifier : The number or name of a specific unit contained within a building or complex, as assigned by that building or complex
- "UNIT" – Unit Designator: Indicates the type of specific unit contained within a building or complex. E.g. Apartment, Floor
- "DAL" – Delivery Address Line: A delivery address line is frequently used instead of breaking out delivery mode, delivery installation, etc. An address generally has only a delivery address line or a street address line, but not both.
- "DINST" – Delivery Installation Type: Indicates the type of delivery installation (the facility to which the mail will be delivered prior to final shipping via the delivery mode.) Example: post office, letter carrier depot, community mail center, station, etc.
- "DINSTA" – Delivery Installation Area: The location of the delivery installation, usually a town or city, and is only required if the area is different from the municipality. Area to which mail delivery service is provided from any postal facility or service such as an individual letter carrier, rural route, or postal route.
- "DINSTQ" – Delivery Installation Qualifier: A number, letter or name identifying a delivery installation. E.g., for Station A, the delivery installation qualifier would be 'A'.
- "DMOD" – Delivery Mode: Indicates the type of service offered, method of delivery. For example: post office box, rural route, general delivery, etc.
- "DMODID" – Delivery Mode Identifier: Represents the routing information such as a letter carrier route number. It is the identifying number of the designator (the box number or rural route number).
- "SAL" – Street Address Line: A street address line is frequently used instead of breaking out build number, street name, street type, etc. An address generally has only a delivery address line or a street address line, but not both.
- "BNR" – Building Number: The number of a building, house or lot alongside the street. Also known as "primary street number". This does not number the street but rather the building.
- "BNN" – Building Number Numeric: The numeric portion of a building number
- "BNS" – Building Number Suffix: Any alphabetic character, fraction or other text that may appear after the numeric portion of a building number
- "STR" – Street Name: The name of the street, including the type
- "STB" – Street Name Base: The base name of a roadway or artery recognized by a municipality (excluding street type and direction)
- "STTYP" – Street Type: The designation given to the street. (e.g. Street, Avenue, Crescent, etc.)
- "DIR" – Direction (e.g., N, S, W, E)
- "INT" – Intersection: An intersection denotes that the actual address is located at or close to the intersection of two or more streets

"CAR" – Care Of: The name of the party who will take receipt at the specified address, and will take on responsibility for ensuring delivery to the target recipient

"CEN" – Census Tract: A geographic sub-unit delineated for demographic purposes.

"CNT" – Country

"CPA" – County or Parish: A sub-unit of a state or province. (49 of the United States of America use the term "county;" Louisiana uses the term "parish".)

"CTY" – Municipality: The name of the city, town, village, or other community or delivery center

"DEL" – Delimiter: Delimiters are printed without framing white space. If no value component is provided, the delimiter appears as a line break.

"POB" – Post Box: A numbered box located in a post station.

"PRE" – Precinct: A subsection of a municipality

"STA" – State or Province: A sub-unit of a country with limited sovereignty in a federally organized country.

"ZIP" – Postal Code: A postal code designating a region defined by the postal service.

"DPID" – Delivery Point Identifier : A value that uniquely identifies the postal address.

Annotation

Specifies whether an address part names the street, city, country, postal code, post box, etc. If the type is NULL the address part is unclassified.

CodeSystem "AddressPartType", OID: 2.16.840.1.113883.5.16, Owner: HL7

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **dt:AddressPartType**

Derivation: restriction of xs:string

Facets: **enumeration:**

"AL" – Address Line: An address line is for either an additional locator, a delivery address or a street address. An address generally has only a delivery address line or a street address line, but not both.

"ADL" – Additional Locator : This can be a unit designator, such as apartment number, suite number, or floor. There may be several unit designators in an address (e.g., "3rd floor, Appt. 342"). This can also be a designator pointing away from the location, rather than specifying a smaller location within some larger one (e.g., Dutch "t.o." means "opposite to" for house boats located across the street facing houses)

"UNID" – Unit Identifier : The number or name of a specific unit contained within a building or complex, as assigned by that building or complex

"UNIT" – Unit Designator: Indicates the type of specific unit contained within a building or complex. E.g. Apartment, Floor

"DAL" – Delivery Address Line: A delivery address line is frequently used instead of breaking out delivery mode, delivery installation, etc. An address generally has only a delivery address line or a street address line, but not both.

"DINST" – Delivery Installation Type: Indicates the type of delivery installation (the facility to which the mail will be delivered prior to final shipping via the delivery mode.) Example: post office, letter carrier depot, community mail center, station, etc.

"DINSTA" – Delivery Installation Area: The location of the delivery installation, usually a town or city, and is only required if the area is different from the municipality. Area to which mail delivery service is provided from any postal facility or service such as an individual letter carrier, rural route, or postal route.

"DINSTQ" – Delivery Installation Qualifier: A number, letter or name identifying a delivery installation. E.g., for Station A, the delivery installation qualifier would be 'A'.

"DMOD" – Delivery Mode: Indicates the type of service offered, method of delivery. For example: post office box, rural route, general delivery, etc.

"DMODID" – Delivery Mode Identifier: Represents the routing information such as a letter carrier route number. It is the identifying number of the designator (the box number or rural route number).

"SAL" – Street Address Line: A street address line is frequently used instead of breaking out build number, street name, street type, etc. An address generally has only a delivery address line or a street address line, but not both.

"BNR" – Building Number: The number of a building, house or lot alongside the street. Also known as "primary street number". This does not number the street but rather the building.

"BNN" – Building Number Numeric: The numeric portion of a building number

"BNS" – Building Number Suffix: Any alphabetic character, fraction or other text that may appear after the numeric

portion of a building number

"STR" – Street Name: The name of the street, including the type

"STB" – Street Name Base: The base name of a roadway or artery recognized by a municipality (excluding street type and direction)

"STTYP" – Street Type: The designation given to the street. (e.g. Street, Avenue, Crescent, etc.)

"DIR" – Direction (e.g., N, S, W, E)

"INT" – Intersection: An intersection denotes that the actual address is located at or close to the intersection of two or more streets

"CAR" – Care Of: The name of the party who will take receipt at the specified address, and will take on responsibility for ensuring delivery to the target recipient

"CEN" – Census Tract: A geographic sub-unit delineated for demographic purposes.

"CNT" – Country

"CPA" – County or Parish: A sub-unit of a state or province. (49 of the United States of America use the term "county;" Louisiana uses the term "parish".)

"CTY" – Municipality: The name of the city, town, village, or other community or delivery center

"DEL" – Delimiter: Delimiters are printed without framing white space. If no value component is provided, the delimiter appears as a line break.

"POB" – Post Box: A numbered box located in a post station.

"PRE" – Precinct: A subsection of a municipality

"STA" – State or Province: A sub-unit of a country with limited sovereignty in a federally organized country.

"ZIP" – Postal Code: A postal code designating a region defined by the postal service.

"DPID" – Delivery Point Identifier : A value that uniquely identifies the postal address.

XML Source (w/o annotations (31))

```
<xs:simpleType name="AddressPartType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AL"/>
    <xs:enumeration value="ADL"/>
    <xs:enumeration value="UNID"/>
    <xs:enumeration value="UNIT"/>
    <xs:enumeration value="DAL"/>
    <xs:enumeration value="DINST"/>
    <xs:enumeration value="DINSTA"/>
    <xs:enumeration value="DINSTQ"/>
    <xs:enumeration value="DMOD"/>
    <xs:enumeration value="DMODID"/>
    <xs:enumeration value="SAL"/>
    <xs:enumeration value="BNR"/>
    <xs:enumeration value="BNN"/>
    <xs:enumeration value="BNS"/>
    <xs:enumeration value="STR"/>
    <xs:enumeration value="STB"/>
    <xs:enumeration value="STTYP"/>
    <xs:enumeration value="DIR"/>
    <xs:enumeration value="INT"/>
    <xs:enumeration value="CAR"/>
    <xs:enumeration value="CEN"/>
```

```
      <xs:enumeration value="CNT"/>
      <xs:enumeration value="CPA"/>
      <xs:enumeration value="CTY"/>
      <xs:enumeration value="DEL"/>
      <xs:enumeration value="POB"/>
      <xs:enumeration value="PRE"/>
      <xs:enumeration value="STA"/>
      <xs:enumeration value="ZIP"/>
      <xs:enumeration value="DPID"/>
    </xs:restriction>
  </xs:simpleType>
```

simpleType "dt:CalendarCycle"

Namespace: `urn:hl7-org:cdsdt:r2`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration:
"CY" – year
"MY" – month of the year
"CM" – month (continuous)
"CW" – week (continuous)
"WM" – week of the month
"WY" – week of the year
"DM" – day of the month
"CD" – day (continuous)
"DY" – day of the year
"DW" – day of the week (begins with monday)
"HD" – hour of the day
"CH" – hour (continuous)
"NH" – minute of the hour
"CN" – minute (continuous)
"SN" – second of the minute
"CS" – second (continuous)

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:CalendarCycle`**

Derivation: restriction of `xs:string`
Facets: **enumeration:**
"CY" – year
"MY" – month of the year
"CM" – month (continuous)
"CW" – week (continuous)
"WM" – week of the month
"WY" – week of the year
"DM" – day of the month
"CD" – day (continuous)
"DY" – day of the year
"DW" – day of the week (begins with monday)
"HD" – hour of the day
"CH" – hour (continuous)
"NH" – minute of the hour

"CN" – minute (continuous)
"SN" – second of the minute
"CS" – second (continuous)

XML Source (w/o annotations (16))

```
<xs:simpleType name="CalendarCycle">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CY"/>
    <xs:enumeration value="MY"/>
    <xs:enumeration value="CM"/>
    <xs:enumeration value="CW"/>
    <xs:enumeration value="WM"/>
    <xs:enumeration value="WY"/>
    <xs:enumeration value="DM"/>
    <xs:enumeration value="CD"/>
    <xs:enumeration value="DY"/>
    <xs:enumeration value="DW"/>
    <xs:enumeration value="HD"/>
    <xs:enumeration value="CH"/>
    <xs:enumeration value="NH"/>
    <xs:enumeration value="CN"/>
    <xs:enumeration value="SN"/>
    <xs:enumeration value="CS"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:Code"

Namespace: `urn:hl7-org:cdsdt:r2`

Simple Content Model

`xs:string`

Annotation

A code representing the string data. For example, the string data may be a user-message out of a message-catalog where the code represents the identifier of the message in the message catalog.

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **dt:Code**

Derivation: restriction of `xs:string`

XML Source (w/o annotations (1))

```
<xs:simpleType name="Code">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

simpleType "dt:Compression"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:
"DF" – Deflate : The deflate compressed data format as specified in IETF RFC 1951.

"GZ" – GZIP : A compressed data format that is compatible with the widely used GZIP utility as specified in IETF RFC 1952(uses the deflate algorithm).

"ZL" – ZLIB : A compressed data format that also uses the deflate algorithm. Specified as IETF RFC 1950.

"Z" – Compress : Original UNIX compress algorithm and file format using the LZC algorithm (a variant of LZW). Patent encumbered and less efficient than deflate.

"BZ" – BZIP : bzip-2 compression format. See [http://www.bzip.org/] for more information.

"Z7" – Z7 : 7z compression file format. See [http://www.7-zip.org/7z.html] for more information.

Annotation

The compression algorithm, specified in the HL7 CompressionAlgorithm code system.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **dt:Compression**

Derivation: restriction of xs:string

Facets: **enumeration:**
"DF" – Deflate : The deflate compressed data format as specified in IETF RFC 1951.

"GZ" – GZIP : A compressed data format that is compatible with the widely used GZIP utility as specified in IETF RFC 1952(uses the deflate algorithm).

"ZL" – ZLIB : A compressed data format that also uses the deflate algorithm. Specified as IETF RFC 1950.

"Z" – Compress : Original UNIX compress algorithm and file format using the LZC algorithm (a variant of LZW). Patent encumbered and less efficient than deflate.

"BZ" – BZIP : bzip-2 compression format. See [http://www.bzip.org/] for more information.

"Z7" – Z7 : 7z compression file format. See [http://www.7-zip.org/7z.html] for more information.

XML Source (w/o annotations (7))

```
<xs:simpleType name="Compression">
  <xs:restriction base="xs:string">
    <xs:enumeration value="DF"/>
    <xs:enumeration value="GZ"/>
```

```
    <xs:enumeration value="ZL"/>
    <xs:enumeration value="Z"/>
    <xs:enumeration value="BZ"/>
    <xs:enumeration value="ZZ"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:Decimal"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

xs:double

Annotation

A number that is not restricted to an integer, and may contain fractional values between two integers.

Type Definition Detail

Type Derivation Tree

xs:double (restriction)

└ **dt:Decimal**

Derivation: restriction of xs:double

XML Source (w/o annotations (1))

```
<xs:simpleType name="Decimal">
  <xs:restriction base="xs:double"/>
</xs:simpleType>
```

simpleType "dt:EntityNamePartQualifier"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"LS" – Legal Status : For organizations a suffix indicating the legal status, e.g., "Inc.", "Co.", "AG", "GmbH", "B.V." "S.A.", "Ltd." Etc.

"AC" – Academic : Indicates that a prefix like "Dr." or a suffix like "M.D." or "Ph.D." is an academic title

"NB" – Nobility : In Europe and Asia, there are still people with nobility titles (aristocrats). German "von" is generally a nobility title, not a mere voorvoegsel. Others are "Earl of" or "His Majesty King of..." etc. Rarely used nowadays, but some systems do keep track of this

"PR" – Professional : Primarily in the British Imperial culture people tend to have an abbreviation of their professional organization as part of their credential suffices

"HON" – Honorific : A honorific such as "The Right Honourable" or "Weledelgeleerde Heer".

"BR" – Birth : A name that a person was given at birth or established as a consequence of adoption.

Note: this is not used for temporary names assigned at birth such as "Baby of Smith" - which is just a name with a use code of "TEMP".

"AD" – Acquired : A name part a person acquired.

The name part may be acquired by adoption, or the person may have chosen to use the name part for some other reason.

Note: this differs from an Other/Psuedonym/Alias in that an acquired name part is acquired on a formal basis rather than an informal one (e.g. registered as part of the official name)

"SP" – Spouse : The name assumed from the partner in a marital relationship. Usually the spouse's family name. Note that no inference about gender can be made from the existence of spouse names

"MID" – Middle Name : Indicates that the name part is a middle name.
Usage Notes:
In general, the english 'middle name' concept is all of the given names after the first. This qualifier may be used to explicitly indicate which given names are considered to be middle names.
The middle name qualifier may also be used with family names. This is a Scandinavian use case, matching the concept of "mellomnavn" / "mellannamn". Note that there are specific rules that indicate what names may be taken as a mellannamn in different Scandinavian countries

"CL" – Callme : Callme is used to indicate which of the various name parts is used when interacting with the person

"IN" – Initial : Indicates that a name part is just an initial. Initials do not imply a trailing period since this would not work with non-Latin scripts. Initials may consist of more than one letter, e.g., "Ph." could stand for "Philippe" or "Th." for "Thomas"

"PFX" – Prefix : A prefix has a strong association to the immediately following name part. A prefix has no implicit trailing white space (it has implicit leading white space though).

"SFX" – Suffix : A suffix has a strong association to the immediately preceding name part. A suffix has no implicit leading white space (it has implicit trailing white space though).

Annotation

The qualifier is a set of codes each of which specifies a certain subcategory of the name part in addition to the main name part type. For example, a given name may be flagged as a nickname, a family name may be a pseudonym or a name of public records.

CodeSystem "EntityNamePartTypeQualifierR2", OID: 2.16.840.1.113883.5.1122, Owner: HL7

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:EntityNamePartQualifier`**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

"LS" – Legal Status : For organizations a suffix indicating the legal status, e.g., "Inc.", "Co.", "AG", "GmbH", "B.V." "S.A.", "Ltd." Etc.

"AC" – Academic : Indicates that a prefix like "Dr." or a suffix like "M.D." or "Ph.D." is an academic title

"NB" – Nobility : In Europe and Asia, there are still people with nobility titles (aristocrats). German "von" is generally a nobility title, not a mere voorvoegsel. Others are "Earl of" or "His Majesty King of..." etc. Rarely used nowadays, but some systems do keep track of this

"PR" – Professional : Primarily in the British Imperial culture people tend to have an abbreviation of their professional organization as part of their credential suffices

"HON" – Honorific : A honorific such as "The Right Honourable" or "Weledelgeleerde Heer".

"BR" – Birth : A name that a person was given at birth or established as a consequence of adoption.

Note: this is not used for temporary names assigned at birth such as "Baby of Smith" - which is just a name with a use code of "TEMP".

"AD" – Acquired : A name part a person acquired.

The name part may be acquired by adoption, or the person may have chosen to use the name part for some other reason.

Note: this differs from an Other/Psuedonym/Alias in that an acquired name part is acquired on a formal basis rather than an informal one (e.g. registered as part of the official name)

"SP" – Spouse : The name assumed from the partner in a marital relationship. Usually the spouse's family name. Note that no inference about gender can be made from the existence of spouse names

"MID" – Middle Name : Indicates that the name part is a middle name.
Usage Notes:
In general, the english 'middle name' concept is all of the given names after the first. This qualifier may be used to explicitly indicate which given names are considered to be middle names.
The middle name qualifier may also be used with family names. This is a Scandinavian use case, matching the

concept of "mellomnavn" / "mellannamn". Note that there are specific rules that indicate what names may be taken as a mellannamn in different Scandinavian countries

"CL" – Callme : Callme is used to indicate which of the various name parts is used when interacting with the person

"IN" – Initial : Indicates that a name part is just an initial. Initials do not imply a trailing period since this would not work with non-Latin scripts. Initials may consist of more than one letter, e.g., "Ph." could stand for "Philippe" or "Th." for "Thomas"

"PFX" – Prefix : A prefix has a strong association to the immediately following name part. A prefix has no implicit trailing white space (it has implicit leading white space though).

"SFX" – Suffix : A suffix has a strong association to the immediately preceding name part. A suffix has no implicit leading white space (it has implicit trailing white space though).

XML Source (w/o annotations (14))

```
<xs:simpleType name="EntityNamePartQualifier">
  <xs:restriction base="xs:string">
    <xs:enumeration value="LS"/>
    <xs:enumeration value="AC"/>
    <xs:enumeration value="NB"/>
    <xs:enumeration value="PR"/>
    <xs:enumeration value="HON"/>
    <xs:enumeration value="BR"/>
    <xs:enumeration value="AD"/>
    <xs:enumeration value="SP"/>
    <xs:enumeration value="MID"/>
    <xs:enumeration value="CL"/>
    <xs:enumeration value="IN"/>
    <xs:enumeration value="PFX"/>
    <xs:enumeration value="SFX"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:EntityNamePartType"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"FAM" – Family : Family name, this is the name that links to the genealogy. In some cultures (e.g. Eritrea) the family name of a son is the first name of his father

"GIV" – Given: Given name.
Note: don't call it "first name" since this given names do not always come first

"TITLE" – Title : Part of the name that is acquired as a title due to academic, legal, employment or nobility status etc.
Note: Title name parts include name parts that come after the name such as qualifications

"DEL" – Delimiter : A delimiter has no meaning other than being literally printed in this name representation. A delimiter has no implicit leading and trailing white space

Annotation

Indicates whether the name part is a given name, family name, prefix, suffix, etc.

CodeSystem "EntityNamePartTypeR2", OID: 2.16.840.1.113883.5.1122, Owner: HL7

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:EntityNamePartType`**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

- `"FAM"` – Family : Family name, this is the name that links to the genealogy. In some cultures (e.g. Eritrea) the family name of a son is the first name of his father
- `"GIV"` – Given: Given name.
  Note: don't call it "first name" since this given names do not always come first
- `"TITLE"` – Title : Part of the name that is acquired as a title due to academic, legal, employment or nobility status etc.
  Note: Title name parts include name parts that come after the name such as qualifications
- `"DEL"` – Delimiter : A delimiter has no meaning other than being literally printed in this name representation. A delimiter has no implicit leading and trailing white space

XML Source (w/o annotations (5))

```
<xs:simpleType name="EntityNamePartType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="FAM"/>
    <xs:enumeration value="GIV"/>
    <xs:enumeration value="TITLE"/>
    <xs:enumeration value="DEL"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:EntityNameUse"

Namespace: `urn:hl7-org:cdsdt:r2`

Simple Content Model

`enumeration of` `xs:string`

Simple Content Restrictions:

Enumeration:

- `"ABC"` – Alphabetic: Alphabetic transcription of name (Japanese: romaji)
- `"IDE"` – Ideographic : Ideographic representation of name (e.g., Japanese kanji, Chinese characters)
- `"SYL"` – Syllabic: Syllabic transcription of name (e.g., Japanese kana, Korean hangul)
- `"C"` – Customary : Known as/conventional/the one you normally use
- `"OR"` – Official Registry Name : the formal name as registered in an official (government) registry, but which name might not be commonly used. May correspond to the concept of legal name
- `"T"` – Temporary : A temporary name. Note that a name valid time can provide more detailed information. This may also be used for temporary names assigned at birth or in emergency situations.
- `"I"` – Indigenous/Tribal: e.g. Chief Red Cloud
- `"P"` – Other/Pseudonym/Alias: A non-official name by which the person is sometimes known. (This may also be used to record informal names such as a nickname)
- `"ANON"` – Anonymous : Anonymous assigned name (used to protect a person's identity for privacy reasons)
- `"A"` – Business Name : A name used in a Professional or Business context .

  Examples: Continuing to use a maiden name in a professional context, or using a stage performing name (some of these names are also pseudonyms)
- `"R"` – Religious : A name assumed as part of a religious vocation. e.g. Sister Mary Francis, Brother John

"OLD" – No Longer in Use : This name is no longer in use (note: Names may also carry valid time ranges . This code is used to cover the situations where it is known that the name is no longer valid, but no particular time range for its use is known)

"DN" – Do Not Use : This name should no longer be used when interacting with the person (i.e . in addition to no longer being used, the name should not be even mentioned when interacting with the person)

Note: applications are not required to compare names labeled "Do Not Use" and other names in order to eliminate name parts that are common between the other name and a name labeled "Do Not Use".

"M" – Maiden Name : A name used prior to marriage.

Note that marriage naming customs vary greatly around the world. This name use is for use by applications that collect and store "maiden" names. Though the concept of maiden name is often gender specific, the use of this term is not gender specific. The use of this term does not imply any particular history for a person's name, nor should the maiden name be determined algorithmically

"PHON" – Phonetic : The name as understood by the data enterer, i.e. a close approximation of a phonetic spelling of the name, not based on a phonetic algorithm.

"SRCH" – Search Type Uses: A name intended for use in searching or matching

Annotation

A set of codes advising a system or user which name in a set of names to select for a given purpose.

CodeSystem "EntityNameUseR2", OID: 2.16.840.1.113883.5.1120, Owner: HL7

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **dt:EntityNameUse**

Derivation: restriction of xs:string

Facets: **enumeration:**

"ABC" – Alphabetic: Alphabetic transcription of name (Japanese: romaji)

"IDE" – Ideographic : Ideographic representation of name (e.g., Japanese kanji, Chinese characters)

"SYL" – Syllabic: Syllabic transcription of name (e.g., Japanese kana, Korean hangul)

"C" – Customary : Known as/conventional/the one you normally use

"OR" – Official Registry Name : the formal name as registered in an official (government) registry, but which name might not be commonly used. May correspond to the concept of legal name

"T" – Temporary : A temporary name. Note that a name valid time can provide more detailed information. This may also be used for temporary names assigned at birth or in emergency situations.

"I" – Indigenous/Tribal: e.g. Chief Red Cloud

"P" – Other/Pseudonym/Alias: A non-official name by which the person is sometimes known. (This may also be used to record informal names such as a nickname)

"ANON" – Anonymous : Anonymous assigned name (used to protect a person's identity for privacy reasons)

"A" – Business Name : A name used in a Professional or Business context .

Examples: Continuing to use a maiden name in a professional context, or using a stage performing name (some of these names are also pseudonyms)

"R" – Religious : A name assumed as part of a religious vocation. e.g. Sister Mary Francis, Brother John

"OLD" – No Longer in Use : This name is no longer in use (note: Names may also carry valid time ranges . This code is used to cover the situations where it is known that the name is no longer valid, but no particular time range for its use is known)

"DN" – Do Not Use : This name should no longer be used when interacting with the person (i.e . in addition to no longer being used, the name should not be even mentioned when interacting with the person)

Note: applications are not required to compare names labeled "Do Not Use" and other names in order to eliminate name parts that are common between the other name and a name labeled "Do Not Use".

"M" – Maiden Name : A name used prior to marriage.

Note that marriage naming customs vary greatly around the world. This name use is for use by applications that collect and store "maiden" names. Though the concept of maiden name is often gender specific, the use of this term is not gender specific. The use of this term does not imply any particular history for a person's name, nor should the maiden name be determined algorithmically

"PHON" – Phonetic : The name as understood by the data enterer, i.e. a close approximation of a phonetic spelling of the name, not based on a phonetic algorithm.

"SRCH" – Search Type Uses: A name intended for use in searching or matching

XML Source (w/o annotations (17))

```
<xs:simpleType name="EntityNameUse">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ABC"/>
    <xs:enumeration value="IDE"/>
    <xs:enumeration value="SYL"/>
    <xs:enumeration value="C"/>
    <xs:enumeration value="OR"/>
    <xs:enumeration value="T"/>
    <xs:enumeration value="I"/>
    <xs:enumeration value="P"/>
    <xs:enumeration value="ANON"/>
    <xs:enumeration value="A"/>
    <xs:enumeration value="R"/>
    <xs:enumeration value="OLD"/>
    <xs:enumeration value="DN"/>
    <xs:enumeration value="M"/>
    <xs:enumeration value="PHON"/>
    <xs:enumeration value="SRCH"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:IntegrityCheckAlgorithm"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "SHA1" – Secure Hash Algorithm - 1 : This algorithm is defined in FIPS PUB 180-1: Secure Hash Standard. As of April 17, 1995
"SHA256" – Secure Hash Algorithm - 256 : This algorithm is defined in FIPS PUB 180-2: Secure Hash Standard

Annotation

The algorithm used to compute the integrityCheck value.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **dt:IntegrityCheckAlgorithm**

Derivation: restriction of xs:string
Facets: **enumeration:** "SHA1" – Secure Hash Algorithm - 1 : This algorithm is defined in FIPS PUB 180-1: Secure Hash Standard. As of April 17, 1995

"SHA256" – Secure Hash Algorithm - 256 : This algorithm is defined in FIPS PUB 180-2: Secure Hash Standard

XML Source (w/o annotations (3))

```
<xs:simpleType name="IntegrityCheckAlgorithm">
  <xs:restriction base="xs:string">
    <xs:enumeration value="SHA1"/>
    <xs:enumeration value="SHA256"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:PostalAddressUse"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"H" – Home address : A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available.

"HP" – Primary Home: The primary home, to reach a person after business hours.

"HV" – Vacation Home: A vacation home, to reach a person while on vacation.

"WP" – Work Place: An office address. First choice for business related contacts during business hours.

"DIR" – Direct: Indicates a work place address or telecommunication address that reaches the individual or organization directly without intermediaries. For phones, often referred to as a 'private line'.

"PUB" – Public: Indicates a work place address or telecommunication address that is a 'standard' address which may reach a reception service, mail-room, or other intermediary prior to the target entity.

"BAD" – Bad Address: A flag indicating that the address is bad, in fact, useless.

"PHYS" – Physical Visit Address: Used primarily to visit an address.

"PST" – Postal Address: Used to send mail.

"TMP" – Temporary Address: A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information.

"ABC" – Alphabetic: Alphabetic transcription of name (Japanese: romaji)

"IDE" – Ideographic: Ideographic representation of name (e.g., Japanese kanji, Chinese characters)

"SYL" – Syllabic: Syllabic transcription of name (e.g., Japanese kana, Korean hangul)

"SRCH" – Search Type Uses: A name intended for use in searching or matching.

"SNDX" – Soundex: An address spelled according to the SoundEx algorithm.

"PHON" – Phonetic: The address as understood by the data enterer, i.e. a close approximation of a phonetic spelling of the address, not based on a phonetic algorithm.

Annotation

A set of codes advising a system or user which address in a set of like addresses to select for a given purpose.

CodeSystem "PostalAddressUse", OID: 2.16.840.1.113883.5.1012, Owner: HL7

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:PostalAddressUse`**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

- `"H"` – Home address : A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available.
- `"HP"` – Primary Home: The primary home, to reach a person after business hours.
- `"HV"` – Vacation Home: A vacation home, to reach a person while on vacation.
- `"WP"` – Work Place: An office address. First choice for business related contacts during business hours.
- `"DIR"` – Direct: Indicates a work place address or telecommunication address that reaches the individual or organization directly without intermediaries. For phones, often referred to as a 'private line'.
- `"PUB"` – Public: Indicates a work place address or telecommunication address that is a 'standard' address which may reach a reception service, mail-room, or other intermediary prior to the target entity.
- `"BAD"` – Bad Address: A flag indicating that the address is bad, in fact, useless.
- `"PHYS"` – Physical Visit Address: Used primarily to visit an address.
- `"PST"` – Postal Address: Used to send mail.
- `"TMP"` – Temporary Address: A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information.
- `"ABC"` – Alphabetic: Alphabetic transcription of name (Japanese: romaji)
- `"IDE"` – Ideographic: Ideographic representation of name (e.g., Japanese kanji, Chinese characters)
- `"SYL"` – Syllabic: Syllabic transcription of name (e.g., Japanese kana, Korean hangul)
- `"SRCH"` – Search Type Uses: A name intended for use in searching or matching.
- `"SNDX"` – Soundex: An address spelled according to the SoundEx algorithm.
- `"PHON"` – Phonetic: The address as understood by the data enterer, i.e. a close approximation of a phonetic spelling of the address, not based on a phonetic algorithm.

XML Source (w/o annotations (17))

```
<xs:simpleType name="PostalAddressUse">
  <xs:restriction base="xs:string">
    <xs:enumeration value="H"/>
    <xs:enumeration value="HP"/>
    <xs:enumeration value="HV"/>
    <xs:enumeration value="WP"/>
    <xs:enumeration value="DIR"/>
    <xs:enumeration value="PUB"/>
    <xs:enumeration value="BAD"/>
    <xs:enumeration value="PHYS"/>
    <xs:enumeration value="PST"/>
    <xs:enumeration value="TMP"/>
    <xs:enumeration value="ABC"/>
    <xs:enumeration value="IDE"/>
    <xs:enumeration value="SYL"/>
    <xs:enumeration value="SRCH"/>
```

```
    <xs:enumeration value="SNDX"/>
    <xs:enumeration value="PHON"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:set_EntityNamePartQualifier"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

list of ("LS" | "AC" | "NB" | "PR" | "HON" | "BR" | "AD" | "SP" | "MID" | "CL" | "IN" | "PFX" | "SFX")

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ dt:EntityNamePartQualifier [134] (list)

└ **dt:set_EntityNamePartQualifier**

Derivation: list of dt:EntityNamePartQualifier

XML Source

```
<xs:simpleType name="set_EntityNamePartQualifier">
  <xs:list itemType="dt:EntityNamePartQualifier"/>
</xs:simpleType>
```

simpleType "dt:set_EntityNameUse"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

list of ("ABC" | "IDE" | "SYL" | "C" | "OR" | "T" | "I" | "P" | "ANON" | "A" | "R" | "OLD" | "DN" | "M" | "PHON" | "SRCH")

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ dt:EntityNameUse [137] (list)

└ **dt:set_EntityNameUse**

Derivation: list of dt:EntityNameUse

XML Source

```
<xs:simpleType name="set_EntityNameUse">
```

```
  <xs:list itemType="dt:EntityNameUse"/>
</xs:simpleType>
```

simpleType "dt:set_PostalAddressUse"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "PHYS" | "PST" | "TMP" | "ABC" | "IDE" | "SYL" | "SRCH" | "SNDX" | "PHON")

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ dt:PostalAddressUse [140] (list)

└ **dt:set_PostalAddressUse**

Derivation: list of dt:PostalAddressUse

XML Source

```
<xs:simpleType name="set_PostalAddressUse">
  <xs:list itemType="dt:PostalAddressUse"/>
</xs:simpleType>
```

simpleType "dt:set_TelecommunicationAddressUse"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" | "MC" | "PG")

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ dt:TelecommunicationAddressUse [144] (list)

└ **dt:set_TelecommunicationAddressUse**

Derivation: list of dt:TelecommunicationAddressUse

XML Source

```
<xs:simpleType name="set_TelecommunicationAddressUse">
  <xs:list itemType="dt:TelecommunicationAddressUse"/>
</xs:simpleType>
```

simpleType "dt:set_TelecommunicationCapability"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

list of ("voice" | "fax" | "data" | "tty" | "sms")

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ dt:TelecommunicationCapability [146] (list)

└ **dt:set_TelecommunicationCapability**

Derivation: list of dt:TelecommunicationCapability

XML Source

```
<xs:simpleType name="set_TelecommunicationCapability">
  <xs:list itemType="dt:TelecommunicationCapability"/>
</xs:simpleType>
```

simpleType "dt:TelecommunicationAddressUse"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"H" – Home address : A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available

"HP" – Primary Home: The primary home, to reach a person after business hours.

"HV" – Vacation Home: vacation home, to reach a person while on vacation.

"WP" – Work Place: An office address. First choice for business related contacts during business hours.

"DIR" – Direct: Indicates a work place address or telecommunication address that reaches the individual or organization directly without intermediaries. For phones, often referred to as a 'private line'.

"PUB" – Public: Indicates a work place address or telecommunication address that is a 'standard' address which may reach a reception service, mail-room, or other intermediary prior to the target entity.

"BAD" – Bad Address: A flag indicating that the address is bad, in fact, useless.

"TMP" – Temporary Address: A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information.

"AS" – Answering Service: An automated answering machine used for less urgent cases and if the main purpose of contact is to leave a message or access an automated announcement.

"EC" – Emergency Contact: A contact specifically designated to be used for emergencies. This is the first choice in emergencies, independent of any other use codes.

"MC" – Mobile Contact: A telecommunication device that moves and stays with its owner. May have characteristics of all other use codes, suitable for urgent matters, not the first choice for routine business.

"PG" – Pager: A paging device suitable to solicit a callback or to leave a very short message.

Annotation

One or more codes advising a system or user which telecommunication address in a set of like addresses to select for a given telecommunication need.

CodeSystem "TelecommunicationAddressUse", OID: 2.16.840.1.113883.5.1011, Owner: HL7

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:TelecommunicationAddressUse`**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

- `"H"` – Home address : A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available
- `"HP"` – Primary Home: The primary home, to reach a person after business hours.
- `"HV"` – Vacation Home: vacation home, to reach a person while on vacation.
- `"WP"` – Work Place: An office address. First choice for business related contacts during business hours.
- `"DIR"` – Direct: Indicates a work place address or telecommunication address that reaches the individual or organization directly without intermediaries. For phones, often referred to as a 'private line'.
- `"PUB"` – Public: Indicates a work place address or telecommunication address that is a 'standard' address which may reach a reception service, mail-room, or other intermediary prior to the target entity.
- `"BAD"` – Bad Address: A flag indicating that the address is bad, in fact, useless.
- `"TMP"` – Temporary Address: A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information.
- `"AS"` – Answering Service: An automated answering machine used for less urgent cases and if the main purpose of contact is to leave a message or access an automated announcement.
- `"EC"` – Emergency Contact: A contact specifically designated to be used for emergencies. This is the first choice in emergencies, independent of any other use codes.
- `"MC"` – Mobile Contact: A telecommunication device that moves and stays with its owner. May have characteristics of all other use codes, suitable for urgent matters, not the first choice for routine business.
- `"PG"` – Pager: A paging device suitable to solicit a callback or to leave a very short message.

XML Source (w/o annotations (13))

```
<xs:simpleType name="TelecommunicationAddressUse">
  <xs:restriction base="xs:string">
    <xs:enumeration value="H"/>
    <xs:enumeration value="HP"/>
    <xs:enumeration value="HV"/>
    <xs:enumeration value="WP"/>
    <xs:enumeration value="DIR"/>
    <xs:enumeration value="PUB"/>
    <xs:enumeration value="BAD"/>
    <xs:enumeration value="TMP"/>
    <xs:enumeration value="AS"/>
    <xs:enumeration value="EC"/>
    <xs:enumeration value="MC"/>
    <xs:enumeration value="PG"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:TelecommunicationCapability"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:
"voice" – Voice : This device can receive voice calls (i.e. talking to another person, or a recording device, or a voice activated computer)
"fax" – Fax : This device can receive faxes.
"data" – Data : This device can receive data calls (i.e. modem)
"tty" – Text : This device is a text telephone.
"sms" – SMS : This device can receive SMS messages

Annotation

One or more codes advising a system or user what telecommunication capabilities are known to be associated with the telecommunication address.

CodeSystem "TelecommunicationCapabilities", OID: 2.16.840.1.113883.5.1118, Owner: HL7

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **dt:TelecommunicationCapability**

Derivation: restriction of xs:string
Facets: **enumeration:**
"voice" – Voice : This device can receive voice calls (i.e. talking to another person, or a recording device, or a voice activated computer)
"fax" – Fax : This device can receive faxes.
"data" – Data : This device can receive data calls (i.e. modem)
"tty" – Text : This device is a text telephone.
"sms" – SMS : This device can receive SMS messages

XML Source (w/o annotations (6))

```
<xs:simpleType name="TelecommunicationCapability">
  <xs:restriction base="xs:string">
    <xs:enumeration value="voice"/>
    <xs:enumeration value="fax"/>
    <xs:enumeration value="data"/>
    <xs:enumeration value="tty"/>
    <xs:enumeration value="sms"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:TimeStamp"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

xs:string

Simple Content Restrictions:

Pattern: [1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))(((0[1-9])|([1-2][0-9])|(3[0-1]))((([0-1][0-9])|(2[0-3]))(([0-5][0-9])(([0-5][0-9])(\.[0-9]{1,4})?)?)?)?)?)?([+\-](0[0-9]|1[0-3])([0-5][0-9]))?

Annotation

Represents a timestamp such as 20101127235417.123+0930

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **dt:TimeStamp**

Derivation: restriction of xs:string
Facets: **pattern:** [1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))(((0[1-9])|([1-2][0-9])|(3[0-1]))((([0-1][0-9])|(2[0-3]))(([0-5][0-9])(([0-5][0-9])(\.[0-9]{1,4})?)?)?)?)?)?([+\-](0[0-9]|1[0-3])([0-5][0-9]))?

XML Source (w/o annotations (1))

```
<xs:simpleType name="TimeStamp">
  <xs:restriction base="xs:string">
    <xs:pattern value="[1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))(((0[1-9])|([1-2][0-9])|(3[0-1]))((([0-1][0-9])|(2[0-3]))(([0-5][0-9])(([0-5][0-9])(\.[0-9]{1,4})?)?)?)?)?)?([+\-](0[0-9]|1[0-3])([0-5][0-9]))?"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "dt:Uid"

Namespace: urn:hl7-org:cdsdt:r2

Simple Content Model

xs:string

Annotation

A unique identifier string is a character string which identifies an object in a globally unique and timeless manner. The allowable formats and values and procedures of this data type are strictly controlled by HL7. At this time, user-assigned identifiers SHALL only be certain character representations of ISO Object Identifiers (OID) and DCE Universally Unique Identifiers (UUID). In addition, HL7 reserves the right to assign other forms of UIDs (RUID), such as mnemonic identifiers for code systems.
The sole purpose of UID is to be a globally and timelessly unique identifier. The form of UID, whether it is an OID, a UUID or a RUID, is entirely irrelevant. As far as HL7 is concerned, the only thing one can do with a UID is denote to the object for which it stands. Comparison of UIDs is literal, i.e. if two UIDs are literally identical, they are assumed to denote to the same object. If two UIDs are not literally identical they may not denote to the same object. Note that this comparison is case sensitive; (OID)s do not have letters subject to case, (UUID)s are fixed to uppercase, and (RUID)s have a fixed case.
protected type UniqueIdentifierString alias UID specializes ST.SIMPLE;
No difference in semantics is recognized between the different allowed forms of UID. The different forms are not distinguished by a component within or aside from the identifier string itself.
Even though this specification recognizes no semantic difference between the different forms of the unique identifier forms,

there are differences of how these identifiers are built and managed, which is the sole reason to define subtypes of UID for each of the variants.

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:Uid`**

Derivation: restriction of `xs:string`

XML Source (w/o annotations (1))

```
<xs:simpleType name="Uid">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

simpleType "dt:Uri"

Namespace: `urn:hl7-org:cdsdt:r2`

Simple Content Model

`xs:string`

Annotation

Universal Resource Identifier

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`dt:Uri`**

Derivation: restriction of `xs:string`

XML Source (w/o annotations (1))

```
<xs:simpleType name="Uri">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

Namespace "urn:hl7-org:knowledgeartifact:r1"

Targeting Schemas (48):

action.xsd, actor.xsd, artifactlifecycleeventtype.xsd, artifactlifecycleeventtypecore.xsd, artifactlifecycleeventtypeext.xsd, artifactstatustype.xsd, artifactstatustypecore.xsd, artifactstatustypeext.xsd, artifacttype.xsd, artifacttypecore.xsd, artifacttypeext.xsd, base.xsd, behavior.xsd, behaviortype.xsd, behaviortypecore.xsd, behaviortypeext.xsd, cardinalitytype.xsd, catalogitem.xsd, clinicalexpression.xsd, condition.xsd, conditionroletype.xsd, conditionroletypecore.xsd, conditionroletypeext.xsd, contributortype.xsd, contributortypecore.xsd, contributortypeext.xsd, coveragetype.xsd, coveragetypecore.xsd, coveragetypeext.xsd, dataeventtype.xsd, dataeventtypecore.xsd, dataeventtypeext.xsd, eventtype.xsd, eventtypecore.xsd, eventtypeext.xsd, expression.xsd, knowledgedocument.xsd, literalexpression.xsd, metadata.xsd, rangeconstrainttype.xsd, rangeconstrainttypecore.xsd, rangeconstrainttypeext.xsd, resourcerelationshiptype.xsd, resourcerelationshiptypecore.xsd, resourcerelationshiptypeext.xsd, valuetype.xsd, valuetypecore.xsd, valuetypeext.xsd

Targeting Components:

elements (2 global + 212 local), complexTypes (189), simpleTypes (48)

Global Element Summary		Page
itemDefinitions	A collection of item definitions **Type:** anonymous complexType	*156*
knowledgeDocument	knowledgeDocument represents a serialized Clinical Decision Support (CDS) knowledge artifact. **Type:** KnowledgeDocument [262]	*157*

Complex Type Summary		Page
Abs	The Abs operator returns the absolute value of its argument.	*158*
ActionBase	Actions are the output of the CDS system and represent the tasks that must be carried out by a human or a computer system.	*158*
ActionGroup	This type of action is used to organize a group of related actions into one container.	*161*
Actor	An actor is an entity responsible for the execution of an action.	*163*
Add	The Add operator performs numeric addition of its arguments.	*164*
AddressLiteral	The AddressLiteral expression returns a value of type AD with the given attributes.	*165*
After	The After operator returns true if the first interval starts after the second one ends.	*166*
AggregateExpression	Aggregate expressions perform operations on lists of data, either directly on a list of scalars, or indirectly on a list of objects, with a reference to a property present on each object in the list.	*167*
AllTrue	The AllTrue operator returns true if all the elements in source are true.	*168*
And	The And operator returns the logical conjunction of its arguments.	*169*
AnyTrue	The AnyTrue operator returns true if any element in source is true.	*170*
ArtifactLifeCycleEvent	An event in the life cycle of an artifact.	*171*
As	The As operator allows the result of an expression to be cast as a given target type.	*172*
AtomicAction	An action that is not further broken down into constituent actions.	*173*
Avg	The Avg operator returns the average of the elements in source.	*174*
Before	The Before operator returns true if the first interval ends before the second one starts.	*175*
Begin	The Begin operator returns the starting point of an interval.	*176*
Begins	The Begins operator returns true if the first interval begins the second.	*177*
Behavior	A behavior may be specified for a specific action or a group of actions.	*178*
Behaviors		*178*
BinaryExpression	The BinaryExpression type defines the abstract base type for all expressions that take two arguments.	*179*

BooleanLiteral	The BooleanLiteral expression returns a value of type BL with the given attributes.	*180*
Case	The Case operator allows for multiple conditional expressions to be chained together in a single expression, rather than having to nest multiple Conditional operators.	*181*
CaseItem		*182*
Ceiling	The Ceiling operator returns the first integer greater than or equal to the argument.	*183*
ClinicalRequest	The clinical request expression defines clinical data that will be used by the artifact.	*184*
Coalesce	The Coalesce operator returns the first non-null result in a list of argument.	*187*
CodedOrdinalLiteral	The CodedOrdinalLiteral expression returns a value of type CO with the given attributes.	*188*
CodeLiteral	The CodeLiteral expression returns a value of type CD with the given attributes.	*190*
Collapse	The Collapse operator returns the unique set of intervals that completely covers the ranges present in the given list of intervals.	*192*
CollectInformationAction	This action requests information from the actor.	*192*
Combine	The Combine operator combines a list of strings, optionally separating each string with the given separator.	*194*
ComplexLiteral	The ComplexLiteral expression allows an xml literal of any type to be included in an expression.	*195*
Concat	The Concat operator performs string concatenation of its arguments.	*196*
Condition	A condition specifies when a knowledge component is to be executed.	*197*
Conditional	The Conditional operator evaluates a condition, and returns the then argument if condition evaluates to true; otherwise the result of the else argument is returned.	*198*
Conditions	A collection of conditions that are used to define whether various aspects of the artifact, such as whether or not a particular action should be executed, or whether a particular order set item is applicable to a given patient.	*200*
Contains	The Contains operator returns true if the first operand contains the second.	*200*
Contribution	A contribution is made by a specific contributor (organization, person, etc.), and was made in a particular way, as specified by the contributor's role.	*201*
Convert	The Convert operator converts a value to a specific type.	*202*
Count	The Count operator returns the number of non-null elements in the source.	*203*
Coverage	Specifies various attributes of the patient population for whom and/or environment of care in which the CDS artifact is applicable.	*204*
CreateAction	A new action to be executed by a user or a computer system.	*206*
Current	The Current expression returns the value of the object currently in scope.	*207*
DataRequest	The DataRequest expression provides basic data access.	*208*
Date	The Date operator constructs a date/time value from the given components.	*209*
DateAdd	The DateAdd operator adds numberOfPeriods date periods of the specified granularity to the given date.	*211*
DateDiff	The DateDiff operator returns the number of granularity boundaries occurring between startDate and endDate.	*213*
DateOf	The DateOf operator returns the date (with no time component) of the argument.	*214*
DatePart	The DatePart operator returns the granularity component of the given date.	*215*
DeclareResponseAction	The DeclareResponseAction provides a mechanism to declare a container for responses provided by the user in response to CollectInformationActions.	*216*
Difference	The Difference operator returns the difference of the two arguments.	*218*
Distinct	The Distinct operator takes a list of elements and returns a list containing only the unique elements within the input.	*219*
Divide	The Divide operator performs numeric division of its arguments.	*220*

DocumentationItem	An item type representing the definition of an individual item to be recorded in a structured clinical document.	*220*
End	The End operator returns the ending point of an interval.	*223*
Ends	The Ends operator returns true if the first interval ends the second.	*224*
EntityNameLiteral	The EntityNameLiteral expression returns a value of type EN with the given attributes.	*225*
Equal	The Equal operator returns true if the arguments are equal; and false otherwise.	*226*
Evidence	Reference to research on which the artifact is based.	*227*
Expand	The Expand operator flattens a list of lists into a single list.	*229*
Expression	The Expression type defines the abstract base type for all expressions used in the HeDS expression language.	*230*
ExpressionConstraint	A constraint specified in the form of an expression.	*230*
ExpressionDef	The ExpressionDef type defines an expression and an associated name that can be referenced by any expression in the artifact.	*231*
ExpressionRef	The ExpressionRef type defines an expression that references a previously defined NamedExpression.	*232*
Filter	The Filter operator returns a list with only those elements in the source list for which the condition element evaluates to true.	*233*
FireEventAction	This action fires an event.	*235*
First	Returns the first element in a list.	*236*
Floor	The Floor operator returns the first integer less than or equal to the argument.	*238*
ForEach	The ForEach expression iterates over the list of elements in the source element, and returns a list with the same number of elements, where each element in the new list is the result of evaluating the element expression for each element in the source list.	*238*
Greater	The Greater operator returns true if the first argument is greater than the second argument.	*240*
GreaterOrEqual	The GreaterOrEqual operator returns true if the first argument is greater than or equal to the second argument.	*241*
GroupOrganizationBehavior	For a group of actions, specifies the organizational intent of the grouping.	*242*
GroupSelectionBehavior	For a group of actions, specifies the number of actions that may be chosen by an end user.	*242*
IdentifierLiteral	The IdentifierLiteral expression returns a value of type II with the given attributes.	*243*
IfNull	The IfNull operator replaces a null with the result of a given expression.	*244*
In	The In operator tests for membership in an interval or list.	*245*
IncludedIn	The IncludedIn operator returns true if the first operand is completely included in the second.	*246*
Includes	The Includes operator returns true if the first operand completely includes the second.	*247*
Indexer	The Indexer operator returns the indexth element in a string or list.	*248*
IndexOf	The IndexOf operator returns the 1-based index of the given element in the given source list.	*249*
InlineResource	An Inline Resource consists of both the resource reference information and the actual resource content/payload to be inserted inline.	*251*
IntegerIntervalLiteral	The IntegerIntervalLiteral expression returns a value of type IVL_INT with the given attributes.	*252*
IntegerLiteral	The IntegerLiteral expression returns a value of type INT with the given attributes.	*253*
Intersect	The Intersect operator returns the intersection of the operands.	*254*
Interval	The Interval selector defines an interval value.	*255*
InValueSet	The InValueSet operator returns true if the given code, or list of codes, is in the given value set.	*257*

Is	The Is operator allows the type of a result to be tested.	*258*
IsEmpty	The IsEmpty operator returns true if the list contains no elements.	*259*
IsNotEmpty	The IsNotEmpty operator returns true if the list contains any elements.	*260*
IsNull	The IsNull operator determines whether or not its argument evaluates to null.	*261*
ItemDefinition	An item definition is the equivalent of an item in a data dictionary or a catalog in an electronic health record system.	*262*
KnowledgeDocument	A knowledge document is an instance of a CDS knowledge artifact such as a rule, an order set, or a documentation template	*262*
KnowledgeResource	KnowledgeResource specifies a reference to an associated resource of relevance to the artifact such as a guideline, a performance measure, another knowledge artifact, or a source of evidence for the artifact.	*265*
Last	The Last operator returns the last element in a list.	*268*
Length	The Length operator returns the length of its argument.	*269*
Less	The Less operator returns true if the first argument is less than the second argument.	*270*
LessOrEqual	The LessOrEqual operator returns true if the first argument is less than or equal to the second argument.	*270*
List	The List selector returns a value of type List, whose elements are the result of evaluating the arguments to the List selector, in order.	*271*
ListConstraint	A constraint specifying that the value is from a list included here.	*272*
Literal	The Literal type defines a single scalar value.	*274*
Ln	The Ln operator computes the natural logarithm of its argument.	*275*
Log	The Log operator computes the logarithm of its first argument, using the second argument as the base.	*276*
Lower	The Lower operator returns the lower case of its argument.	*277*
Max	The Max operator returns the maximum element in the source.	*277*
MaxValue	The MaxValue operator returns the maximum representable value for the given type.	*278*
Median	The Median operator returns the median of the elements in source.	*279*
Meets	The Meets operator returns true if the first interval ends on the starting point of the second, or if the first interval starts on the ending point of the second.	*280*
Metadata	The container for all of the metadata associated with a CDS knowledge artifact.	*281*
Min	The Min operator returns the minimum element in the source.	*288*
MinValue	The MinValue operator returns the minimum representable value for the given type.	*289*
Mode	The Mode operator returns the statistical mode of the elements in source.	*290*
ModelReference	A reference to some model by its Universal Resource Identifier.	*291*
Modulo	The Modulo operator computes the remainder of the division of its arguments.	*292*
Multiply	The Multiply operator performs numeric multiplication of its arguments.	*292*
NaryExpression	The NaryExpression type defines an abstract base class for an expression that takes any number of arguments, including zero.	*293*
Negate	The Negate operator returns the negative of its argument.	*294*
Not	The Not operator returns the logical negation of its argument.	*295*
NotEqual	The NotEqual operator returns true if its arguments are not the same value.	*296*
Now	Returns the date and time of the start timestamp associated with the evaluation request.	*297*
Null	The Null operator returns a null, or missing information marker.	*297*
ObjectDescriptor	The ObjectDescriptor type allows arbitrary object descriptors to be built, allowing for the creation of a value that has enough information to uniquely identify an object, but is not the object itself.	*298*
ObjectExpression	The ObjectExpression type allows objects of any type to be built up as an expression.	*300*

ObjectRedefine	The ObjectRedefine expression returns an object of the same type as the source argument, with the same values for each property, except the values of the properties listed in the property elements.	*301*
Or	The Or operator returns the logical disjunction of its arguments.	*302*
Organization	Identifies an organization, a corporation, an institution, or a government department that has relevance to the knowledge artifact.	*303*
Overlaps	The Overlaps operator returns true if the first interval overlaps the second.	*304*
OverlapsAfter	The OverlapsAfter operator returns true if the first interval overlaps and ends after the second.	*305*
OverlapsBefore	The OverlapsBefore operator returns true if the first interval starts before and overlaps the second.	*306*
ParameterDef	The ParameterDef type defines a parameter that can be referenced by name anywhere within an expression.	*307*
ParameterRef	The ParameterRef expression allows the value of a parameter to be referenced as part of an expression.	*308*
Party	Party represents the abstract base type for entities that have addresses and contact information.	*309*
PeriodLiteral	The PeriodLiteral expression returns a value of type PIVL_TS with the given attributes.	*310*
Person	Identifies a person who is associated with the knowledge artifact.	*313*
PhysicalQuantityIntervalLiteral	The PhysicalQuantityIntervalLiteral expression returns a value of type IVL_PQ with the given attributes.	*314*
PhysicalQuantityLiteral	The PhysicalQuantityLiteral expression returns a value of type PQ with the given attributes.	*316*
PopulationStdDev	The PopulationStdDev operator returns the statistical standard deviation of the elements in source.	*317*
PopulationVariance	The PopulationVariance operator returns the statistical population variance of the elements in source.	*318*
Pos	The Pos operator returns the 1-based index of the given pattern in the given string.	*319*
Power	The Power operator raises the first argument to the power given by the second argument.	*320*
PrecheckBehavior	For a particular action, specifies how often the action is expected to be selected in the particular context of the group containing that action.	*321*
Pred	The Pred operator returns the predecessor of the argument.	*322*
ProperIncludedIn	The ProperIncludedIn operator returns true if the first operand is included in the second, and is strictly smaller.	*322*
ProperIncludes	The ProperIncludes operator returns true if the first operand includes the second, and is strictly larger.	*323*
Property	The Property operator returns the value of the property on source specified by the path attribute.	*324*
PropertyExpression	The PropertyExpression type is used within the ObjectLiteral type to provide the value of a specific property within an object literal expression.	*326*
QuantityIntervalLiteral	The QuantityIntervalLiteral expression returns a value of type IVL_QTY with the given attributes.	*327*
RangeConstraint		*328*
RatioLiteral	The RaioLiteral expression returns a value of type RTO with the given numerator and denominator.	*329*
RealIntervalLiteral	The RealIntervalLiteral expression returns a value of type IVL_REAL with the given attributes.	*330*
RealLiteral	The RealLiteral expression returns a value of type Real with the given attributes.	*332*
RemoveAction	This action removes another proposed action or an ongoing action.	*333*
RequestBase	The request expression defines the data that will be used by the artifact.	*334*

RequiredBehavior	For a single action, specifies what level of requiredness is associated with the action.	*337*
ResourceRelationshipReference	ResourceRelationshipReference defines the association between a resource and a resource set.	*337*
ResponseBinding	Defines the attributes required to specify a binding path for documentation item responses.	*339*
RightsDeclaration	This specifies the intellectual property rights associated with this CDS knowledge artifact, including who the rights holder is and what rights they assert.	*340*
Round	The Round operator returns the nearest integer to its argument.	*341*
SetSubsumes	The SetSubsumes operator returns the list of descendants that were subsumed by some code in the list of ancestors.	*343*
SimpleCodeLiteral	The SimpleCodeLiteral expression returns a value of type CS with the given attributes.	*344*
Sort	The Sort operator returns a list with all the elements in source, sorted by the given orderBy.	*345*
Split	The Split operator splits a string into a list of strings using a separator.	*346*
StdDev	The StdDev operator returns the statistical standard deviation of the elements in source.	*347*
StringLiteral	The StringLiteral expression returns a value of type ST with the given attributes.	*348*
Substring	The Substring operator returns the string within stringToSub, beginning at the 1-based index startIndex, and consisting of length characters.	*349*
Subsumes	The Subsumes operator returns true if the operands were of the same code system, and the ancestor operand subsumed the descendant operand in the hierarchy of the code system.	*351*
Subtract	The Subtract operator performs numeric subtraction of its arguments.	*352*
Succ	The Succ operator returns the successor of the argument.	*353*
Sum	The Sum operator returns the sum of non-null elements in the source.	*353*
SupportingEvidence	The evidence grade and the sources of evidence associated with this artifact.	*354*
SupportingResource	Didactic or other informational resources associated with the artifact that can be provided to the CDS recipient.	*355*
TernaryExpression	The TernaryExpression type defines the abstract base type for all expressions that take three arguments.	*356*
TimeOf	The TimeOf operator returns the time (with no date component) of the argument.	*357*
TimestampIntervalLiteral	The TimestampIntervalLiteral expression returns a value of type IVL_TS with the given attributes.	*358*
TimestampLiteral	The TimestampLiteral expression returns a value of type TS with the given attributes.	*359*
Today	The Today operator returns the date (with no time component) of the start timestamp associated with the evaluation request.	*360*
Trigger		*361*
Triggers		*362*
Truncate	The Truncate operator returns the integer component of its argument.	*363*
TruncatedDivide	The TruncatedDivide operator performs integer division of its arguments.	*363*
UnaryExpression	The UnaryExpression type defines the abstract base type for all expressions that take a single argument.	*364*
Union	The Union operator returns the union of the operands.	*365*
UpdateAction	This action changing the value of another existing action.	*366*
Upper	The Upper operator returns the upper case of its argument.	*367*
UrlLiteral	The UrlLiteral expression returns a value of type TEL with the given attributes.	*368*
ValueSet	The ValueSet operator returns a list of codes whose elements are defined by the given value set authority for the given value set id and version.	*369*
ValueSetConstraint	A constraint specifying that the value is an item from the value set specified.	*371*

Variance	The Variance operator returns the statistical variance of the elements in source.	372
VersionedIdentifier	VersionedIdentifier is composed of two parts: (1) an II identifier which identifies the set of all versions of a given resource. (2) the actual version of the instance of interest in this set.	373

Simple Type Summary		Page
ArtifactLifeCycleEventType	A version of an artifact may have different actions performed on it during the course of its life cycle.	374
ArtifactLifeCycleEventTypeCore	A lifecycle event is applicable to a specific version of an artifact.	374
ArtifactLifeCycleEventTypeExt	To add new items to the enumeration ArtifactLifeCycleEvent, comment out or delete the restriction on ArtifactLifeCycleEventTypeCore.	375
ArtifactStatusType	A specific status is associated with each version of an artifact.	376
ArtifactStatusTypeCore		377
ArtifactStatusTypeExt	To add new items to the enumeration ArtifactStatus, comment out or delete the restriction on ArtifactStatusTypeCore.	378
ArtifactType	Three types of artifacts are in scope for Health eDecisions Use Case #1: Event-condition-action rules, order sets and documentation templates.	378
ArtifactTypeCore		379
ArtifactTypeExt		380
Cardinality	Cardinality defines the expected cardinality of an element, single or multiple.	380
ConditionRoleType	The roles that a condition plays in the execution of a component.	381
ConditionRoleTypeCore		382
ConditionRoleTypeExt		382
ContributorType	Enumeration of roles that contribute to the development and maintenance of a knowledge artifact.	383
ContributorTypeCore		383
ContributorTypeExt		384
CoverageType	Specifies clinical metadata that can be used to retrieve, index and/or categorize the knowledge artifact.	385
CoverageTypeCore		385
CoverageTypeExt		386
DataEventType	Enumeration of types of events related to access, creation, removal, or update of data.	387
DataEventTypeCore		388
DataEventTypeExt		388
DateGranularity	The DateGranularity type specifies the granularities available for temporal operations such as DateAdd, DateDiff, and DatePart.	389
EventType	An enumeration of event types.	390
EventTypeCore		390
EventTypeExt		391
GroupOrganizationBehaviorType	Defines organization behavior of a group: gives the reason why the items are grouped together.	392
GroupOrganizationBehaviorTypeCore		392
GroupOrganizationBehaviorTypeExt		393

GroupSelectionBehaviorType	Defines selection behavior of a group: specifies the number of selectable items in the group that may be selected by the end user when the items of the group are displayed.	*394*
GroupSelectionBehaviorTypeCore		*394*
GroupSelectionBehaviorTypeExt		*395*
PrecheckBehaviorType	Defines selection frequency behavior for an action or group; i.e., for most frequently selected items, the end-user system may provide convenience options in the UI (such as pre-selection) in order to (1) communicate to the end user what the most frequently selected item is, or should, be in a particular context, and (2) save the end user time.	*396*
PrecheckBehaviorTypeCore		*397*
PrecheckBehaviorTypeExt		*397*
RangeConstraintType	The enumeration of different types of range constraints on values.	*398*
RangeConstraintTypeCore		*398*
RangeConstraintTypeExt		*399*
RequestCardinality	RequestCardinality defines the expected cardinality of the request, single or multiple.	*400*
RequiredBehaviorType	Defines requiredness behavior for selecting an action or an action group; i.e., whether the action or action group is required or optional.	*401*
RequiredBehaviorTypeCore		*401*
RequiredBehaviorTypeExt		*402*
ResourceRelationshipType	A specific status is associated with each version of an artifact.	*402*
ResourceRelationshipTypeCore		*403*
ResourceRelationshipTypeExt	To add new items to the enumeration ArtifactStatus, comment out or delete the restriction on ArtifactStatusTypeCore.	*404*
ValueType	A specification of a constraint on the range of values for an item.	*404*
ValueTypeCore		*405*
ValueTypeExt		*406*

element <itemDefinitions> (global)

Namespace: urn:hl7-org:knowledgeartifact:r1
Type: anonymous complexType
Includes: definition of 1 element

XML Representation Summary

```
<itemDefinitions>
   Content: item*
</itemDefinitions>
```

Content model elements (1):

item (in itemDefinitions) [157]

Annotation

A collection of item definitions

XML Source (w/o annotations (1))

```
<xs:element name="itemDefinitions">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="item" type="ItemDefinition"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

item

Type: *ItemDefinition* [262], complex content

XML Representation Summary

```
<item>
    Content: identifier?
</item>
```

element <knowledgeDocument> (global)

Namespace: urn:hl7-org:knowledgeartifact:r1
Type: KnowledgeDocument [262]

XML Representation Summary

```
<knowledgeDocument>
    Content: metadata, externalData?, expressions?, triggers?, conditions?, behaviors?, actionGroup
</knowledgeDocument>
```

Content model elements (7):

actionGroup (in knowledgeDocument) [265], externalData [264],
behaviors (in knowledgeDocument) [265], metadata [264],
conditions (in knowledgeDocument) [265], triggers [264]
expressions [264],

Annotation

knowledgeDocument represents a serialized Clinical Decision Support (CDS) knowledge artifact. It is intended to define a general serialization format for an Order Set, a Event-Condition-Action (ECA) Rule, and a Document Template. Given the general intended purpose for this schema, it is important to note that the serialization of a given CDS artifact is defined by both this schema, in conjunction with a template defining artifact-specific constraints. For additional information on these artifact-specific constraints, please refer to the Implementation Guide.

XML Source (w/o annotations (1))

```
<xs:element name="knowledgeDocument" type="KnowledgeDocument"/>
```

complexType "Abs"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Abs operator returns the absolute value of its argument.

If the argument is null, the result is null.

The Abs operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **Abs**

XML Source (w/o annotations (1))

```
<xs:complexType name="Abs">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ActionBase"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definitions of 6 elements

XML Representation Summary

```
<...>
    Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?
</...>
```

Content Model Elements (6):

actionId [160], conditions (defined in ActionBase complexType) [160],
actors [160], supportingEvidence (defined in ActionBase complexType) [160],
behaviors (defined in ActionBase complexType) [160], supportingResources [160]

Annotation

Actions are the output of the CDS system and represent the tasks that must be carried out by a human or a computer system.

XML Source (w/o annotations (7))

```
<xs:complexType abstract="true" name="ActionBase">
  <xs:sequence minOccurs="0">
    <xs:element maxOccurs="1" minOccurs="0" name="actionId" type="dt:II"/>
    <xs:element minOccurs="0" name="supportingEvidence" type="SupportingEvidence"/>
    <xs:element minOccurs="0" name="supportingResources" type="SupportingResource"/>
    <!-- TODO -->
    <xs:element maxOccurs="1" minOccurs="0" name="actors">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="actor" type="Actor"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="behaviors" type="Behaviors"/>
    <!-- Additional conditions specific to the action execution -->
    <xs:element maxOccurs="1" minOccurs="0" name="conditions" type="Conditions"/>
    <!-- <xs:element name="applicableScenario" type="Expression" minOccurs="0"
    maxOccurs="1"> <xs:annotation> <xs:documentation>The clinical (sub) scenario
    in which this action is applicable. This scenario is additive to any scenarios
    specified in the containers of this action such as action groups or the knowledge
    document. In other words, this scenario does not override the scenario constraints
    specified in the container elements. </xs:documentation> </xs:annotation>
    </xs:element> -->
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 6/6)

actionId

Type: dt:II [100], empty content

An identifier for the action. The identifier must be unique within the scope of the artifact.

XML Representation Summary

```
<actionId
    extension      = xs:string
    identifierName = xs:string
    root           = xs:string
/>
```

supportingEvidence

Type: SupportingEvidence [354], complex content

The evidence grade and the sources of evidence associated with this artifact.

XML Representation Summary

```
<supportingEvidence>
    Content: evidence+
</supportingEvidence>
```

supportingResources

Type: SupportingResource [355], complex content

Didactic or other informational resources associated with the action that can be provided to the CDS recipient. Information resources can include inline text commentary and links to web resources.

XML Representation Summary

```
<supportingResources>
    Content: resource+
</supportingResources>
```

actors

Type: anonymous complexType, complex content

XML Representation Summary

```
<actors>
    Content: actor+
</actors>
```

behaviors

Type: Behaviors [178], complex content

The behaviors associated with how the action is presented and executed. The semantics and the validity of behaviors for actions are described elsewhere.

XML Representation Summary

```
<behaviors>
    Content: behavior+
</behaviors>
```

conditions

Type: Conditions [200], complex content

The conditions section lists all conditions that pertain to the action. Conditions define the logic that determine the applicability of the action in the given context, any

precondition or post condition, and/or any inclusion and exclusion criteria for the given action.

XML Representation Summary

```
<conditions>
    Content: condition+
</conditions>
```

complexType "ActionGroup"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 4 elements

XML Representation Summary

```
<...>
    Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
             title?, description?, representedConcepts?, subElements
</...>
```

Content Model Elements (10):

actionId [160],
actors [160],
behaviors (defined in *ActionBase* complexType) [160],
conditions (defined in *ActionBase* complexType) [160],
description (defined in ActionGroup complexType) [162],
representedConcepts [163],
subElements [163],
supportingEvidence (defined in *ActionBase* complexType) [160],
supportingResources [160],
title (defined in ActionGroup complexType) [162]

Annotation

This type of action is used to organize a group of related actions into one container.

The semantics of how the group's subelements interact with which each other and how the subelements might be presented are specified in the group behavior.

Type Definition Detail

Type Derivation Tree

ActionBase [158] (extension)
└ **ActionGroup**

XML Source (w/o annotations (8))

```
<xs:complexType name="ActionGroup">
  <xs:complexContent>
    <xs:extension base="ActionBase">
      <xs:sequence>
```

```
        <xs:element minOccurs="0" name="title" type="dt:ST"/>
        <xs:element minOccurs="0" name="description" type="dt:ST"/>
        <xs:element minOccurs="0" name="representedConcepts">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="concept" type="dt:CD"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="subElements">
          <xs:complexType>
            <xs:sequence>
              <xs:choice maxOccurs="unbounded">
                <xs:element name="simpleAction" type="AtomicAction"/>
                <xs:element name="actionGroup" type="ActionGroup"/>
                <xs:element name="actionGroupReference" type="VersionedIdentifier"/>
              </xs:choice>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 4/10)

title

Type: dt:ST [124], empty content

A brief title that is shown to the user of the artifact, i.e., the recipient of the CDS

XML Representation Summary

```
<title
    value = xs:string
/>
```

description

Type: dt:ST [124], empty content

A lengthier description of the action group that can
be displayed to the user or the recipient of the CDS

XML Representation Summary

```
<description
    value = xs:string
/>
```

representedConcepts

Type: anonymous complexType, complex content

The concept(s) represented by this action group. For instance, a section may represent a group of beta-blockers, a composite orderable such as an insulin sliding scale, or a set of order sentences for a particular orderable (e.g., Tylenol).

XML Representation Summary

```
<representedConcepts>
    Content: concept+
</representedConcepts>
```

subElements

Type: anonymous complexType, complex content

The constituent elements of the group can be of different types including subgroups, simple or atomic actions, and subgroups embedded by reference.

While the group allows artifacts of different types to be mixed and matched in any combination, a particular type of artifact might further restrict the combinations. For example, an artifact type might require subelements of a particular group to be either groups or simple actions; elements of both types cannot exist in the group.

XML Representation Summary

```
<subElements>
    Content: (simpleAction | actionGroup | actionGroupReference)+
</subElements>
```

complexType "Actor"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: actor
</...>
```

Content Model Elements (1):

actor (in actor in actors) [164]

Annotation

An actor is an entity responsible for the execution of an action.

XML Source (w/o annotations (2))

```
<xs:complexType name="Actor">
  <xs:sequence>
    <xs:element name="actor" type="ExpressionDef"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

actor

Type: `ExpressionDef [231]`, complex content

The expression must evaluate to one of the following:
1. CD or a List of CD. In this case, the actor defines the role or roles of entities to execute the action.
2. II or a List of II. In this case, the actor defines entities, such as a provider or a team.

XML Representation Summary

```
<actor
    name = xs:string
    >
    Content: expression
</actor>
```

complexType "Add"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Add operator performs numeric addition of its arguments.

If either argument is null, the result is null.

The Add operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Add**

XML Source (w/o annotations (1))

```
<xs:complexType name="Add">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "AddressLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   use = list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "PHYS" | "PST" | "TMP" | "ABC" |
         "IDE" | "SYL" | "SRCH" | "SNDX" | "PHON")
   >
   Content: description?, part+
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], part (defined in AddressLiteral complexType) [166]

Annotation

The AddressLiteral expression returns a value of type AD with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **AddressLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="AddressLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
```

```
        <xs:element maxOccurs="unbounded" name="part" type="dt:ADXP"/>
      </xs:sequence>
      <xs:attribute name="use" type="dt:set_PostalAddressUse" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

use

Type: dt:set_PostalAddressUse [143]
Use: optional

Attribute Value

```
list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "PHYS" | "PST" | "TMP" | "ABC" | "IDE" |
"SYL" | "SRCH" | "SNDX" | "PHON")
```

Content Element Detail (all declarations; defined within this component only; 1/2)

part

Type: dt:ADXP [82], empty content

XML Representation Summary

```
<part
    type  = ("AL" | "ADL" | "UNID" | "UNIT" | "DAL" | "DINST" | "DINSTA" | "DINSTQ" | "DMOD" | "DMODID" | "SAL" | "BNR"
            | "BNN" | "BNS" | "STR" | "STB" | "STTYP" | "DIR" | "INT" | "CAR" | "CEN" | "CNT" | "CPA" | "CTY" | "DEL" |
            "POB" | "PRE" | "STA" | "ZIP" | "DPID")
    value = xs:string
/>
```

complexType "After"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The After operator returns true if the first interval starts after the second one ends. In other words, if the starting point of the first interval is greater than the ending point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **After**

XML Source (w/o annotations (1))

```
<xs:complexType name="After">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- After(i1, i2) = Begin(i1) > End(i2) -->
</xs:complexType>
```

complexType "AggregateExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

Aggregate expressions perform operations on lists of data, either directly on a list of scalars, or indirectly on a list of objects, with a reference to a property present on each object in the list.

Aggregate expressions deal with missing information by excluding missing values from consideration before performing the aggregated operation. For example, in a Sum over Dose, any instance of Medication with no value for Dose would be ignored.

An aggregate operation performed over an empty list is defined to return null, except as noted in the documentation for each operator (Count, AllTrue, and AnyTrue are the exceptions).

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ ***AggregateExpression***

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="AggregateExpression">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="source" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="path" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

path

Type: `xs:string`, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

complexType "AllTrue"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
    path = xs:string
    >
    Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in Expression complexType) [230], source (defined in AggregateExpression complexType) [168]

Annotation

The AllTrue operator returns true if all the elements in source are true.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, true is returned.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ AggregateExpression [167] (extension)
      └ AllTrue
```

XML Source (w/o annotations (1))

```
<xs:complexType name="AllTrue">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "And"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand*
</...>
```

Content Model Elements (2):

description (defined in Expression complexType) [230], operand (defined in NaryExpression complexType) [294]

Annotation

The And operator returns the logical conjunction of its arguments. Note that this operator is defined as n-ary, allowing any number of arguments. The result of And with no arguments is defined to be false. The result of And with a single argument is defined to be the result of the argument. The result of and with two arguments is defined using 3-valued logic semantics. This means that if either argument is false, the result is false; if both arguments are true, the result is true; otherwise, the result is null. The result of more than two arguments is defined as successive invocations of And.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *NaryExpression* [293] (extension)

└ **And**

XML Source (w/o annotations (1))

```
<xs:complexType name="And">
  <xs:complexContent>
    <xs:extension base="NaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "AnyTrue"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The AnyTrue operator returns true if any element in source is true.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, false is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *AggregateExpression* [167] (extension)

└ **AnyTrue**

XML Source (w/o annotations (1))

```
<xs:complexType name="AnyTrue">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ArtifactLifeCycleEvent"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: eventType, eventDateTime
</...>
```

Content Model Elements (2):

eventDateTime [172], eventType (in artifactLifeCycleEvent) [172]

Annotation

An event in the life cycle of an artifact. Both the type of event are specified, as well as the point in time in which that event took place.

XML Source (w/o annotations (1))

```
<xs:complexType name="ArtifactLifeCycleEvent">
  <xs:sequence>
    <xs:element name="eventType">
      <xs:complexType>
        <xs:attribute name="value" type="ArtifactLifeCycleEventType" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="eventDateTime" type="dt:TS"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

eventType

Type: anonymous complexType, empty content

XML Representation Summary

```
<eventType
    value = (("Created" | "Pre-published" | "Published" | "Reviewed" | "Withdrawn" | "Superseded") | ("Created" | "Pre-
            published" | "Published" | "Reviewed" | "Withdrawn" | "Superseded"))
/>
```

eventDateTime

Type: dt:TS [126], empty content

XML Representation Summary

```
<eventDateTime
    value = xs:string
/>
```

complexType "As"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
    asType = xs:QName
    strict = xs:boolean : "false"
    >
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The As operator allows the result of an expression to be cast as a given target type. This allows expressions to be written that are statically typed against the expected run-time type of the argument. If the argument is not of the specified type, strict determines whether the result is null (the default), or an exception is thrown.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **As**

XML Source (w/o annotations (1))

```
<xs:complexType name="As">
  <xs:complexContent>
    <xs:extension base="UnaryExpression">
      <xs:attribute name="asType" type="xs:QName" use="required"/>
      <xs:attribute default="false" name="strict" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

asType

Type: `xs:QName`, predefined
Use: required

strict

Type: `xs:boolean`, predefined
Use: optional

Attribute Value

Default: `"false"`

complexType "AtomicAction"

Namespace: `urn:hl7-org:knowledgeartifact:r1`
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 element

XML Representation Summary

```
<...>
  Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
           textEquivalent?
</...>
```

Content Model Elements (7):

actionId [160],
supportingEvidence (defined in *ActionBase* complexType) [160],
actors [160],
supportingResources [160],
behaviors (defined in *ActionBase* complexType) [160],
textEquivalent [174]

conditions (defined in ActionBase complexType) [160],

Annotation

An action that is not further broken down into constituent actions.

Type Definition Detail

Type Derivation Tree

ActionBase [158] (extension)

└ **AtomicAction**

XML Source (w/o annotations (2))

```
<xs:complexType abstract="true" name="AtomicAction">
  <xs:complexContent>
    <xs:extension base="ActionBase">
      <xs:sequence>
        <xs:element minOccurs="0" name="textEquivalent" type="dt:ST"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/7)

textEquivalent

Type: dt:ST [124], empty content

A brief textual description of the action that summarizes the action

XML Representation Summary

```
<textEquivalent
    value = xs:string
/>
```

complexType "Avg"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Avg operator returns the average of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *AggregateExpression* [167] (extension)
  └ **Avg**

XML Source (w/o annotations (1))

```
<xs:complexType name="Avg">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Before"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Before operator returns true if the first interval ends before the second one starts. In other words, if the ending point of the first interval is less than the starting point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Before
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Before">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Before(i1, i2) = End(i1) < Begin(i2) -->
</xs:complexType>
```

complexType "Begin"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Begin operator returns the starting point of an interval.

If the beginning of the interval is open, this operator returns the Successor of the defined beginning point of the interval.

If the beginning of the interval is closed, this operator returns the defined beginning point of the interval if it is not null.

Otherwise, it returns the minimum value of the point type of the interval.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **Begin**

XML Source (w/o annotations (1))

```
<xs:complexType name="Begin">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
  <!-- Begin(i) = if not(i.beginOpen) then IfNull(i.begin, T.MinimumValue()) else Succ(i.begin) -->
</xs:complexType>
```

complexType "Begins"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Begins operator returns true if the first interval begins the second. In other words, if the starting point of the first is equal to the starting point of the second interval and the ending point of the first interval is less than or equal to the ending point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Begins
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Begins">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Begins(i1, i2) = Begin(i1) = Begin(i2) and End(i1) <= End(i2) -->
</xs:complexType>
```

complexType "Behavior"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = xs:anySimpleType
/>
```

Annotation

A behavior may be specified for a specific action or a group of actions. This is the base type for all Behaviors.

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="Behavior">
  <xs:attribute name="value"/>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

  Type: xs:anySimpleType
  Use: optional

complexType "Behaviors"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: behavior+
</...>
```

Content Model Elements (1):

behavior [179]

XML Source

```
<xs:complexType name="Behaviors">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" name="behavior" type="Behavior"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

behavior

Type: *Behavior* [178], empty content

XML Representation Summary

```
<behavior
    value = xs:anySimpleType
/>
```

complexType "BinaryExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The BinaryExpression type defines the abstract base type for all expressions that take two arguments.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **BinaryExpression**

XML Source (w/o annotations (1))

```
<xs:complexType name="BinaryExpression">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="2" minOccurs="2" name="operand" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

operand

Type: *Expression* [230], complex content

XML Representation Summary

```
<operand>
    Content: description?
</operand>
```

complexType "BooleanLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = xs:boolean
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The BooleanLiteral expression returns a value of type BL with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **BooleanLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="BooleanLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

  Type: `xs:boolean`, predefined
  Use: required

complexType "Case"

Namespace: `urn:hl7-org:knowledgeartifact:r1`
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
  Content: description?, comparand?, caseItem+, else
</...>
```

Content Model Elements (4):

caseItem [182], description (defined in *Expression* complexType) [230],
comparand [182], else (defined in Case complexType) [182]

Annotation

The Case operator allows for multiple conditional expressions to be chained together in a single expression, rather than having to nest multiple Conditional operators. In addition, the comparand operand provides a variant on the case that allows a single value to be compared in each conditional.

If a comparand is not provided, the type of each when element of the caseItems within the Case is expected to be boolean. If a comparand is provided, the type of each when element of the caseItems within the Case is expected to be of the same type as the comparand. An else element must always be provided.

The static type of the then argument within the first caseItem determines the type of the result, and the then argument of each subsequent caseItem and the else argument must be of that same type.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Case**

XML Source (w/o annotations (1))

```
<xs:complexType name="Case">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="comparand" type="Expression"/>
        <xs:element maxOccurs="unbounded" minOccurs="1" name="caseItem" type="CaseItem"/>
        <xs:element maxOccurs="1" minOccurs="1" name="else" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/4)

comparand

Type: *Expression* [230], complex content

XML Representation Summary

```
<comparand>
    Content: description?
</comparand>
```

caseItem

Type: CaseItem [182], complex content

XML Representation Summary

```
<caseItem>
    Content: when, then
</caseItem>
```

else

Type: *Expression* [230], complex content

XML Representation Summary

```
<else>
    Content: description?
</else>
```

complexType "CaseItem"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: when, then
</...>
```

Content Model Elements (2):

then (in caseItem) [183], when [183]

XML Source

```
<xs:complexType name="CaseItem">
  <xs:sequence>
    <xs:element maxOccurs="1" minOccurs="1" name="when" type="Expression"/>
    <xs:element maxOccurs="1" minOccurs="1" name="then" type="Expression"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

when

Type: *Expression* [230], complex content

XML Representation Summary

```
<when>
    Content: description?
</when>
```

then

Type: *Expression* [230], complex content

XML Representation Summary

```
<then>
    Content: description?
</then>
```

complexType "Ceiling"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Ceiling operator returns the first integer greater than or equal to the argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ UnaryExpression [364] (extension)
      └ Ceiling
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Ceiling">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ClinicalRequest"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 5 attributes, 3 elements

XML Representation Summary

```
<...
    scope            = xs:string
    cardinality      = ("Single" | "Multiple")
    dataType         = xs:QName
    templateId       = xs:string
    idProperty       = xs:string
    triggerType      = (("DataElementAdded" | "DataElementModified" | "DataElementRemoved" |
                       "DataElementAccessed" | "DataElementAccessEnded") | ("DataElementAdded" |
                       "DataElementModified" | "DataElementRemoved" | "DataElementAccessed" |
                       "DataElementAccessEnded"))
    isInitial        = xs:boolean : "true"
    codeProperty     = xs:string
    dateProperty     = xs:string
    subjectProperty  = xs:string
    useValueSets     = xs:boolean : "false"
    useSubsumption   = xs:boolean : "false"
    >
    Content: description?, timeOffset?, codes?, dateRange?, subject?
</...>
```

Content Model Elements (5):

codes (defined in ClinicalRequest complexType) [187], subject [187], dateRange [187], timeOffset [336], description (defined in *Expression* complexType) [230],

Annotation

The clinical request expression defines clinical data that will be used by the artifact. This expression expands on the functionality provided by the RequestBase to provide clinically relevant filtering criteria in a well-defined and computable way. This operation defines the integration boundary for artifacts.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *RequestBase* [334] (extension)

└ **ClinicalRequest**

XML Source (w/o annotations (9))

```
<xs:complexType name="ClinicalRequest">
  <xs:complexContent>
    <xs:extension base="RequestBase">
```

```
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="codes" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="dateRange" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="subject" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="codeProperty" type="xs:string" use="optional"/>
      <xs:attribute name="dateProperty" type="xs:string" use="optional"/>
      <xs:attribute name="subjectProperty" type="xs:string" use="optional"/>
      <xs:attribute default="false" name="useValueSets" type="xs:boolean" use="optional"/>
      <xs:attribute default="false" name="useSubsumption" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 5/12)

codeProperty

Type: `xs:string`, predefined
Use: optional

The codeProperty attribute optionally specifies which property of the model contains the Code or Codes for the clinical statement.

Note that this property could potentially be specified elsewhere as part of an implementation catalog, rather than on each Request expression, but allowing them at the request expression level gives the most flexibility.

dateProperty

Type: `xs:string`, predefined
Use: optional

The dateProperty attribute optionally specifies which property of the model contains the clinically relevant date for the clinical statement.

Note that this property could potentially be specified elsewhere as part of an implementation catalog, rather than on each Request expression, but allowing them at the request expression level gives the most flexibility.

subjectProperty

Type: `xs:string`, predefined
Use: optional

The subjectProperty optionally specifies which property of the model contains the related value for the subject of the clinical request.

useValueSets

Type: `xs:boolean`, predefined
Use: optional

The useValueSets attribute determines whether references to value sets in the Codes element will be expanded, or left as value set references for the purposes of communicating across the data boundary.

Attribute Value

Default: `"false"`

useSubsumption

Type: `xs:boolean`, predefined
Use: optional

The useSubsumption attribute determines whether subsumption should be used to determine whether or not a given data item should be included in the result.

Attribute Value

Default: `"false"`

Content Element Detail (all declarations; defined within this component only; 3/5)

codes

Type: *Expression* [230], complex content

The codes element optionally allows a set of codes to be provided. The codes list restricts the set of clinical statements returned to only those clinical statements that matched some code in the set.

XML Representation Summary

```
<codes>
    Content: description?
</codes>
```

dateRange

Type: *Expression* [230], complex content

The dateRange element optionally allows a date range to be provided. The clinical statements returned would be only those clinical statements whose date fell within the range specified.

XML Representation Summary

```
<dateRange>
    Content: description?
</dateRange>
```

subject

Type: *Expression* [230], complex content

The subject element specifies how the data in the request is related to the subject of the artifact. For a typical clinical decision support artifact, the subject is assumed to be a single patient, and the use of these properties (subject and subjectProperty) is optional, meaning that the engine is assuming a single-patient context and is already managing the per-patient relationships of the data requests involved in the artifact. However, making this relationship between the subject and the related data requests explicit adds more flexibility, potentially allowing an artifact to range over something other than patient; encounters, for example.

XML Representation Summary

```
<subject>
    Content: description?
</subject>
```

complexType "Coalesce"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

XML Representation Summary

```
<...>
   Content: description?, operand*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *NaryExpression* complexType) [294]

Annotation

The Coalesce operator returns the first non-null result in a list of argument. If all arguments evaluate to null, the result is null. The static type of the first argument determines the type of the result, and all subsequent arguments must be of that same type.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ NaryExpression [293] (extension)
      └ Coalesce
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Coalesce">
  <xs:complexContent>
    <xs:extension base="NaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "CodedOrdinalLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 5 attributes

XML Representation Summary

```
<...
   codeSystem      = xs:string
   code            = xs:string
   codeSystemName  = xs:string
   displayName     = xs:string
   value           = xs:double
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The CodedOrdinalLiteral expression returns a value of type CO with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **CodedOrdinalLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="CodedOrdinalLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="codeSystem" type="dt:Uid" use="required"/>
      <xs:attribute name="code" type="dt:Code" use="required"/>
      <xs:attribute name="codeSystemName" type="xs:string" use="optional"/>
      <xs:attribute name="displayName" type="xs:string" use="optional"/>
      <xs:attribute name="value" type="dt:Decimal" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 5/5)

codeSystem

Type: dt:Uid [147]
Use: required

Attribute Value

xs:string

code

Type: dt:Code [132]
Use: required

Attribute Value

xs:string

codeSystemName

Type: xs:string, predefined
Use: optional

displayName

Type: xs:string, predefined
Use: optional

value

Type: dt:Decimal [134]
Use: required

Attribute Value

xs:double

complexType "CodeLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 7 attributes

XML Representation Summary

```
<...
   codeSystem        = xs:string
   code              = xs:string
   codeSystemName    = xs:string
   displayName       = xs:string
   codeSystemVersion = xs:string
   valueSet          = xs:string
   valueSetVersion   = xs:string
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The CodeLiteral expression returns a value of type CD with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **CodeLiteral**

XML Source (w/o annotations (1))

<xs:complexType name="**CodeLiteral**">

```
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="codeSystem" type="dt:Uid" use="required"/>
      <xs:attribute name="code" type="dt:Code" use="required"/>
      <xs:attribute name="codeSystemName" type="xs:string" use="optional"/>
      <xs:attribute name="displayName" type="xs:string" use="optional"/>
      <xs:attribute name="codeSystemVersion" type="xs:string" use="optional"/>
      <xs:attribute name="valueSet" type="dt:Uid" use="optional"/>
      <xs:attribute name="valueSetVersion" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 7/7)

codeSystem

Type: dt:Uid [147]
Use: required

Attribute Value

xs:string

code

Type: dt:Code [132]
Use: required

Attribute Value

xs:string

codeSystemName

Type: xs:string, predefined
Use: optional

displayName

Type: xs:string, predefined
Use: optional

codeSystemVersion

Type: xs:string, predefined
Use: optional

valueSet

Type: dt:Uid [147]
Use: optional

Attribute Value

xs:string

valueSetVersion

Type: xs:string, predefined
Use: optional

complexType "Collapse"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Collapse operator returns the unique set of intervals that completely covers the ranges present in the given list of intervals.

If the list of intervals is empty, the result is empty. If the list of intervals contains a single interval, the result is a list with that interval. If the list of intervals contains nulls, they will be excluded from the resulting list.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ UnaryExpression [364] (extension)
      └ Collapse
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Collapse">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "CollectInformationAction"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
   Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
            textEquivalent?, documentationConcept, initialValue?, responseBinding?
</...>
```

Content Model Elements (10):

actionId [160],
actors [160],
behaviors (defined in ActionBase complexType) [160],
conditions (defined in ActionBase complexType) [160],
documentationConcept [194],
initialValue [194],
responseBinding [194],
supportingEvidence (defined in ActionBase complexType) [160],
supportingResources [160],
textEquivalent [174]

Annotation

This action requests information from the actor. The information request is specified as a DocumentationItem.

Type Definition Detail

Type Derivation Tree

```
ActionBase [158] (extension)
  └ AtomicAction [173] (extension)
      └ CollectInformationAction
```

XML Source (w/o annotations (4))

```
<xs:complexType name="CollectInformationAction">
  <xs:complexContent>
    <xs:extension base="AtomicAction">
      <xs:sequence>
        <xs:element name="documentationConcept" type="DocumentationItem"/>
        <xs:element maxOccurs="1" minOccurs="0" name="initialValue" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="responseBinding" type="ResponseBinding"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/10)

documentationConcept

Type: DocumentationItem [220], complex content

This provides a specification of the information to be collected from the user.

XML Representation Summary

```
<documentationConcept>
    Content: identifier?, displayText, description?, itemCodes?, responseDataType, responseCardinality?, responseRange*
</documentationConcept>
```

initialValue

Type: *Expression* [230], complex content

An expression to compute an initial value for the documentation concept. The initial value could be computed from previous data about the patient available via expressions specified in the external data

XML Representation Summary

```
<initialValue>
    Content: description?
</initialValue>
```

responseBinding

Type: ResponseBinding [339], empty content

Defines the response binding for the documentation item.

XML Representation Summary

```
<responseBinding
    container = xs:string : "Responses"
    property  = xs:string
/>
```

complexType "Combine"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description?, source, separator?
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], source (defined in Combine complexType) [195]

separator (defined in Combine complexType) [195],

Annotation

The Combine operator combines a list of strings, optionally separating each string with the given separator.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Combine**

XML Source (w/o annotations (1))

```
<xs:complexType name="Combine">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="source" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="separator" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

separator

Type: *Expression* [230], complex content

XML Representation Summary

```
<separator>
    Content: description?
</separator>
```

complexType "ComplexLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
   Content: description?, value
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], value (defined in ComplexLiteral complexType) [196]

Annotation

The ComplexLiteral expression allows an xml literal of any type to be included in an expression.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ComplexLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="ComplexLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="value" type="xs:anyType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

value

Type: xs:anyType, any content

XML Representation Summary

```
<value>
   ...
</value>
```

complexType "Concat"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *NaryExpression* complexType) [294]

Annotation

The Concat operator performs string concatenation of its arguments.

If any argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *NaryExpression* [293] (extension)
  └ **Concat**

XML Source (w/o annotations (1))

```
<xs:complexType name="Concat">
  <xs:complexContent>
    <xs:extension base="NaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Condition"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
   Content: logic, conditionRole
</...>
```

Content Model Elements (2):

conditionRole [198], logic [198]

Annotation

A condition specifies when a knowledge component is to be executed. For example, an ECA rule uses an ApplicableScenario condition to determine whether or not the action described by the artifact should be executed.

XML Source (w/o annotations (3))

```
<xs:complexType name="Condition">
  <xs:sequence>
    <xs:element name="logic" type="Expression"/>
    <xs:element name="conditionRole">
      <xs:complexType>
        <xs:attribute name="value" type="ConditionRoleType"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

logic

Type: *Expression* [230], complex content

The logic specification of the condition. Often, though not necessarily, the logic is an expression about patient data. The expression must evaluate to a Boolean value.

XML Representation Summary

```
<logic>
    Content: description?
</logic>
```

conditionRole

Type: anonymous complexType, empty content

The role determines when to evaluate the expression and how to proceed based on the expression results. Different artifact types use different types of conditions to control various aspects of the artifact. See the condition role type enumeration documentation for more discussion.

XML Representation Summary

```
<conditionRole
    value = ("ApplicableScenario" | "ApplicableScenario")
/>
```

complexType "Conditional"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
    Content: description?, condition, then, else
</...>
```

Content Model Elements (4):

condition (defined in Conditional complexType) [199], else (defined in Conditional complexType) [200],
description (defined in *Expression* complexType) [230], then (defined in Conditional complexType) [199]

Annotation

The Conditional operator evaluates a condition, and returns the then argument if condition evaluates to true; otherwise the result of the else argument is returned. The static type of the then argument determines the result type of the conditional, and the else argument must be of that same type.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Conditional**

XML Source (w/o annotations (1))

```
<xs:complexType name="Conditional">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="condition" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="then" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="else" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/4)

condition

Type: *Expression* [230], complex content

XML Representation Summary

```
<condition>
    Content: description?
</condition>
```

then

Type: *Expression* [230], complex content

XML Representation Summary

```
<then>
    Content: description?
</then>
```

else

Type: Expression [230], complex content

XML Representation Summary

```
<else>
    Content: description?
</else>
```

complexType "Conditions"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: condition+
</...>
```

Content Model Elements (1):

condition (defined in Conditions complexType) [200]

Annotation

A collection of conditions that are used to define whether various aspects of the artifact, such as whether or not a particular action should be executed, or whether a particular order set item is applicable to a given patient.

XML Source (w/o annotations (1))

```
<xs:complexType name="Conditions">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" name="condition" type="Condition"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

condition

Type: Condition [197], complex content

XML Representation Summary

```
<condition>
    Content: logic, conditionRole
</condition>
```

complexType "Contains"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Contains operator returns true if the first operand contains the second.

There are two overloads of this operator:
List, T : The type of T must be the same as the element type of the list.
Interval, T : The type of T must be the same as the point type of the interval.

For the List, T overload, this operator returns true if the given element is in the list.

For the Interval, T overload, this operator returns true if the given point is greater than or equal to the beginning point of the interval, and less than or equal to the ending point of the interval. For open interval boundaries, exclusive comparison operators are used. For closed interval boundaries, if the interval boundary is null, the result of the boundary comparison is considered true.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Contains
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Contains">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Contains(i, p) = (if not(i.beginOpen) then p >= IsNull(i.begin, p) else p > i.begin) and (if
  not(i.endOpen) then p <= IsNull(i.end, p) else p < i.end) -->
</xs:complexType>
```

complexType "Contribution"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: contributor, role
</...>
```

Content Model Elements (2):

contributor [202], role [202]

Annotation

A contribution is made by a specific contributor (organization, person, etc.), and was made in a particular way, as specified by the contributor's role. For example, a contributor may have been an author, or may have been a reviewer.

XML Source (w/o annotations (1))

```
<xs:complexType name="Contribution">
  <xs:sequence>
    <xs:element name="contributor" type="Party"/>
    <xs:element name="role">
      <xs:complexType>
        <xs:attribute name="value" type="ContributorType"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

contributor

Type: Party [309], complex content

XML Representation Summary

```
<contributor>
    Content: addresses?, contacts?
</contributor>
```

role

Type: anonymous complexType, empty content

XML Representation Summary

```
<role
    value = (("Author" | "Editor" | "Endorser" | "Reviewer") | ("Author" | "Editor" | "Endorser" | "Reviewer"))
/>
```

complexType "Convert"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   toType = xs:QName
   >
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Convert operator converts a value to a specific type. The result of the operator is the value of the argument converted to the target type, if possible. Note that use of this operator may result in a run-time exception being thrown if there is no valid conversion from the actual value to the target type.

This operator supports conversion between String and each of Boolean, Integer, Real, and Timestamp, as well as conversion from Integer to Real.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ UnaryExpression [364] (extension)
      └ Convert
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Convert">
  <xs:complexContent>
    <xs:extension base="UnaryExpression">
      <xs:attribute name="toType" type="xs:QName" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

toType

Type: xs:QName, predefined
Use: required

complexType "Count"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Count operator returns the number of non-null elements in the source.

If a path is specified, the count returns the number of elements that have a value for the property specified by the path.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *AggregateExpression* [167] (extension)
  └ **Count**

XML Source (w/o annotations (1))

```
<xs:complexType name="Count">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Coverage"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
   Content: focus, description?, value?
</...>
```

Content Model Elements (3):

description (in coverage) [205], value (in coverage) [205]

focus [205],

Annotation

Specifies various attributes of the patient population for whom and/or environment of care in which the CDS artifact is applicable.

XML Source (w/o annotations (1))

```
<xs:complexType name="Coverage">
  <xs:sequence>
    <xs:element name="focus">
      <xs:complexType>
        <xs:attribute name="value" type="CoverageType" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="description" type="dt:ST"/>
    <xs:element minOccurs="0" name="value" type="dt:CD"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/3)

focus

Type: anonymous complexType, empty content

XML Representation Summary

```
<focus
    value = (("PatientGender" | "PatientAgeGroup" | "ClinicalFocus" | "TargetUser" | "WorkflowSetting" | "WorkflowTask"
            | "ClinicalVenue") | ("PatientGender" | "PatientAgeGroup" | "ClinicalFocus" | "TargetUser" |
            "WorkflowSetting" | "WorkflowTask" | "ClinicalVenue"))
/>
```

description

Type: dt:ST [124], empty content

XML Representation Summary

```
<description
    value = xs:string
/>
```

value

Type: dt:CD [85], complex content

XML Representation Summary

```
<value
    code              = xs:string
    codeSystem        = xs:string
    codeSystemName    = xs:string
    codeSystemVersion = xs:string
    valueSet          = xs:string
    valueSetVersion   = xs:string
    >
    Content: dt:displayName?, dt:originalText?, dt:translation*
</value>
```

complexType "CreateAction"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
             textEquivalent?, actionSentence?
</...>
```

Content Model Elements (8):

actionId [160],

actionSentence (defined in CreateAction complexType) [207],

actors [160],

behaviors (defined in *ActionBase* complexType) [160],

conditions (defined in *ActionBase* complexType) [160],

supportingEvidence (defined in *ActionBase* complexType) [160],

supportingResources [160],

textEquivalent [174]

Annotation

A new action to be executed by a user or a computer system. The sentence provides the details of the action to be executed.

Type Definition Detail

Type Derivation Tree

ActionBase [158] (extension)
└ *AtomicAction* [173] (extension)
  └ **CreateAction**

XML Source (w/o annotations (2))

```
<xs:complexType name="CreateAction">
  <xs:complexContent>
```

```
    <xs:extension base="AtomicAction">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="actionSentence" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/8)

actionSentence

Type: *Expression* [230], complex content

The parameters of the action that is to be
executed. For example, an action may be to order a medication.

XML Representation Summary

```
<actionSentence>
    Content: description?
</actionSentence>
```

complexType "Current"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    scope = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The Current expression returns the value of the object currently in scope. For example, within a ForEach expression, this returns the current element being considered in the iteration.

It is an error to invoke the Current operator outside the context of a scoped operation.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Current**

XML Source (w/o annotations (1))

```
<xs:complexType name="Current">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="scope" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- scope

  Type: xs:string, predefined
  Use: optional

complexType "DataRequest"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   scope        = xs:string
   cardinality  = ("Single" | "Multiple")
   dataType     = xs:QName
   templateId   = xs:string
   idProperty   = xs:string
   triggerType  = (("DataElementAdded" | "DataElementModified" | "DataElementRemoved" |
                  "DataElementAccessed" | "DataElementAccessEnded") | ("DataElementAdded" |
                  "DataElementModified" | "DataElementRemoved" | "DataElementAccessed" |
                  "DataElementAccessEnded"))
   isInitial    = xs:boolean : "true"
   >
   Content: description?, timeOffset?
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], timeOffset [336]

Annotation

The DataRequest expression provides basic data access.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ RequestBase [334] (extension)
      └ DataRequest
```

XML Source (w/o annotations (1))

```
<xs:complexType name="DataRequest">
  <xs:complexContent>
    <xs:extension base="RequestBase"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Date"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 7 elements

XML Representation Summary

```
<...>
   Content: description?, year, month, day, hour?, minute?, second?, millisecond?
</...>
```

Content Model Elements (8):

day [210],	minute [211],
description (defined in *Expression* complexType) [230],	month [210],
hour [211],	second [211],
millisecond [211],	year [210]

Annotation

The Date operator constructs a date/time value from the given components.

If any of year, month, or day is null, the result is null. The hour, minute, second, and millisecond may all be null, provided that no value appears in a granularity that is strictly smaller than a granularity that has already been provided. For example, hour may be non-null, and if minute, second, and millisecond are all null, they are assumed to be 0. However, if hour is null, minute, second, and millisecond must all be null as well.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Date**

XML Source (w/o annotations (1))

```
<xs:complexType name="Date">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="year" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="month" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="day" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="hour" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="minute" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="second" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="millisecond" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 7/8)

year

Type: *Expression* [230], complex content

XML Representation Summary

```
<year>
    Content: description?
</year>
```

month

Type: *Expression* [230], complex content

XML Representation Summary

```
<month>
    Content: description?
</month>
```

day

Type: *Expression* [230], complex content

XML Representation Summary

```
<day>
    Content: description?
</day>
```

hour

Type: *Expression* [230], complex content

XML Representation Summary

```
<hour>
    Content: description?
</hour>
```

minute

Type: *Expression* [230], complex content

XML Representation Summary

```
<minute>
    Content: description?
</minute>
```

second

Type: *Expression* [230], complex content

XML Representation Summary

```
<second>
    Content: description?
</second>
```

millisecond

Type: *Expression* [230], complex content

XML Representation Summary

```
<millisecond>
    Content: description?
</millisecond>
```

complexType "DateAdd"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
    Content: description?, date, granularity, numberOfPeriods
</...>
```

Content Model Elements (4):

date (defined in DateAdd complexType) [213], granularity (defined in DateAdd complexType) [213], description (defined in *Expression* complexType) [230], numberOfPeriods [213]

Annotation

The DateAdd operator adds numberOfPeriods date periods of the specified granularity to the given date.

Note that this is different than adding an Interval to a date time, because for operations on granularities such as month and year, the interval is not well-defined due to varying month and year lengths. As a result, DateAdd is used to provide well-defined and consistent semantics for date arithmetic involving months and years.

If any argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **DateAdd**

XML Source (w/o annotations (1))

```
<xs:complexType name="DateAdd">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="date" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="granularity" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="numberOfPeriods" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/4)

date

Type: *Expression* [230], complex content

XML Representation Summary

```
<date>
    Content: description?
</date>
```

granularity

Type: *Expression* [230], complex content

XML Representation Summary

```
<granularity>
    Content: description?
</granularity>
```

numberOfPeriods

Type: *Expression* [230], complex content

XML Representation Summary

```
<numberOfPeriods>
    Content: description?
</numberOfPeriods>
```

complexType "DateDiff"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
    Content: description?, startDate, endDate, granularity
</...>
```

Content Model Elements (4):

description (defined in *Expression* complexType) [230], granularity (defined in DateDiff complexType) [214], endDate [214], startDate [214]

Annotation

The DateDiff operator returns the number of granularity boundaries occurring between startDate and endDate.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **DateDiff**

XML Source (w/o annotations (1))

```
<xs:complexType name="DateDiff">
  <xs:complexContent>
```

```
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="startDate" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="endDate" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="granularity" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/4)

startDate

Type: *Expression* [230], complex content

XML Representation Summary

```
<startDate>
    Content: description?
</startDate>
```

endDate

Type: *Expression* [230], complex content

XML Representation Summary

```
<endDate>
    Content: description?
</endDate>
```

granularity

Type: *Expression* [230], complex content

XML Representation Summary

```
<granularity>
    Content: description?
</granularity>
```

complexType "DateOf"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in Expression complexType) [230], operand (defined in UnaryExpression complexType) [365]

Annotation

The DateOf operator returns the date (with no time component) of the argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ UnaryExpression [364] (extension)
      └ DateOf
```

XML Source (w/o annotations (1))

```
<xs:complexType name="DateOf">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "DatePart"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: description?, date, granularity
</...>
```

Content Model Elements (3):

date (defined in DatePart complexType) [216], granularity (defined in DatePart complexType) [216]

description (defined in Expression complexType) [230],

Annotation

The DatePart operator returns the granularity component of the given date.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **DatePart**

XML Source (w/o annotations (1))

```
<xs:complexType name="DatePart">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="date" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="granularity" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

date

Type: *Expression* [230], complex content

XML Representation Summary

```
<date>
    Content: description?
</date>
```

granularity

Type: *Expression* [230], complex content

XML Representation Summary

```
<granularity>
    Content: description?
</granularity>
```

complexType "DeclareResponseAction"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   name = xs:string : "Responses"
   >
   Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
            textEquivalent?
</...>
```

Content Model Elements (7):

actionId [160],
actors [160],
behaviors (defined in *ActionBase* complexType) [160],
conditions (defined in *ActionBase* complexType) [160],
supportingEvidence (defined in *ActionBase* complexType) [160],
supportingResources [160],
textEquivalent [174]

Annotation

The DeclareResponseAction provides a mechanism to declare a container for responses provided by the user in response to CollectInformationActions. The intended semantics are to provide a container that can be used to access responses within expressions used subsequently in an artifact.

The DeclareResponseAction creates a named container within the Parameters scope of the artifact, and expressions may access the contents of a response using a ParameterRef expression.

The container is expected to be a collection of name-value pairs, and the intended semantics are to allow the Property expression to be used, in connection with a ParameterRef expression as the source, to retrieve the current value for a property.

The CollectInformationAction contains a responseBinding attribute that specifies the name of the container, and the name of the property to be used to store the response value.

If no Name attribute is provided, the response container will be named Responses.

Type Definition Detail

Type Derivation Tree

```
ActionBase [158] (extension)
  └ AtomicAction [173] (extension)
      └ DeclareResponseAction
```

XML Source (w/o annotations (1))

```
<xs:complexType name="DeclareResponseAction">
  <xs:complexContent>
    <xs:extension base="AtomicAction">
      <xs:attribute default="Responses" name="name" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

name

Type: `xs:string`, predefined
Use: optional

Attribute Value

Default: `"Responses"`

complexType "Difference"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Difference operator returns the difference of the two arguments.

This operator has two overloads:
List, List
Interval, Interval

For the list overload, this operator returns a list with the elements that appear in the first operand, that do not appear in the second operand.

For the interval overload, this operator returns the portion of the first interval that does not overlap with the second. If the arguments do not overlap, or if the second argument is properly contained within the first, this operator returns null.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Difference
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Difference">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
```

```
</xs:complexType>
```

complexType "Distinct"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in Distinct complexType) [219]

Annotation

The Distinct operator takes a list of elements and returns a list containing only the unique elements within the input. For example, given the list of integers { 1, 1, 1, 2, 2, 3, 4, 4 }, the result of Distinct would be { 1, 2, 3, 4 }.

The operator uses equality comparison semantics as defined in the Equal operator.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **Distinct**

XML Source (w/o annotations (1))

```
<xs:complexType name="Distinct">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element name="source" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

complexType "Divide"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Divide operator performs numeric division of its arguments. Note that the result of Divide is a decimal, even if its arguments are integers.

If either argument is null, the result is null.

The Divide operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Divide
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Divide">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "DocumentationItem"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 6 elements

XML Representation Summary

```
<...>
   Content: identifier?, displayText, description?, itemCodes?, responseDataType, responseCardinality?,
            responseRange*
</...>
```

Content Model Elements (7):

description (in documentationConcept) [222],
displayText (in documentationConcept) [222],
identifier (defined in *ItemDefinition* complexType) [262],
itemCodes [222],
responseCardinality [223],
responseDataType [222],
responseRange [223]

Annotation

An item type representing the definition of an individual item to be recorded in a structured clinical document.

Type Definition Detail

Type Derivation Tree

ItemDefinition [262] (extension)

└ **DocumentationItem**

XML Source (w/o annotations (8))

```
<xs:complexType name="DocumentationItem">
  <xs:complexContent>
    <xs:extension base="ItemDefinition">
      <xs:sequence>
        <xs:element name="displayText" type="dt:ST"/>
        <xs:element maxOccurs="1" minOccurs="0" name="description" type="dt:ST"/>
        <xs:element maxOccurs="1" minOccurs="0" name="itemCodes">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" minOccurs="0" name="itemCode" type="dt:CD"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="responseDataType">
          <xs:complexType>
            <xs:attribute name="value" type="ValueType"/>
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="1" minOccurs="0" name="responseCardinality">
          <xs:complexType>
            <xs:attribute name="value" type="Cardinality"/>
```

```
            </xs:complexType>
          </xs:element>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="responseRange" type="RangeConstraint"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 6/7)

displayText

Type: dt:ST [124], empty content

Brief text or title for this documentation item that is the caption displayed to the user performing the documentation.

XML Representation Summary

```
<displayText
    value = xs:string
/>
```

description

Type: dt:ST [124], empty content

A lengthier description of this documentation item that is displayed to the user.

XML Representation Summary

```
<description
    value = xs:string
/>
```

itemCodes

Type: anonymous complexType, complex content

A collection of codes for concepts that are the equivalents of this documentation item.

XML Representation Summary

```
<itemCodes>
    Content: itemCode*
</itemCodes>
```

responseDataType

Type: anonymous complexType, empty content

The data type of the value or the response that is entered by the user.

XML Representation Summary

```
<responseDataType
    value = (("Boolean" | "Code" | "CodedOrdinal" | "EntityName" | "Identifier" | "Integer" | "IntegerInterval" |
            "Period" | "PhysicalQuantity" | "PhysicalQuantityInterval" | "QuantityInterval" | "Ratio" | "Real" |
            "RealInterval" | "SimpleCode" | "String" | "Timestamp" | "TimestampInterval" | "URL") | ("Boolean" | "Code"
            | "CodedOrdinal" | "EntityName" | "Identifier" | "Integer" | "IntegerInterval" | "Period" |
            "PhysicalQuantity" | "PhysicalQuantityInterval" | "QuantityInterval" | "Ratio" | "Real" | "RealInterval" |
            "SimpleCode" | "String" | "Timestamp" | "TimestampInterval" | "URL"))
/>
```

responseCardinality

Type: anonymous complexType, empty content

The number of allowed responses, single or multiple.

XML Representation Summary

```
<responseCardinality
    value = ("Single" | "Multiple")
/>
```

responseRange

Type: *RangeConstraint* [328], complex content

The constraints (within the responseDataType) on the values that may be entered by the user

XML Representation Summary

```
<responseRange>
    Content: constraintType
</responseRange>
```

complexType "End"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The End operator returns the ending point of an interval.

If the ending of the interval is open, this operator returns the Predecessor of the defined ending point of the interval.

If the ending of the interval is closed, this operator returns the defined ending point of the interval if it is not null.

Otherwise, it returns the maximum value of the point type of the interval.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **End**

XML Source (w/o annotations (1))

```
<xs:complexType name="End">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
  <!-- End(i) = if not(i.endOpen) then IfNull(i.end, T.MaximumValue()) else Pred(i.end) -->
</xs:complexType>
```

complexType "Ends"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Ends operator returns true if the first interval ends the second. In other words, if the starting point of the first interval is greater than or equal to the starting point of the second, and the ending point of the first interval is equal to the ending point of the second.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Ends**

XML Source (w/o annotations (1))

```
<xs:complexType name="Ends">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Ends(i1, i2) = Begin(i1) >= Begin(i2) and End(i1) = End(i2) -->
</xs:complexType>
```

complexType "EntityNameLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   use = list of ("ABC" | "IDE" | "SYL" | "C" | "OR" | "T" | "I" | "P" | "ANON" | "A" | "R" | "OLD" |
         "DN" | "M" | "PHON" | "SRCH")
   >
   Content: description?, part+
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], part (defined in EntityNameLiteral complexType) [226]

Annotation

The EntityNameLiteral expression returns a value of type EN with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **EntityNameLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="EntityNameLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
```

```
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="part" type="dt:ENXP"/>
        </xs:sequence>
        <xs:attribute name="use" type="dt:set_EntityNameUse" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

use

Type: dt:set_EntityNameUse [142]
Use: optional

Attribute Value

```
list of ("ABC" | "IDE" | "SYL" | "C" | "OR" | "T" | "I" | "P" | "ANON" | "A" | "R" | "OLD" | "DN" | "M" |
"PHON" | "SRCH")
```

Content Element Detail (all declarations; defined within this component only; 1/2)

part

Type: dt:ENXP [98], empty content

XML Representation Summary

```
<part
    qualifier = list of ("LS" | "AC" | "NB" | "PR" | "HON" | "BR" | "AD" | "SP" | "MID" | "CL" | "IN" | "PFX" | "SFX")
    type      = ("FAM" | "GIV" | "TITLE" | "DEL")
    value     = xs:string
/>
```

complexType "Equal"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Equal operator returns true if the arguments are equal; and false otherwise. Equality semantics are defined to be value-based.

For scalars this means that equality returns true if and only if the result of each argument evaluates to the same value.

For object types, this means that equality returns true if and only if the objects are of the same type, and the values for all properties are the same.

For list types, this means that equality returns true if and only if the lists contain elements of the same type, have the same number of elements, and for each element in the lists, in order, the elements are equal using the same semantics.

For interval types, equality returns true if and only if the intervals are over the same point type, and they have the same value for the beginning and ending points of the interval.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Equal**

XML Source (w/o annotations (1))

```
<xs:complexType name="Equal">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Evidence"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
  Content: qualityOfEvidence?, strengthOfRecommendation?, resources?
</...>
```

Content Model Elements (3):

qualityOfEvidence [228], strengthOfRecommendation [228]

resources (in evidence) [229],

Annotation

Reference to research on which the artifact is based. This evidence can be 'graded' depending on its quality and pedigree and the strength of the recommendations it makes.

XML Source (w/o annotations (4))

```
<xs:complexType name="Evidence">
  <xs:sequence>
```

```
    <xs:element maxOccurs="1" minOccurs="0" name="qualityOfEvidence" type="dt:CD"/>
    <xs:element maxOccurs="1" minOccurs="0" name="strengthOfRecommendation" type="dt:CD"/>
    <xs:element maxOccurs="1" minOccurs="0" name="resources" type="SupportingResource"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/3)

qualityOfEvidence

Type: dt:CD [85], complex content

The quality of the evidence associated with this artifact. The codeSystem attribute specifies the quality scale used to grade this evidence source while the code specifies the actual quality score (represented as a coded value) associated with this evidence reference. CodeSystemName specifies the name of the scale. DisplayName specifies the display name of the coded value (the score).

XML Representation Summary

```
<qualityOfEvidence
    code              = xs:string
    codeSystem        = xs:string
    codeSystemName    = xs:string
    codeSystemVersion = xs:string
    valueSet          = xs:string
    valueSetVersion   = xs:string
    >
    Content: dt:displayName?, dt:originalText?, dt:translation*
</qualityOfEvidence>
```

strengthOfRecommendation

Type: dt:CD [85], complex content

The strength of the recommendation assigned to this reference. The codeSystem attribute specifies the scale used to grade this evidence source while the code specifies the actual score (represented as a coded value) for the strength of the evidence. CodeSystemName specifies the name of the scale. DisplayName specifies the display name of the coded value (the score).

XML Representation Summary

```
<strengthOfRecommendation
    code              = xs:string
    codeSystem        = xs:string
    codeSystemName    = xs:string
    codeSystemVersion = xs:string
    valueSet          = xs:string
    valueSetVersion   = xs:string
    >
    Content: dt:displayName?, dt:originalText?, dt:translation*
</strengthOfRecommendation>
```

resources

Type: SupportingResource [355], complex content

The set of resource references associated with the evidence.

XML Representation Summary

```
<resources>
    Content: resource+
</resources>
```

complexType "Expand"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Expand operator flattens a list of lists into a single list.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
└ **Expand**

XML Source (w/o annotations (1))

```
<xs:complexType name="Expand">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Expression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The Expression type defines the abstract base type for all expressions used in the HeDS expression language.

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="Expression">
  <xs:sequence>
    <xs:element maxOccurs="1" minOccurs="0" name="description" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

description

Type: xs:string, predefined, simple content

XML Representation Summary

```
<description>
    Content:        { xs:string }
</description>
```

complexType "ExpressionConstraint"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: constraintType, constraint
</...>
```

Content Model Elements (2):

constraint [231], constraintType [329]

Annotation

A constraint specified in the form of an expression. The constraint type and the constraint expression combine together to specify the full constraint. For example, the constraint type specifies that the lower bound is being specified and the expression provides the lower bound value.

Type Definition Detail

Type Derivation Tree

RangeConstraint [328] (extension)

└ **ExpressionConstraint**

XML Source (w/o annotations (2))

```
<xs:complexType name="ExpressionConstraint">
  <xs:complexContent>
    <xs:extension base="RangeConstraint">
      <xs:sequence>
        <xs:element name="constraint" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

constraint

Type: *Expression* [230], complex content

XML Representation Summary

```
<constraint>
    Content: description?
</constraint>
```

complexType "ExpressionDef"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
    name = xs:string
    >
    Content: expression
</...>
```

Content Model Elements (1):

expression (defined in ExpressionDef complexType) [232]

Annotation

The ExpressionDef type defines an expression and an associated name that can be referenced by any expression in the artifact. The name must be unique within the artifact.

XML Source (w/o annotations (1))

```
<xs:complexType name="ExpressionDef">
  <xs:sequence>
    <xs:element name="expression" type="Expression"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string"/>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

name

Type: xs:string, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/1)

expression

Type: *Expression* [230], complex content

XML Representation Summary

```
<expression>
    Content: description?
</expression>
```

complexType "ExpressionRef"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    name = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The ExpressionRef type defines an expression that references a previously defined NamedExpression. The result of evaluating an ExpressionReference is the result of evaluating the referenced NamedExpression.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ExpressionRef**

XML Source (w/o annotations (1))

```
<xs:complexType name="ExpressionRef">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="name" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- name

  Type: xs:string, predefined
  Use: optional

complexType "Filter"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 2 elements

XML Representation Summary

```
<...
    scope = xs:string
    >
    Content: description?, source, condition
</...>
```

Content Model Elements (3):

condition (defined in Filter complexType) [235], source (defined in Filter complexType) [235]
description (defined in *Expression* complexType) [230],

Annotation

The Filter operator returns a list with only those elements in the source list for which the condition element evaluates to true.

If the source argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **Filter**

XML Source (w/o annotations (1))

```
<xs:complexType name="Filter">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="source" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="condition" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="scope" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

scope

Type: xs:string, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 2/3)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

condition

Type: *Expression* [230], complex content

XML Representation Summary

```
<condition>
    Content: description?
</condition>
```

complexType "FireEventAction"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
             textEquivalent?, eventType, actionSentence?
</...>
```

Content Model Elements (9):

actionId [160],

actionSentence (defined in FireEventAction complexType) [236],

actors [160],

behaviors (defined in *ActionBase* complexType) [160],

conditions (defined in *ActionBase* complexType) [160],

eventType (defined in FireEventAction complexType) [236],

supportingEvidence (defined in *ActionBase* complexType) [160],

supportingResources [160],

textEquivalent [174]

Annotation

This action fires an event. The event can serve as a trigger to another artifact.

Type Definition Detail

Type Derivation Tree

ActionBase [158] (extension)

└ AtomicAction [173] (extension)

└ **FireEventAction**

XML Source (w/o annotations (3))

```
<xs:complexType name="FireEventAction">
  <xs:complexContent>
    <xs:extension base="AtomicAction">
      <xs:sequence>
        <xs:element name="eventType" type="EventType"/>
        <xs:element maxOccurs="1" minOccurs="0" name="actionSentence" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/9)

eventType

Type: EventType [390], simple content

The type of the event that is fired.

XML Representation Summary

```
<eventType>
    Content:        { ("DataEvent" | "PeriodicEvent") | ("DataEvent" | "PeriodicEvent") }
</eventType>
```

actionSentence

Type: Expression [230], complex content

The action sentence is the payload of the
event. Another artifact receives this payload as an input.

XML Representation Summary

```
<actionSentence>
    Content: description?
</actionSentence>
```

complexType "First"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   orderBy = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in First complexType) [237]

Annotation

Returns the first element in a list. If the order by attribute is specified, the list is sorted by that ordering prior to returning the first element.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **First**

XML Source (w/o annotations (1))

```
<xs:complexType name="First">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element name="source" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="orderBy" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

orderBy

Type: xs:string, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

complexType "Floor"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Floor operator returns the first integer less than or equal to the argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ UnaryExpression [364] (extension)
      └ Floor
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Floor">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ForEach"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 2 elements

XML Representation Summary

```
<...
   scope = xs:string
   >
   Content: description?, source, element
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], source (defined in ForEach complexType) [240]

element (defined in ForEach complexType) [240],

Annotation

The ForEach expression iterates over the list of elements in the source element, and returns a list with the same number of elements, where each element in the new list is the result of evaluating the element expression for each element in the source list.

If the source argument is null, the result is null.

If the element argument evaluates to null for some item in the source list, the resulting list will contain a null for that element.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ForEach**

XML Source (w/o annotations (1))

```
<xs:complexType name="ForEach">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="source" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="element" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="scope" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- scope

  Type: xs:string, predefined
  Use: optional

Content Element Detail (all declarations; defined within this component only; 2/3)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

element

Type: *Expression* [230], complex content

XML Representation Summary

```
<element>
    Content: description?
</element>
```

complexType "Greater"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Greater operator returns true if the first argument is greater than the second argument.

If either argument is null, the result is null.

The Greater operator is defined for the Integer, Real, String, Timestamp, and PhysicalQuantity types.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Greater**

XML Source (w/o annotations (1))

```
<xs:complexType name="Greater">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "GreaterOrEqual"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The GreaterOrEqual operator returns true if the first argument is greater than or equal to the second argument.

If either argument is null, the result is null.

The GreaterOrEqual operator is defined for the Integer, Real, String, Timestamp, and PhysicalQuantity types.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **GreaterOrEqual**

XML Source (w/o annotations (1))

```
<xs:complexType name="GreaterOrEqual">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
```

```
  </xs:complexContent>
</xs:complexType>
```

complexType "GroupOrganizationBehavior"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = (("VisualGroup" | "LogicalGroup" | "SentenceGroup") | ("VisualGroup" | "LogicalGroup" |
           "SentenceGroup"))
/>
```

Annotation

For a group of actions, specifies the organizational intent of the grouping. This is meant to provide a hint to the system which displays the group of actions to an end user.

Type Definition Detail

Type Derivation Tree

Behavior [178] (restriction)

└ **GroupOrganizationBehavior**

XML Source (w/o annotations (1))

```
<xs:complexType name="GroupOrganizationBehavior">
  <xs:complexContent>
    <xs:restriction base="Behavior">
      <xs:attribute name="value" type="GroupOrganizationBehaviorType"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

Type: GroupOrganizationBehaviorType [392]
Use: optional

Attribute Value

```
("VisualGroup" | "LogicalGroup" | "SentenceGroup") | ("VisualGroup" | "LogicalGroup" | "SentenceGroup")
```

complexType "GroupSelectionBehavior"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = (("Any" | "All" | "AllOrNone" | "ExactlyOne" | "AtMostOne" | "OneOrMore") | ("Any" | "All"
            | "AllOrNone" | "ExactlyOne" | "AtMostOne" | "OneOrMore"))
/>
```

Annotation

For a group of actions, specifies the number of actions that may be chosen by an end user.

Type Definition Detail

Type Derivation Tree

Behavior [178] (restriction)

└ **GroupSelectionBehavior**

XML Source (w/o annotations (1))

```
<xs:complexType name="GroupSelectionBehavior">
  <xs:complexContent>
    <xs:restriction base="Behavior">
      <xs:attribute name="value" type="GroupSelectionBehaviorType"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

Type: GroupSelectionBehaviorType [394]
Use: optional

Attribute Value

```
("Any" | "All" | "AllOrNone" | "ExactlyOne" | "AtMostOne" | "OneOrMore") | ("Any" | "All" | "AllOrNone" |
"ExactlyOne" | "AtMostOne" | "OneOrMore")
```

complexType "IdentifierLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
   root      = xs:string
   extension = xs:string
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The IdentifierLiteral expression returns a value of type II with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **IdentifierLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="IdentifierLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="root" type="dt:Uid" use="required"/>
      <xs:attribute name="extension" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

root

Type: dt:Uid [147]
Use: required

Attribute Value

xs:string

extension

Type: xs:string, predefined
Use: optional

complexType "IfNull"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The IfNull operator replaces a null with the result of a given expression. If the first argument evaluates to null, returns the result of the second argument. Otherwise, returns the result of the first argument.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **IfNull**

XML Source (w/o annotations (1))

```
<xs:complexType name="IfNull">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "In"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The In operator tests for membership in an interval or list.

There are two overloads of this operator:
T, List : The type of T must be the same as the element type of the list.
T, Interval : The type of T must be the same as the point type of the interval.

For the T, List overload, this operator returns true if the given element is in the given list.

For the T, Interval overload, this operator returns true if the given point is greater than or equal to the beginning point of the interval, and less than or equal to the ending point of the interval. For open interval boundaries, exclusive comparison operators are used. For closed interval boundaries, if the interval boundary is null, the result of the boundary comparison is considered true.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ BinaryExpression [179] (extension)
  └ **In**

XML Source (w/o annotations (1))

```
<xs:complexType name="In">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- In(p, i) = (if not(i.beginOpen) then p >= IsNull(i.begin, p) else p > i.begin) and (if
  not(i.endOpen) then p <= IsNull(i.end, p) else p < i.end) -->
</xs:complexType>
```

complexType "IncludedIn"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The IncludedIn operator returns true if the first operand is completely included in the second.

There are two overloads of this operator:

List, List : The element type of both lists must be the same.
Interval, Interval : The point type of both intervals must be the same.

For the List, List overload, this operator returns true if every element in the first list is included in the second list.

For the Interval, Interval overload, this operator returns true if the beginning point of the first interval is greater than or equal to the beginning point of the second interval, and the ending point of the first interval is less than or equal to the ending point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ BinaryExpression [179] (extension)
  └ **IncludedIn**

XML Source (w/o annotations (1))

```
<xs:complexType name="IncludedIn">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- IncludedIn(i1, i2) = Begin(i1) >= Begin(i2) and End(i1) <= End(i2) -->
</xs:complexType>
```

complexType "Includes"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Includes operator returns true if the first operand completely includes the second.

There are two overloads of this operator:
List, List : The element type of both lists must be the same.
Interval, Interval : The point type of both intervals must be the same.

For the List, List overload, this operator returns true if the first operand includes every element of the second operand.

For the Interval, Interval overload, this operator returns true if beginning point of the first interval is less than or equal to the beginning point of the second interval, and the ending point of the first interval is greater than or equal to the ending point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Includes**

XML Source (w/o annotations (1))

```
<xs:complexType name="Includes">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Includes(i1, i2) = Begin(i1) <= Begin(i2) and End(i1) >= End(i2) -->
</xs:complexType>
```

complexType "Indexer"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: description?, operand, index
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], operand (defined in Indexer complexType) [249]

index [249],

Annotation

The Indexer operator returns the indexth element in a string or list.

Indexes in strings and lists are defined to be 1-based.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Indexer**

XML Source (w/o annotations (1))

```
<xs:complexType name="Indexer">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="operand" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="index" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

operand

Type: *Expression* [230], complex content

XML Representation Summary

```
<operand>
    Content: description?
</operand>
```

index

Type: *Expression* [230], complex content

XML Representation Summary

```
<index>
    Content: description?
</index>
```

complexType "IndexOf"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description?, source, element
</...>
```

Content Model Elements (3):

description (defined in Expression complexType) [230], source (defined in IndexOf complexType) [250]

element (defined in IndexOf complexType) [250],

Annotation

The IndexOf operator returns the 1-based index of the given element in the given source list.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **IndexOf**

XML Source (w/o annotations (1))

```
<xs:complexType name="IndexOf">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="source" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="element" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

source

Type: Expression [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

element

Type: Expression [230], complex content

XML Representation Summary

```
<element>
    Content: description?
</element>
```

complexType "InlineResource"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: identifiers?, templateIds?, title?, location?, description?, citation?, content?
</...>
```

Content Model Elements (7):

citation [267],

content [252],

description (defined in KnowledgeResource complexType) [267],

identifiers (defined in KnowledgeResource complexType) [267],

location [267],

templateIds (defined in KnowledgeResource complexType) [267],

title (defined in KnowledgeResource complexType) [267]

Annotation

An Inline Resource consists of both the resource reference information and the actual resource content/payload to be inserted inline. The content of the document must be represented in valid xhtml format within the content/div node.

Type Definition Detail

Type Derivation Tree

KnowledgeResource [265] (extension)

└ **InlineResource**

XML Source (w/o annotations (2))

```
<xs:complexType name="InlineResource">
  <xs:complexContent>
    <xs:extension base="KnowledgeResource">
      <xs:sequence>
        <xs:element minOccurs="0" name="content">
          <xs:complexType>
            <xs:sequence>
              <!--xs:element ref="xhtml:div"/-->
            </xs:sequence>
```

```
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/7)

content

Type: anonymous complexType, empty content

The document content in xhtml format.

XML Representation Summary

```
<content/>
```

complexType "IntegerIntervalLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 4 attributes

XML Representation Summary

```
<...
    low         = xs:int
    high        = xs:int
    lowClosed   = xs:boolean
    highClosed  = xs:boolean
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The IntegerIntervalLiteral expression returns a value of type IVL_INT with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **IntegerIntervalLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="IntegerIntervalLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="low" type="xs:int"/>
      <xs:attribute name="high" type="xs:int"/>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 4/4)

- low

  Type: xs:int, predefined
  Use: optional

- high

  Type: xs:int, predefined
  Use: optional

- lowClosed

  Type: xs:boolean, predefined
  Use: optional

- highClosed

  Type: xs:boolean, predefined
  Use: optional

complexType "IntegerLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = xs:int
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The IntegerLiteral expression returns a value of type INT with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **IntegerLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="IntegerLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="xs:int" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

  Type: xs:int, predefined
  Use: required

complexType "Intersect"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand*
</...>
```

Content Model Elements (2):

description (defined in Expression complexType) [230], operand (defined in NaryExpression complexType) [294]

Annotation

The Intersect operator returns the intersection of the operands.

This operator has two overloads:
List
Interval

For the list overload, this operator returns a list with the elements that appear in all the lists.

For the interval overload, this operator returns the interval that defines the overlapping portion of the arguments. If the arguments do not overlap, this operator returns null.

If any of the arguments are null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ NaryExpression [293] (extension)
      └ Intersect
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Intersect">
  <xs:complexContent>
    <xs:extension base="NaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Interval"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
   beginOpen  =  xs:boolean : "false"
   endOpen    =  xs:boolean : "false"
   >
   Content: description?, begin?, end?
</...>
```

Content Model Elements (3):

begin [257], end [257]

description (defined in Expression complexType) [230],

Annotation

The Interval selector defines an interval value. An interval must be defined using a point type that supports comparison, as well as Successor and Predecessor operations, and Minimum and Maximum Value operations.

The beginning and ending of the interval may each be defined as open or closed. Following standard terminology usage in interval mathematics, an open interval is defined to exclude the specified point, whereas a closed interval includes the point. The default is closed, indicating an inclusive interval.

The begin and end elements are both optional. If the begin element is not specified, the beginning point of the resulting interval is null. If the end element is not specified, the ending point of the resulting interval is null.

The static type of the Begin argument determines the type of the interval, and the End argument must be of the same type.

If the beginning point of the interval is null and the beginning of the interval is open, the beginning boundary of the interval is interpreted as unknown, and computations involving the beginning boundary will result in null.

If the beginning point of the interval is null and the beginning of the interval is closed, the interval is interpreted to begin at the start of the range of the point type, and computations involving the beginning point will be performed with that interpretation.

If the ending point of the interval is null and the ending of the interval is open, the ending boundary of the interval is unknown, and computations involving the ending point will result in null.

If the ending point of the interval is null and the ending of the interval is closed, the interval is interpreted to end at the end of the range of the point type, and computations involving the ending point will be performed with that interpretation.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Interval**

XML Source (w/o annotations (1))

```
<xs:complexType name="Interval">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="begin" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="end" type="Expression"/>
      </xs:sequence>
      <xs:attribute default="false" name="beginOpen" type="xs:boolean" use="optional"/>
      <xs:attribute default="false" name="endOpen" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

beginOpen

Type: `xs:boolean`, predefined
Use: optional

Attribute Value

Default: "false"

endOpen

Type: xs:boolean, predefined
Use: optional

Attribute Value

Default: "false"

Content Element Detail (all declarations; defined within this component only; 2/3)

begin

Type: *Expression* [230], complex content

XML Representation Summary

```
<begin>
    Content: description?
</begin>
```

end

Type: *Expression* [230], complex content

XML Representation Summary

```
<end>
    Content: description?
</end>
```

complexType "InValueSet"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 attributes

XML Representation Summary

```
<...
    id        = xs:string
    version   = xs:string
    authority = xs:string
    >
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The InValueSet operator returns true if the given code, or list of codes, is in the given value set.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **InValueSet**

XML Source (w/o annotations (4))

```
<xs:complexType name="InValueSet">
  <xs:complexContent>
    <xs:extension base="UnaryExpression">
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="version" type="xs:string" use="optional"/>
      <xs:attribute name="authority" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 3/3)

id

Type: xs:string, predefined
Use: required

The unique identifier of the value set to be retrieved. This identifier is evaluated within the context of the authority.

version

Type: xs:string, predefined
Use: optional

The version of the value set to be retrieved. If an asterisk (*) is used, the current version of the value set is returned.

authority

Type: xs:string, predefined
Use: optional

The authority for the value set to be retrieved. This value represents a conceptual identifier for the steward of the value set. An integration would need to be able to map this to a specific service providing the value sets maintained by this authority.

complexType "Is"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   isType = xs:QName
   >
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Is operator allows the type of a result to be tested. The language must support the ability to test against any type. If the run-time type of the argument is of the type being tested, the result of the operator is true; otherwise, the result is false.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
  └ **Is**

XML Source (w/o annotations (1))

```
<xs:complexType name="Is">
  <xs:complexContent>
    <xs:extension base="UnaryExpression">
      <xs:attribute name="isType" type="xs:QName" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- isType

  Type: xs:QName, predefined
  Use: required

complexType "IsEmpty"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The IsEmpty operator returns true if the list contains no elements.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
  └ **IsEmpty**

XML Source (w/o annotations (1))

```
<xs:complexType name="IsEmpty">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "IsNotEmpty"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The IsNotEmpty operator returns true if the list contains any elements.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **IsNotEmpty**

XML Source (w/o annotations (1))

```
<xs:complexType name="IsNotEmpty">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "IsNull"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The IsNull operator determines whether or not its argument evaluates to null. If the argument evaluates to null, the result is true; otherwise, the result is false.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **IsNull**

XML Source (w/o annotations (1))

```
<xs:complexType name="IsNull">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ItemDefinition"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: identifier?
</...>
```

Content Model Elements (1):

identifier (defined in *ItemDefinition* complexType) [262]

Annotation

An item definition is the equivalent of an item in a data dictionary or a catalog in an electronic health record system.

XML Source (w/o annotations (2))

```
<xs:complexType abstract="true" name="ItemDefinition">
  <xs:sequence>
    <xs:element minOccurs="0" name="identifier" type="VersionedIdentifier"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

identifier

Type: VersionedIdentifier [373], empty content

The identifier for the item.

XML Representation Summary

```
<identifier
    extension      = xs:string
    identifierName = xs:string
    root           = xs:string
    version        = xs:string
/>
```

complexType "KnowledgeDocument"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 7 elements

XML Representation Summary

```
<...>
   Content: metadata, externalData?, expressions?, triggers?, conditions?, behaviors?, actionGroup
</...>
```

Content Model Elements (7):

actionGroup (in knowledgeDocument) [265], externalData [264],
behaviors (in knowledgeDocument) [265], metadata [264],
conditions (in knowledgeDocument) [265], triggers [264]
expressions [264],

Annotation

A knowledge document is an instance of a CDS knowledge artifact such as a rule, an order set, or a documentation template

XML Source (w/o annotations (10))

```
<xs:complexType name="KnowledgeDocument">
  <xs:sequence>
    <xs:element maxOccurs="1" minOccurs="1" name="metadata" type="Metadata"/>
    <!-- Probably should constrain using schematron the expression type to
    Request -->
    <xs:element maxOccurs="1" minOccurs="0" name="externalData">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="parameter" type="ParameterDef"/>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="def" type="ExpressionDef"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0" name="expressions">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="def" type="ExpressionDef"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0" name="triggers" type="Triggers"/>
    <xs:element maxOccurs="1" minOccurs="0" name="conditions" type="Conditions"/>
    <xs:element maxOccurs="1" minOccurs="0" name="behaviors" type="Behaviors"/>
    <!-- actions -->
    <xs:element maxOccurs="1" minOccurs="1" name="actionGroup" type="ActionGroup"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 7/7)

metadata

Type: Metadata [281], complex content

The metadata section of the knowledge document defines the core metadata associated with this CDS knowledge artifact such as (1) the unique identifier for this artifact, (2) the unique identifier for its associated template(s), (3) the title and description of the artifact, (4) the status and history of the artifact, (5) any relevant entities associated with this artifact, and (6) information needed to categorize and retrieve the artifact.

XML Representation Summary

```
<metadata>
    Content: identifiers, artifactType, schemaIdentifier, templateIds?, dataModels?, title, description?, documentation?,
             relatedResources?, supportingEvidence?, applicability?, keyTerms?, categories?, language?, status, eventHistory?,
             contributions?, publishers?, usageTerms?
</metadata>
```

externalData

Type: anonymous complexType, complex content

The externalData section allows a CDS artifact author to define 'named expressions' to fetch information from an external source and bind this information to the 'context' of the knowledge artifact for later reference by the logic modules (e.g., the condition for the knowledge artifact or actions). It is the responsibility of the implementation to determine the nature of this boundary and how to fetch this information. For instance, one may write an expression to retrieve from a patient vMR the age of a patient or a list of clinical problems whose problem code are contained in a given ICD-9 value set. The age value and the list of patient problems may then be used in the 'condition' section of the same knowledge artifact to determine the applicability of the knowledge document to the given patient.

XML Representation Summary

```
<externalData>
    Content: parameter*, def*
</externalData>
```

expressions

Type: anonymous complexType, complex content

The expressions section allows a CDS artifact author to define 'named expressions' that can be referenced anywhere within expressions in the artifact. This allows expression logic to be reused, as well as to be organized for readability and maintainability.

XML Representation Summary

```
<expressions>
    Content: def*
</expressions>
```

triggers

Type: Triggers [362], complex content

The triggers section defines the list of all triggers that 'activate' or 'trigger' the CDS knowledge artifact. For instance, opening a patient record may trigger a rule to execute if the conditions of the rule are met.

XML Representation Summary

```
<triggers>
    Content: trigger+
</triggers>
```

conditions

Type: Conditions [200], complex content

The conditions section lists all conditions that pertain to the knowledge artifact. Conditions define the logic that determine the applicability of the artifact in the given context, any precondition or post condition, and/or any inclusion and exclusion criteria for the given CDS artifact. Conditions are structured as expressions to be evaluated in the target system.

XML Representation Summary

```
<conditions>
    Content: condition+
</conditions>
```

behaviors

Type: Behaviors [178], complex content

The behaviors section defines the set of behaviors for this knowledge document. While there are no artifact-level behaviors defined at this time, this element is included as a point of extension, should it be needed.

XML Representation Summary

```
<behaviors>
    Content: behavior+
</behaviors>
```

actionGroup

Type: ActionGroup [161], complex content

The actionGroups is essentially the top-level container for the groups of actions that make up a given knowledge document. This container defines the main content of the knowledge artifact: logical grouping constructs such as the clinical sections and orderables in an order set, the tasks to be performed by a rule, or the sections and menu choices that make up a document template.

XML Representation Summary

```
<actionGroup>
    Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?, title?, description?,
             representedConcepts?, subElements
</actionGroup>
```

complexType "KnowledgeResource"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 6 elements

XML Representation Summary

```
<...>
    Content: identifiers?, templateIds?, title?, location?, description?, citation?
</...>
```

Content Model Elements (6):

citation [267], location [267],

description (defined in KnowledgeResource templateIds (defined in KnowledgeResource

complexType) [267],

identifiers (defined in KnowledgeResource complexType) [267],

complexType) [267],

title (defined in KnowledgeResource complexType) [267]

Annotation

KnowledgeResource specifies a reference to an associated resource of relevance to the artifact such as a guideline, a performance measure, another knowledge artifact, or a source of evidence for the artifact.

XML Source (w/o annotations (9))

```
<xs:complexType name="KnowledgeResource">
  <xs:sequence>
    <xs:element minOccurs="0" name="identifiers">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="identifier" type="VersionedIdentifier"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="templateIds">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="templateId" type="dt:II"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="1" minOccurs="0" name="title" type="dt:ST"/>
    <xs:element maxOccurs="1" minOccurs="0" name="location" type="dt:TEL"/>
    <xs:element maxOccurs="1" minOccurs="0" name="description" type="dt:ST"/>
    <xs:element maxOccurs="1" minOccurs="0" name="citation" type="dt:ST"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 6/6)

identifiers

Type: anonymous complexType, complex content

The set of unique identifiers for this resource.

XML Representation Summary

```
<identifiers>
    Content: identifier
</identifiers>
```

templateIds

Type: anonymous complexType, complex content

The set of unique identifiers for the templates associated with this resource.

XML Representation Summary

```
<templateIds>
    Content: templateId
</templateIds>
```

title

Type: dt:ST [124], empty content

The title of the document

XML Representation Summary

```
<title
    value = xs:string
/>
```

location

Type: dt:TEL [124], empty content

The URL of the given resource.

XML Representation Summary

```
<location
    capabilities = list of ("voice" | "fax" | "data" | "tty" | "sms")
    use          = list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" | "MC" | "PG")
    value        = xs:anyURI
/>
```

description

Type: dt:ST [124], empty content

A short textual description of the resource.

XML Representation Summary

```
<description
    value = xs:string
/>
```

citation

Type: dt:ST [124], empty content

The resource citation.

XML Representation Summary

```
<citation
    value = xs:string
/>
```

complexType "Last"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
    orderBy = xs:string
    >
    Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in Last complexType) [269]

Annotation

The Last operator returns the last element in a list. If the order by attribute is specified, the list is sorted by that ordering prior to returning the last element.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **Last**

XML Source (w/o annotations (1))

```
<xs:complexType name="Last">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element name="source" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="orderBy" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

orderBy

Type: `xs:string`, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

complexType "Length"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Length operator returns the length of its argument.

For strings, the length is the number of characters in the string.

For intervals, the length is defined as the successor of the ending point minus the beginning point.

If the argument is null, or either the beginning or ending point of the interval is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
  └ **Length**

XML Source (w/o annotations (1))

```
<xs:complexType name="Length">
```

```
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Less"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Less operator returns true if the first argument is less than the second argument.

If either argument is null, the result is null.

The Less operator is defined for the Integer, Real, String, Timestamp, and PhysicalQuantity types.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Less
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Less">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "LessOrEqual"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The LessOrEqual operator returns true if the first argument is less than or equal to the second argument.

If either argument is null, the result is null.

The LessOrEqual operator is defined for the Integer, Real, String, Timestamp, and PhysicalQuantity types.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ LessOrEqual
```

XML Source (w/o annotations (1))

```
<xs:complexType name="LessOrEqual">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "List"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   key = xs:string
   >
   Content: description?, element*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], element (defined in List complexType) [272]

Annotation

The List selector returns a value of type List, whose elements are the result of evaluating the arguments to the List selector, in order. The static type of the first argument determines the type of the resulting list, and each subsequent argument must be of that same type.

If any argument is null, the resulting list will have null for that element.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **List**

XML Source (w/o annotations (1))

```
<xs:complexType name="List">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="element" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="key" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

key

Type: `xs:string`, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

element

Type: *Expression* [230], complex content

XML Representation Summary

```
<element>
    Content: description?
</element>
```

complexType "ListConstraint"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   strictSelection = xs:boolean
   >
   Content: constraintType, item+
</...>
```

Content Model Elements (2):

constraintType [329], item (defined in ListConstraint complexType) [274]

Annotation

A constraint specifying that the value is from a list included here.

Type Definition Detail

Type Derivation Tree

RangeConstraint [328] (extension)

└ **ListConstraint**

XML Source (w/o annotations (7))

```
<xs:complexType name="ListConstraint">
  <xs:complexContent>
    <xs:extension base="RangeConstraint">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="1" name="item">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="value" type="Expression"/>
              <xs:element minOccurs="0" name="codes">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="code" type="dt:CD"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element minOccurs="0" name="displayText" type="dt:ST"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="strictSelection" type="xs:boolean"/>
    </xs:extension>
  </xs:complexContent>
```

```
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

strictSelection

Type: `xs:boolean`, predefined
Use: optional

If strictSelection is set to true, the value
entered by the user must be
an item from the list. If this is set
to false, the value may not be restricted to this list.

Content Element Detail (all declarations; defined within this component only; 1/2)

item

Type: anonymous complexType, complex content

An item in the list of possible values

XML Representation Summary

```
<item>
    Content: value, codes?, displayText?
</item>
```

complexType "Literal"

Namespace: `urn:hl7-org:knowledgeartifact:r1`
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
    valueType  =  xs:QName
    value      =  xs:anySimpleType
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The Literal type defines a single scalar value. For example, the literal 5, the boolean value true or the string "antithrombotic".

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Literal**

XML Source (w/o annotations (1))

```
<xs:complexType name="Literal">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="valueType" type="xs:QName" use="required"/>
      <xs:attribute name="value" type="xs:anySimpleType" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

- valueType

  Type: xs:QName, predefined
  Use: required

- value

  Type: xs:anySimpleType, predefined
  Use: optional

complexType "Ln"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Ln operator computes the natural logarithm of its argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **Ln**

XML Source (w/o annotations (1))

```
<xs:complexType name="Ln">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Log"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Log operator computes the logarithm of its first argument, using the second argument as the base.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Log**

XML Source (w/o annotations (1))

```
<xs:complexType name="Log">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
```

```
</xs:complexType>
```

complexType "Lower"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Lower operator returns the lower case of its argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
  └ **Lower**

XML Source (w/o annotations (1))

```
<xs:complexType name="Lower">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Max"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Max operator returns the maximum element in the source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *AggregateExpression* [167] (extension)

└ **Max**

XML Source (w/o annotations (1))

```
<xs:complexType name="Max">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "MaxValue"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   valueType = xs:QName
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The MaxValue operator returns the maximum representable value for the given type.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **MaxValue**

XML Source (w/o annotations (1))

```
<xs:complexType name="MaxValue">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="valueType" type="xs:QName" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- valueType

  Type: xs:QName, predefined
  Use: required

complexType "Median"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Median operator returns the median of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *AggregateExpression* [167] (extension)

└ **Median**

XML Source (w/o annotations (1))

```
<xs:complexType name="Median">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Meets"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Meets operator returns true if the first interval ends on the starting point of the second, or if the first interval starts on the ending point of the second. In other words, if the ending point of the first interval is equal to the starting point of the second, or if the starting point of the first interval is equal to the ending point of the second.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Meets**

XML Source (w/o annotations (1))

```
<xs:complexType name="Meets">
```

```
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Meets(i1, i2) = End(i1) = Begin(i2) -->
</xs:complexType>
```

complexType "Metadata"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 19 elements

XML Representation Summary

```
<...>
  Content: identifiers, artifactType, schemaIdentifier, templateIds?, dataModels?, title, description?,
           documentation?, relatedResources?, supportingEvidence?, applicability?, keyTerms?,
           categories?, language?, status, eventHistory?, contributions?, publishers?, usageTerms?
</...>
```

Content Model Elements (19):

applicability [286],	language [286],
artifactType [284],	publishers [287],
categories [286],	relatedResources [285],
contributions [287],	schemaIdentifier [284],
dataModels [284],	status [287],
description (in metadata) [285],	supportingEvidence (in metadata) [286],
documentation [285],	templateIds (in metadata) [284],
eventHistory [287],	title (in metadata) [285],
identifiers (in metadata) [284],	usageTerms [288]
keyTerms [286],	

Annotation

The container for all of the metadata associated with a CDS knowledge artifact. Ideally, the metadata for artifacts is provided independently by the publisher for determining which artifact to retrieve.

XML Source (w/o annotations (17))

```
<xs:complexType name="Metadata">
  <xs:sequence>
    <xs:element name="identifiers">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="identifier" type="VersionedIdentifier"/>
        </xs:sequence>
      </xs:complexType>
```

```
</xs:element>
<xs:element name="artifactType">
  <xs:complexType>
    <xs:attribute name="value" type="ArtifactType" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="schemaIdentifier" type="VersionedIdentifier"/>
<xs:element minOccurs="0" name="templateIds">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="templateId" type="VersionedIdentifier"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="dataModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="modelReference" type="ModelReference"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="title" type="dt:ST"/>
<xs:element minOccurs="0" name="description" type="dt:ST"/>
<xs:element minOccurs="0" name="documentation" type="InlineResource"/>
<xs:element minOccurs="0" name="relatedResources">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="relatedResource" type="ResourceRelationshipReference"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="supportingEvidence" type="SupportingEvidence"/>
<xs:element minOccurs="0" name="applicability">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="coverage" type="Coverage"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="keyTerms">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="term" type="dt:CD"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
    <xs:element minOccurs="0" name="categories">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="category" type="dt:CD"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="language" type="dt:CD"/>
    <xs:element name="status">
      <xs:complexType>
        <xs:attribute name="value" type="ArtifactStatusType" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="eventHistory">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="artifactLifeCycleEvent" type="ArtifactLifeCycleEvent"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="contributions">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="contribution" type="Contribution"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="publishers">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="publisher" type="Party"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="usageTerms">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="rightsDeclaration" type="RightsDeclaration"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 19/19)

identifiers

Type: anonymous complexType, complex content

Each version of a CDS knowledge artifact may have more than one identifier associated with it. Note that each identifier must be globally unique in the universe of CDS knowledge artifacts in which a particular artifact resides.

XML Representation Summary

```
<identifiers>
    Content: identifier+
</identifiers>
```

artifactType

Type: anonymous complexType, empty content

Currently three types of artifacts are in scope for Health eDecisions Use Case #1: order sets, event-condition-action rules, and documentation templates. Additional types will be added in future revisions of the standard.

XML Representation Summary

```
<artifactType
    value = (("Rule" | "Order Set" | "Documentation Template") | ("Rule" | "Order Set" | "Documentation Template"))
/>
```

schemaIdentifier

Type: VersionedIdentifier [373], empty content

This is the identifier of the XML schema (and its version) which governs the structure of this CDS Knowledge Artifact.

XML Representation Summary

```
<schemaIdentifier
    extension      = xs:string
    identifierName = xs:string
    root           = xs:string
    version        = xs:string
/>
```

templateIds

Type: anonymous complexType, complex content

These are the identifiers of templates which
further constrain the structure of this knowledge artifact.

XML Representation Summary

```
<templateIds>
    Content: templateId+
</templateIds>
```

dataModels

Type: anonymous complexType, complex content

Set of data models referenced in the Expression objects in this knowledge artifact.

XML Representation Summary

```
<dataModels>
    Content: modelReference+
</dataModels>
```

title

Type: dt:ST [124], empty content

XML Representation Summary

```
<title
    value = xs:string
/>
```

description

Type: dt:ST [124], empty content

XML Representation Summary

```
<description
    value = xs:string
/>
```

documentation

Type: InlineResource [251], complex content

Documentation for this knowledge reference may consist of a reference to an external resource; the documentation may also be included in-line if desired.

XML Representation Summary

```
<documentation>
    Content: identifiers?, templateIds?, title?, location?, description?, citation?, content?
</documentation>
```

relatedResources

Type: anonymous complexType, complex content

A set of resources related to this artifact, along with an indication of the type of relationship. An artifact may be derived from or depend on other artifacts, along with other types of relationships. See the Artifact Lifecycle diagram in the Implementation Guide for more information.

XML Representation Summary

```
<relatedResources>
    Content: relatedResource+
</relatedResources>
```

supportingEvidence

Type: SupportingEvidence [354], complex content

XML Representation Summary

```
<supportingEvidence>
    Content: evidence+
</supportingEvidence>
```

applicability

Type: anonymous complexType, complex content

Specifies the conditions under which this artifact is applicable.

XML Representation Summary

```
<applicability>
    Content: coverage+
</applicability>
```

keyTerms

Type: anonymous complexType, complex content

Provides a list of coded key terms that pertain to this artifact.

XML Representation Summary

```
<keyTerms>
    Content: term+
</keyTerms>
```

categories

Type: anonymous complexType, complex content

Provides a list of coded categories to which this artifact belongs.

XML Representation Summary

```
<categories>
    Content: category+
</categories>
```

language

Type: dt:CD [85], complex content

XML Representation Summary

```
<language
    code              = xs:string
    codeSystem        = xs:string
    codeSystemName    = xs:string
    codeSystemVersion = xs:string
    valueSet          = xs:string
    valueSetVersion   = xs:string
    >
    Content: dt:displayName?, dt:originalText?, dt:translation*
</language>
```

status

Type: anonymous complexType, empty content

XML Representation Summary

```
<status
    value = (("Draft" | "InTest" | "Active" | "Inactive") | ("Draft" | "InTest" | "Active" | "Inactive"))
/>
```

eventHistory

Type: anonymous complexType, complex content

This is the history of events which have occurred for this particular version of the artifact.

XML Representation Summary

```
<eventHistory>
    Content: artifactLifeCycleEvent+
</eventHistory>
```

contributions

Type: anonymous complexType, complex content

Includes a list of people and/or organizations who have contributed to the development of this artifact. Contributions are not necessarily tied to specific versions of the artifact.

XML Representation Summary

```
<contributions>
    Content: contribution+
</contributions>
```

publishers

Type: anonymous complexType, complex content

The set of people and/or organizations who publish the artifact.

XML Representation Summary

```
<publishers>
    Content: publisher+
</publishers>
```

usageTerms

Type: anonymous complexType, complex content

This is the set of rights reserved by the person
or organization holding the rights to this artifact, along with
the set of permissions granted to consumers.

XML Representation Summary

```
<usageTerms>
    Content: rightsDeclaration+
</usageTerms>
```

complexType "Min"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
    path = xs:string
    >
    Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Min operator returns the minimum element in the source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *AggregateExpression* [167] (extension)

└ **Min**

XML Source (w/o annotations (1))

```
<xs:complexType name="Min">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "MinValue"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
  valueType = xs:QName
  >
  Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The MinValue operator returns the minimum representable value for the given type.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **MinValue**

XML Source (w/o annotations (1))

```
<xs:complexType name="MinValue">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="valueType" type="xs:QName" use="required"/>
```

```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- valueType

  Type: `xs:QName`, predefined
  Use: required

complexType "Mode"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Mode operator returns the statistical mode of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ AggregateExpression [167] (extension)
      └ Mode
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Mode">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ModelReference"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description, referencedModel
</...>
```

Content Model Elements (2):

description (in modelReference) [291], referencedModel [291]

Annotation

A reference to some model by its Universal Resource Identifier.

XML Source (w/o annotations (4))

```
<xs:complexType name="ModelReference">
  <xs:sequence>
    <xs:element name="description" type="dt:ST"/>
    <xs:element name="referencedModel">
      <xs:complexType>
        <xs:attribute name="value" type="xs:anyURI"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

description

Type: dt:ST [124], empty content

The object is assigned a name that can be used in this artifact.

XML Representation Summary

```
<description
    value = xs:string
/>
```

referencedModel

Type: anonymous complexType, empty content

The identifier of the object that is being referenced.

XML Representation Summary

```
<referencedModel
    value = xs:anyURI
/>
```

complexType "Modulo"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Modulo operator computes the remainder of the division of its arguments.

If either argument is null, the result is null.

The Modulo operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Modulo
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Modulo">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Multiply"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Multiply operator performs numeric multiplication of its arguments.

If either argument is null, the result is null.

The Multiply operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Multiply
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Multiply">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "NaryExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 element

XML Representation Summary

```
<...>
   Content: description?, operand*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *NaryExpression* complexType) [294]

Annotation

The NaryExpression type defines an abstract base class for an expression that takes any number of arguments, including zero.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ ***NaryExpression***

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="NaryExpression">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="operand" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

operand

Type: *Expression* [230], complex content

XML Representation Summary

```
<operand>
    Content: description?
</operand>
```

complexType "Negate"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Negate operator returns the negative of its argument.

If the argument is null, the result is null.

The Negate operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **Negate**

XML Source (w/o annotations (1))

```
<xs:complexType name="Negate">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Not"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Not operator returns the logical negation of its argument. If the argument is true, the result is false; if the argument is false, the result is true; otherwise, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **Not**

XML Source (w/o annotations (1))

```
<xs:complexType name="Not">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "NotEqual"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The NotEqual operator returns true if its arguments are not the same value.

The NotEqual operator is a shorthand for invocation of logical negation of the Equal operator.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **NotEqual**

XML Source (w/o annotations (1))

```
<xs:complexType name="NotEqual">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
```

```
</xs:complexType>
```

complexType "Now"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

Returns the date and time of the start timestamp associated with the evaluation request. Now is defined in this way for two reasons:
1) The operation will always return the same value within any given evaluation, ensuring that the result of an expression containing Now will always return the same result.

2) The operation will return the timestamp associated with the evaluation request, allowing the evaluation to be performed with the same timezone information as the data delivered with the evaluation request.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Now**

XML Source (w/o annotations (1))

```
<xs:complexType name="Now">
  <xs:complexContent>
    <xs:extension base="Expression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Null"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    valueType = xs:QName
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The Null operator returns a null, or missing information marker. To avoid the need to cast this result, the operator is defined to return a typed null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Null**

XML Source (w/o annotations (1))

```
<xs:complexType name="Null">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="valueType" type="xs:QName" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

valueType

Type: xs:QName, predefined
Use: required

complexType "ObjectDescriptor"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
    objectType = xs:QName
    >
    Content: description?, property*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], property (defined in ObjectDescriptor complexType) [299]

Annotation

The ObjectDescriptor type allows arbitrary object descriptors to be built, allowing for the creation of a value that has enough information to uniquely identify an object, but is not the object itself. Object descriptors are used as part of the update and remove action sentences to allow the object to be updated to be described without creating the object itself.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ObjectDescriptor**

XML Source (w/o annotations (1))

```
<xs:complexType name="ObjectDescriptor">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="property" type="PropertyExpression"/>
      </xs:sequence>
      <xs:attribute name="objectType" type="xs:QName"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

objectType

Type: xs:QName, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

property

Type: PropertyExpression [326], complex content

XML Representation Summary

```
<property
    name = xs:string
    >
    Content: value
</property>
```

complexType "ObjectExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
    objectType = xs:QName
    >
    Content: description?, property*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], property (defined in ObjectExpression complexType) [301]

Annotation

The ObjectExpression type allows objects of any type to be built up as an expression. The objectType attribute specifies the type of the object being built, and the list of property elements specify the values for the properties of the object. Note that the value of a property may be any expression, including another ObjectExpression.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ObjectExpression**

XML Source (w/o annotations (1))

```
<xs:complexType name="ObjectExpression">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="property" type="PropertyExpression"/>
      </xs:sequence>
      <xs:attribute name="objectType" type="xs:QName"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

objectType

Type: `xs:QName`, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

property

Type: `PropertyExpression [326]`, complex content

XML Representation Summary

```
<property
    name = xs:string
    >
    Content: value
</property>
```

complexType "ObjectRedefine"

Namespace: `urn:hl7-org:knowledgeartifact:r1`
Includes: definitions of 1 attribute, 2 elements

XML Representation Summary

```
<...
    scope = xs:string
    >
    Content: description?, source?, property+
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230],

property (in actionSentence defined in UpdateAction complexType) [302],

source (in actionSentence defined in UpdateAction complexType) [302]

Annotation

The ObjectRedefine expression returns an object of the same type as the source argument, with the same values for each property, except the values of the properties listed in the property elements. For those properties, the values will be set based on the expressions in those elements. This operator allows an object to be "copied" with new values specified only for a given list of properties.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ObjectRedefine**

XML Source (w/o annotations (1))

```
<xs:complexType name="ObjectRedefine">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="source" type="Expression"/>
        <xs:element maxOccurs="unbounded" minOccurs="1" name="property" type="PropertyExpression"/>
      </xs:sequence>
      <xs:attribute name="scope" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

scope

Type: xs:string, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 2/3)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

property

Type: PropertyExpression [326], complex content

XML Representation Summary

```
<property
    name = xs:string
    >
    Content: value
</property>
```

complexType "Or"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

<...>

Content: description?, operand*

</...>

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *NaryExpression* complexType) [294]

Annotation

The Or operator returns the logical disjunction of its arguments. Note that this operator is defined as n-ary, allowing any number of arguments. The result of Or with no arguments is defined to be true. The result of Or with a single argument is defined to be the result of the argument. The result of Or with two arguments is defined using 3-valued logic semantics. This means that if either argument is true, the result is true; if both arguments are false, the result is false; otherwise, the result is null. The result of more than two arguments is defined as successive invocations of Or.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *NaryExpression* [293] (extension)

└ **Or**

XML Source (w/o annotations (1))

```
<xs:complexType name="Or">
  <xs:complexContent>
    <xs:extension base="NaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Organization"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

<...>

Content: addresses?, contacts?, name

</...>

Content Model Elements (3):

addresses [310], name (in affiliation) [304]

contacts [310],

Annotation

Identifies an organization, a corporation, an institution, or a government department that has relevance to the knowledge artifact. Note that organization extends Party by adding a name attribute of type ST.

Type Definition Detail

Type Derivation Tree

Party [309] (extension)

└ **Organization**

XML Source (w/o annotations (2))

```
<xs:complexType name="Organization">
  <xs:complexContent>
    <xs:extension base="Party">
      <xs:sequence>
        <xs:element name="name" type="dt:ST"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/3)

name

Type: dt:ST [124], empty content

The name of the organization.

XML Representation Summary

```
<name
    value = xs:string
/>
```

complexType "Overlaps"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Overlaps operator returns true if the first interval overlaps the second. In other words, if the ending point of the first interval is greater than or equal to the starting point of the second interval, and the starting point of the first interval is less than or equal to the ending point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ Overlaps
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Overlaps">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- Overlaps(i1, i2) = End(i1) >= Begin(i2) and Begin(i1) <= End(i2) -->
</xs:complexType>
```

complexType "OverlapsAfter"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The OverlapsAfter operator returns true if the first interval overlaps and ends after the second. In other words, if the starting point of the first interval contains the ending point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **OverlapsAfter**

XML Source (w/o annotations (1))

```
<xs:complexType name="OverlapsAfter">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- OverlapsAfter(i1, i2) = Contains(i1, End(i2)) -->
</xs:complexType>
```

complexType "OverlapsBefore"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
  Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The OverlapsBefore operator returns true if the first interval starts before and overlaps the second. In other words, if the first interval contains the starting point of the second interval.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **OverlapsBefore**

XML Source (w/o annotations (1))

```
<xs:complexType name="OverlapsBefore">
```

```
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- OverlapsBefore(i1, i2) = Contains(i1, Begin(i2)) -->
</xs:complexType>
```

complexType "ParameterDef"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 1 element

XML Representation Summary

```
<...
   name          = xs:string
   parameterType = xs:QName
   >
   Content: default?
</...>
```

Content Model Elements (1):

default [308]

Annotation

The ParameterDef type defines a parameter that can be referenced by name anywhere within an expression. Parameters are defined at the artifact level, and may be provided as part of the payload for an evaluation request. If no parameter value is provided, the default element is used to provide the value for the parameter. If no parameter or default is provided, the parameter is defined to be null.

XML Source (w/o annotations (1))

```
<xs:complexType name="ParameterDef">
  <xs:sequence>
    <xs:element maxOccurs="1" minOccurs="0" name="default" type="Expression"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="parameterType" type="xs:QName"/>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

name

Type: xs:string, predefined
Use: optional

parameterType

Type: xs:QName, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/1)

default

Type: Expression [230], complex content

XML Representation Summary

```
<default>
    Content: description?
</default>
```

complexType "ParameterRef"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    name = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in Expression complexType) [230]

Annotation

The ParameterRef expression allows the value of a parameter to be referenced as part of an expression.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **ParameterRef**

XML Source (w/o annotations (1))

```
<xs:complexType name="ParameterRef">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="name" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

name

Type: xs:string, predefined
Use: optional

complexType "Party"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
   Content: addresses?, contacts?
</...>
```

Content Model Elements (2):

addresses [310], contacts [310]

Annotation

Party represents the abstract base type for entities that have addresses and contact information. It is intended to be extended and specialized by the Person and Organization concept. Note that Party allows for polymorphism using the xsi:type construct. For instance, by defining a 'contributor' to be of type 'Party', one allows the contributor to be either a person or a company (its derived types).

XML Source (w/o annotations (5))

```
<xs:complexType abstract="true" name="Party">
  <xs:sequence>
    <xs:element minOccurs="0" name="addresses">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="address" type="dt:AD"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" name="contacts">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="contact" type="dt:TEL"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

addresses

Type: anonymous complexType, complex content

The set of addresses associated with this entity.

XML Representation Summary

```
<addresses>
    Content: address+
</addresses>
```

contacts

Type: anonymous complexType, complex content

The set of contact information associated with this entity.

XML Representation Summary

```
<contacts>
    Content: contact+
</contacts>
```

complexType "PeriodLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 4 elements

XML Representation Summary

```
<...
    alignment   = ("CY" | "MY" | "CM" | "CW" | "WM" | "WY" | "DM" | "CD" | "DY" | "DW" | "HD" | "CH" |
                  "NH" | "CN" | "SN" | "CS")
    isFlexible  = xs:boolean
    >
    Content: description?, phase?, period?, frequency?, count?
</...>
```

Content Model Elements (5):

count [313], period [312], description (defined in *Expression* complexType) [230], phase [312], frequency [312],

Annotation

The PeriodLiteral expression returns a value of type PIVL_TS with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **PeriodLiteral**

XML Source (w/o annotations (6))

```
<xs:complexType name="PeriodLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element minOccurs="0" name="phase" type="dt:IVL_TS"/>
        <xs:element minOccurs="0" name="period" type="dt:PQ"/>
        <xs:element minOccurs="0" name="frequency" type="dt:RTO"/>
        <xs:element minOccurs="0" name="count" type="dt:INT"/>
      </xs:sequence>
      <xs:attribute name="alignment" type="dt:CalendarCycle" use="optional"/>
      <xs:attribute name="isFlexible" type="xs:boolean" use="optional"/>
    </xs:extension>
    <!-- changed from bl to xs:boolean since bl was removed as a type
    Aziz Boxwala
    -->
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

alignment

Type: dt:CalendarCycle [131]
Use: optional

Attribute Value

enumeration of xs:string

Enumeration:

"CY" – year
"MY" – month of the year
"CM" – month (continuous)
"CW" – week (continuous)
"WM" – week of the month
"WY" – week of the year
"DM" – day of the month
"CD" – day (continuous)
"DY" – day of the year
"DW" – day of the week (begins with monday)
"HD" – hour of the day
"CH" – hour (continuous)
"NH" – minute of the hour
"CN" – minute (continuous)
"SN" – second of the minute
"CS" – second (continuous)

isFlexible

Type: `xs:boolean`, predefined
Use: optional

Indicates whether the exact timing is up to the party executing the schedule e.g., to distinguish "every 8 hours" from "3 times a day".

Note: this is sometimes referred to as "institution specified timing".

Content Element Detail (all declarations; defined within this component only; 4/5)

phase

Type: `dt:IVL_TS [113]`, complex content

A prototype of the repeating interval, specifying the duration of each occurrence and anchors the PIVL sequence at a certain point in time. phase also marks the anchor point in time for the entire series of periodically recurring intervals. If count is null or nullFlavored, the recurrence of a PIVL has no beginning or ending, but is infinite in both future and past.

The width of the phase SHALL be less than or equal to the period

XML Representation Summary

```
<phase
    highClosed = xs:boolean
    lowClosed  = xs:boolean
    >
    Content: dt:low?, dt:high?
</phase>
```

period

Type: `dt:PQ [118]`, empty content

A time duration specified as a reciprocal measure of the frequency at which the PIVL repeats.

XML Representation Summary

```
<period
    unit  = xs:string
    value = xs:double
/>
```

frequency

Type: `dt:RTO [122]`, complex content

The number of times the PIVL repeats (numerator) within a specified time-period (denominator). The numerator is an integer, and the denominator is a PQ.TIME.

Only one of period and frequency should be specified. The form chosen should be the form that most naturally conveys the idea to humans. i.e. Every 10 mins (period) or twice a day (frequency).

XML Representation Summary

```
<frequency>
    Content: dt:numerator, dt:denominator
</frequency>
```

count

Type: dt:INT [102], empty content

The number of times the period repeats in total. If count is null, then the period repeats indefinitely both before and after the anchor implicit in the phase.

XML Representation Summary

```
<count
    value = xs:integer
/>
```

complexType "Person"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: addresses?, contacts?, name, affiliation?
</...>
```

Content Model Elements (4):

addresses [310], contacts [310],
affiliation [314], name (defined in Person complexType) [314]

Annotation

Identifies a person who is associated with the knowledge artifact. A person may be a contributor, a rights holder, a publisher, and so on. Person extends party by adding a person name attribute and an affiliation. Note, Person.name should be constrained to be of type EN.PN

Type Definition Detail

Type Derivation Tree

Party [309] (extension)
└ **Person**

XML Source (w/o annotations (3))

```
<xs:complexType name="Person">
  <xs:complexContent>
    <xs:extension base="Party">
      <xs:sequence>
        <xs:element name="name" type="dt:EN"/>
        <xs:element minOccurs="0" name="affiliation" type="Organization"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```

```
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/4)

name

Type: dt:EN [97], complex content

The name of the person.

XML Representation Summary

```
<name
    use = list of ("ABC" | "IDE" | "SYL" | "C" | "OR" | "T" | "I" | "P" | "ANON" | "A" | "R" | "OLD" | "DN" | "M" |
          "PHON" | "SRCH")
    >
    Content: dt:part+
</name>
```

affiliation

Type: Organization [303], complex content

The organizational affiliation for this person.

XML Representation Summary

```
<affiliation>
    Content: addresses?, contacts?, name
</affiliation>
```

complexType "PhysicalQuantityIntervalLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
    lowClosed   = xs:boolean
    highClosed  = xs:boolean
    >
    Content: description?, low?, high?
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230],
high (defined in PhysicalQuantityIntervalLiteral complexType) [315],
low (defined in PhysicalQuantityIntervalLiteral complexType) [315]

Annotation

The PhysicalQuantityIntervalLiteral expression returns a value of type IVL_PQ with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **PhysicalQuantityIntervalLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="PhysicalQuantityIntervalLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="low" type="dt:PQ"/>
        <xs:element maxOccurs="1" minOccurs="0" name="high" type="dt:PQ"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: xs:boolean, predefined
Use: optional

highClosed

Type: xs:boolean, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 2/3)

low

Type: dt:PQ [118], empty content

XML Representation Summary

```
<low
    unit  = xs:string
    value = xs:double
/>
```

high

Type: dt:PQ [118], empty content

XML Representation Summary

```
<high
    unit  = xs:string
    value = xs:double
/>
```

complexType "PhysicalQuantityLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
    value = xs:double
    unit  = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The PhysicalQuantityLiteral expression returns a value of type PQ with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **PhysicalQuantityLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="PhysicalQuantityLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="dt:Decimal" use="required"/>
      <xs:attribute name="unit" type="dt:Code" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

value

Type: dt:Decimal [134]
Use: required

Attribute Value

xs:double

unit

Type: dt:Code [132]
Use: required

Attribute Value

xs:string

complexType "PopulationStdDev"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The PopulationStdDev operator returns the statistical standard deviation of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *AggregateExpression* [167] (extension)

└ **PopulationStdDev**

XML Source (w/o annotations (1))

```
<xs:complexType name="PopulationStdDev">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "PopulationVariance"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
   path = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The PopulationVariance operator returns the statistical population variance of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *AggregateExpression* [167] (extension)

└ **PopulationVariance**

XML Source (w/o annotations (1))

```
<xs:complexType name="PopulationVariance">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Pos"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: description?, pattern, string
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], string [320]

pattern [320],

Annotation

The Pos operator returns the 1-based index of the given pattern in the given string.

If the pattern is not found, the result is 0.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Pos**

XML Source (w/o annotations (1))

```
<xs:complexType name="Pos">
  <xs:complexContent>
```

```
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="pattern" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="1" name="string" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

pattern

Type: *Expression* [230], complex content

XML Representation Summary

```
<pattern>
    Content: description?
</pattern>
```

string

Type: *Expression* [230], complex content

XML Representation Summary

```
<string>
    Content: description?
</string>
```

complexType "Power"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Power operator raises the first argument to the power given by the second argument.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **Power**

XML Source (w/o annotations (1))

```
<xs:complexType name="Power">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "PrecheckBehavior"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
  value = (("Yes" | "No") | ("Yes" | "No"))
/>
```

Annotation

For a particular action, specifies how often the action is expected to be selected in the particular context of the group containing that action. In general, depending on the group selection behavior, there may be zero, one or more actions which are frequently selected. This setting can serve as a hint to the system that displays the action to the end user: some systems will pre-select those actions which are (or should be) most frequently selected.

Type Definition Detail

Type Derivation Tree

Behavior [178] (restriction)

└ **PrecheckBehavior**

XML Source (w/o annotations (1))

```
<xs:complexType name="PrecheckBehavior">
  <xs:complexContent>
    <xs:restriction base="Behavior">
      <xs:attribute name="value" type="PrecheckBehaviorType"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

Type: PrecheckBehaviorType [396]
Use: optional

Attribute Value

```
("Yes" | "No") | ("Yes" | "No")
```

complexType "Pred"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Pred operator returns the predecessor of the argument. For example, the predecessor of 2 is 1.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
└ **Pred**

XML Source (w/o annotations (1))

```
<xs:complexType name="Pred">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "ProperIncludedIn"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The ProperIncludedIn operator returns true if the first operand is included in the second, and is strictly smaller.

There are two overloads of this operator:
List, List : The element type of both lists must be the same.
Interval, Interval : The point type of both intervals must be the same.

For the List, List overload, this operator returns true if every element of the first list is included in the second list, and the first list is strictly smaller.

For the Interval, Interval overload, this operator returns true if the first interval is included in the second interval, and the intervals are not equal.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ ProperIncludedIn
```

XML Source (w/o annotations (1))

```
<xs:complexType name="ProperIncludedIn">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- ProperIncludedIn(i1, i2) = IncludedIn(i1, i2) and i1 <> i2 -->
</xs:complexType>
```

complexType "ProperIncludes"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The ProperIncludes operator returns true if the first operand includes the second, and is strictly larger.

There are two overloads of this operator:
List, List : The element type of both lists must be the same.
Interval, Interval : The point type of both intervals must be the same.

For the List, List overload, this operator returns true if the first list includes every element of the second list, and first list is strictly larger.

For the Interval, Interval overload, this operator returns true if the first interval includes the second interval, and the intervals are not equal.

This operator uses the semantics described in the Begin and End operators to determine interval boundaries.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ BinaryExpression [179] (extension)
      └ ProperIncludes
```

XML Source (w/o annotations (1))

```
<xs:complexType name="ProperIncludes">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
  <!-- ProperIncludes(i1, i2) = Includes(i1, i2) and i1 <> i2 -->
</xs:complexType>
```

complexType "Property"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 1 element

XML Representation Summary

```
<...
   path   =  xs:string
   scope  =  xs:string
   >
   Content: description?, source?
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in Property complexType) [326]

Annotation

The Property operator returns the value of the property on source specified by the path attribute.

If the path attribute contains qualifiers, each qualifier is traversed to obtain the actual value.

If a scope is specified, the name is used to resolve the scope in which the path will be resolved. Scopes can be named by operators such as Filter and ForEach.

Property expressions can also be used to access the individual points and open indicators for interval types using the property names begin, end, beginOpen, and endOpen.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Property**

XML Source (w/o annotations (1))

```
<xs:complexType name="Property">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="source" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="path" type="xs:string" use="required"/>
      <xs:attribute name="scope" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

path

Type: xs:string, predefined
Use: required

scope

Type: xs:string, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

source

Type: Expression [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

complexType "PropertyExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
    name = xs:string
    >
    Content: value
</...>
```

Content Model Elements (1):

value (defined in PropertyExpression complexType) [327]

Annotation

The PropertyExpression type is used within the ObjectLiteral type to provide the value of a specific property within an object literal expression.

XML Source (w/o annotations (1))

```
<xs:complexType name="PropertyExpression">
  <xs:sequence>
    <xs:element maxOccurs="1" minOccurs="1" name="value" type="Expression"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

name

Type: xs:string, predefined

Use: required

Content Element Detail (all declarations; defined within this component only; 1/1)

value

Type: *Expression* [230], complex content

XML Representation Summary

```
<value>
    Content: description?
</value>
```

complexType "QuantityIntervalLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
    lowClosed  = xs:boolean
    highClosed = xs:boolean
    >
    Content: description?, low?, high?
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], low (defined in QuantityIntervalLiteral complexType) [328]
high (defined in QuantityIntervalLiteral complexType) [328],

Annotation

The QuantityIntervalLiteral expression returns a value of type IVL_QTY with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **QuantityIntervalLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="QuantityIntervalLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element minOccurs="0" name="low" type="dt:QTY"/>
```

```
        <xs:element minOccurs="0" name="high" type="dt:QTY"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: xs:boolean, predefined
Use: optional

highClosed

Type: xs:boolean, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 2/3)

low

Type: dt:QTY [121], empty content

XML Representation Summary

```
<low/>
```

high

Type: dt:QTY [121], empty content

XML Representation Summary

```
<high/>
```

complexType "RangeConstraint"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 element

XML Representation Summary

```
<...>
  Content: constraintType
</...>
```

Content Model Elements (1):

constraintType [329]

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="RangeConstraint">
  <xs:sequence>
    <xs:element name="constraintType">
      <xs:complexType>
        <xs:attribute name="value" type="RangeConstraintType"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

constraintType

Type: anonymous complexType, empty content

The constraint type defines how the value range is being constrained. For example, the constraint type may indicate the lower bound of the range.

XML Representation Summary

```
<constraintType
    value = (("Minimum" | "Maximum" | "List" | "Component") | ("Minimum" | "Maximum" | "List" | "Component"))
/>
```

complexType "RatioLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: description?, numerator, denominator
</...>
```

Content Model Elements (3):

denominator [330], numerator [330]

description (defined in *Expression* complexType) [230],

Annotation

The RaioLiteral expression returns a value of type RTO with the given numerator and denominator.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **RatioLiteral**

XML Source (w/o annotations (3))

```
<xs:complexType name="RatioLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element name="numerator" type="dt:QTY"/>
        <xs:element name="denominator" type="dt:QTY"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

numerator

Type: *dt:QTY* [121], empty content

The quantity that is being divided in the ratio

XML Representation Summary

```
<numerator/>
```

denominator

Type: *dt:QTY* [121], empty content

The quantity that divides the numerator in the ratio.
The denominator SHALL not be zero.

XML Representation Summary

```
<denominator/>
```

complexType "RealIntervalLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 4 attributes

XML Representation Summary

```
<...
    low         = xs:double
    high        = xs:double
    lowClosed   = xs:boolean
    highClosed  = xs:boolean
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The RealIntervalLiteral expression returns a value of type IVL_REAL with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **RealIntervalLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="RealIntervalLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="low" type="dt:Decimal"/>
      <xs:attribute name="high" type="dt:Decimal"/>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 4/4)

low

Type: dt:Decimal [134]
Use: optional

Attribute Value

xs:double

high

Type: dt:Decimal [134]
Use: optional

Attribute Value

xs:double

lowClosed

Type: xs:boolean, predefined
Use: optional

highClosed

Type: xs:boolean, predefined
Use: optional

complexType "RealLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   value = xs:double
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The RealLiteral expression returns a value of type Real with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **RealLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="RealLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="dt:Decimal" use="required"/>
```

```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

Type: dt:Decimal [134]
Use: required

Attribute Value

xs:double

complexType "RemoveAction"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
  Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
           textEquivalent?, actionSentence
</...>
```

Content Model Elements (8):

actionId [160],
actionSentence (defined in RemoveAction complexType) [334],
actors [160],
behaviors (defined in *ActionBase* complexType) [160],
conditions (defined in *ActionBase* complexType) [160],
supportingEvidence (defined in *ActionBase* complexType) [160],
supportingResources [160],
textEquivalent [174]

Annotation

This action removes another proposed action or an ongoing action.

Type Definition Detail

Type Derivation Tree

ActionBase [158] (extension)
└ *AtomicAction* [173] (extension)
  └ **RemoveAction**

XML Source (w/o annotations (2))

```
<xs:complexType name="RemoveAction">
  <xs:complexContent>
```

```
    <xs:extension base="AtomicAction">
      <xs:sequence>
        <xs:element name="actionSentence" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/8)

actionSentence

Type: *Expression* [230], complex content

The expression must resolve to the action that is being removed.

XML Representation Summary

```
<actionSentence>
    Content: description?
</actionSentence>
```

complexType "RequestBase"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definitions of 7 attributes, 1 element

XML Representation Summary

```
<...
    scope        = xs:string
    cardinality  = ("Single" | "Multiple")
    dataType     = xs:QName
    templateId   = xs:string
    idProperty   = xs:string
    triggerType  = (("DataElementAdded" | "DataElementModified" | "DataElementRemoved" |
                   "DataElementAccessed" | "DataElementAccessEnded") | ("DataElementAdded" |
                   "DataElementModified" | "DataElementRemoved" | "DataElementAccessed" |
                   "DataElementAccessEnded"))
    isInitial    = xs:boolean : "true"
    >
    Content: description?, timeOffset?
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], timeOffset [336]

Annotation

The request expression defines the data that will be used by the artifact.

The result of a request is defined to return the same data for subsequent invocations within the same evaluation request. This means in particular that patient data updates made during the evaluation request are not visible to the artifact. In effect, the patient data is a snapshot of the data as of the start of the evaluation. This ensures strict deterministic and functional behavior of the artifact, and allows the implementation engine freedom to cache intermediate results in order to improve performance.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ ***RequestBase***

XML Source (w/o annotations (9))

```
<xs:complexType abstract="true" name="RequestBase">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="timeOffset" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="scope" type="xs:string" use="optional"/>
      <xs:attribute name="cardinality" type="RequestCardinality" use="required"/>
      <xs:attribute name="dataType" type="xs:QName" use="required"/>
      <xs:attribute name="templateId" type="xs:string" use="optional"/>
      <xs:attribute name="idProperty" type="xs:string" use="optional"/>
      <xs:attribute name="triggerType" type="DataEventType" use="optional"/>
      <xs:attribute default="true" name="isInitial" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 7/7)

scope

Type: `xs:string`, predefined
Use: optional

The scope attribute optionally specifies a name for the scope for this operator that can be used within expressions to reference the current element.

cardinality

Type: `RequestCardinality` [400]
Use: required

The cardinality attribute defines the expected result cardinality for the expression. If cardinality is Single, the result will be a singleton value of the type specified by the dataType attribute. If cardinality is Multiple, the result will be a list of values of the type specified by the dataType attribute. If a request results in a cardinality higher than expected, a run-time exception should be thrown.

Attribute Value

```
enumeration of xs:string
```

Enumeration: "Single", "Multiple"

dataType

Type: xs:QName, predefined
Use: required

The dataType attribute specifies the type of clinical data being requested.

templateId

Type: xs:string, predefined
Use: optional

The templateId attribute specifies an optional template to be used. If specified, the request is defined to return only objects that conform to the template.

idProperty

Type: xs:string, predefined
Use: optional

The idProperty attribute specifies which property of the model contains the Id for the clinical statement.

triggerType

Type: DataEventType [387]
Use: optional

The triggerType attribute optionally specifies whether this request should only include data that was accessed or updated as a result of some event that is triggering the evaluation of the artifact. This attribute is only valid on a request that is specified as the request for a data changed event trigger.

Attribute Value

```
("DataElementAdded" | "DataElementModified" | "DataElementRemoved" | "DataElementAccessed" |
"DataElementAccessEnded") | ("DataElementAdded" | "DataElementModified" | "DataElementRemoved" |
"DataElementAccessed" | "DataElementAccessEnded")
```

isInitial

Type: xs:boolean, predefined
Use: optional

The isInitial attribute determines whether the request is part of the initial data requirements for the artifact.

Attribute Value

Default: "true"

Content Element Detail (all declarations; defined within this component only; 1/2)

timeOffset

Type: Expression [230], complex content

The timeOffset element optionally specifies an amount of time to delay execution of the artifact if it was triggered as a data event. This element is only valid when used on a request that is specified as the request for a data changed event trigger.

XML Representation Summary

```
<timeOffset>
    Content: description?
</timeOffset>
```

complexType "RequiredBehavior"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = (("Must" | "Could" | "MustUnlessDocumented") | ("Must" | "Could" | "MustUnlessDocumented"))
/>
```

Annotation

For a single action, specifies what level of requiredness is associated with the action.

Type Definition Detail

Type Derivation Tree

```
Behavior [178] (restriction)
  └ RequiredBehavior
```

XML Source (w/o annotations (1))

```
<xs:complexType name="RequiredBehavior">
  <xs:complexContent>
    <xs:restriction base="Behavior">
      <xs:attribute name="value" type="RequiredBehaviorType"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: RequiredBehaviorType [401]
Use: optional

Attribute Value

```
("Must" | "Could" | "MustUnlessDocumented") | ("Must" | "Could" | "MustUnlessDocumented")
```

complexType "ResourceRelationshipReference"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
   Content: relationship, resources
</...>
```

Content Model Elements (2):

relationship [339], resources (in relatedResource) [339]

Annotation

ResourceRelationshipReference defines the association between a resource and a resource set. If one consider such relationship as a triple (subject-predicate-object or node-arc-node) with the source entity as the subject of the triple, then ResourceRelationshipType defines the predicate and the object of this relationship.

XML Source (w/o annotations (4))

```
<xs:complexType name="ResourceRelationshipReference">
  <xs:sequence>
    <xs:element name="relationship">
      <xs:complexType>
        <xs:attribute name="value" type="ResourceRelationshipType" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="1" name="resources">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" name="resource" type="KnowledgeResource"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

relationship

Type: anonymous complexType, empty content

Relationship defines the association between two resources - i.e., the predicate/arc of a triple. The object of the relationship is often a pre-existing resource that predates the subject of the relationship. This predicate can be one of six

types: (1) AdaptedFrom - the subject was adapted from the object of the relationship. For instance, a diabetes visit order set may be adapted from the HbA1c reminder (2) AssociatedResource - the object is associated with the subject. For instance, an HbA1c reminder may be associated with an HbA1c eMeasure (3) DependsOn - the subject depends on the object. For instance, an HbA1c reminder may depend on a Diabetes Value Set (4) DerivedFrom - the subject was derived from the object. For instance, an HbA1c reminder rule may be derived from a Diabetes Guideline (5) SimilarTo - the subject and object are similar. For instance, an HbA1c reminder may be similar to an LDL reminder and (6) VersionOf - the subject is a version of the object (and vice-versa). For instance, the HbA1c reminder v2 may be a version of the HbA1c reminder v1.

XML Representation Summary

```
<relationship
    value = (("AdaptedFrom" | "AssociatedResource" | "DependsOn" | "DerivedFrom" | "SimilarTo" | "VersionOf") |
            ("AdaptedFrom" | "AssociatedResource" | "DependsOn" | "DerivedFrom" | "SimilarTo" | "VersionOf"))
/>
```

resources

Type: anonymous complexType, complex content

The set of resources associated with the subject of this relationship.

XML Representation Summary

```
<resources>
    Content: resource+
</resources>
```

complexType "ResponseBinding"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes

XML Representation Summary

```
<...
    container = xs:string : "Responses"
    property  = xs:string
/>
```

Annotation

Defines the attributes required to specify a binding path for documentation item responses.

The container attribute specifies the name of the response container that will be used. If no container attribute is provided, the default container name of Responses will be used.

The property attribute specifies the name of the property within the container that will be used to store the user response value.

XML Source (w/o annotations (1))

```
<xs:complexType name="ResponseBinding">
  <xs:attribute default="Responses" name="container" type="xs:string" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="required"/>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

- container

  Type: xs:string, predefined
  Use: optional

  Attribute Value

  Default: "Responses"

- property

  Type: xs:string, predefined
  Use: required

complexType "RightsDeclaration"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
  Content: assertedRights, rightsHolder, permissions?
</...>
```

Content Model Elements (3):

assertedRights [341], rightsHolder [341]
permissions (in rightsDeclaration) [341],

Annotation

This specifies the intellectual property rights associated with this CDS knowledge artifact, including who the rights holder is and what rights they assert. It also specifies what permissions are granted for usage. The asserted rights and persmissions are specified as a free-form text string.

XML Source (w/o annotations (1))

```
<xs:complexType name="RightsDeclaration">
  <xs:sequence>
    <xs:element name="assertedRights" type="dt:ST"/>
    <xs:element name="rightsHolder" type="Party"/>
    <xs:element minOccurs="0" name="permissions">
```

```
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" name="permissions" type="dt:ST"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/3)

assertedRights

Type: dt:ST [124], empty content

XML Representation Summary

```
<assertedRights
    value = xs:string
/>
```

rightsHolder

Type: *Party* [309], complex content

XML Representation Summary

```
<rightsHolder>
    Content: addresses?, contacts?
</rightsHolder>
```

permissions

Type: anonymous complexType, complex content

XML Representation Summary

```
<permissions>
    Content: permissions+
</permissions>
```

complexType "Round"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description?, operand, precision?
</...>
```

Content Model Elements (3):

description (defined in Expression complexType) [230], precision [342]

operand (defined in Round complexType) [342],

Annotation

The Round operator returns the nearest integer to its argument. The semantics of round are defined as a traditional round, meaning that a decimal value of 0.5 or higher will round to 1.

If the argument is null, the result is null.

Precision determines the decimal place at which the rounding will occur. If precision is not specified or null, 0 is assumed.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Round**

XML Source (w/o annotations (1))

```
<xs:complexType name="Round">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="operand" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="precision" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

operand

Type: Expression [230], complex content

XML Representation Summary

```
<operand>
    Content: description?
</operand>
```

precision

Type: Expression [230], complex content

XML Representation Summary

```
<precision>
    Content: description?
</precision>
```

complexType "SetSubsumes"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description?, ancestors?, descendents?
</...>
```

Content Model Elements (3):

ancestors [344], description (defined in *Expression* complexType) [230]
descendents [344],

Annotation

The SetSubsumes operator returns the list of descendants that were subsumed by some code in the list of ancestors.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **SetSubsumes**

XML Source (w/o annotations (3))

```
<xs:complexType name="SetSubsumes">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="ancestors" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="descendents" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

ancestors

Type: *Expression* [230], complex content

The codes that will be tested for the ancestor relationship.

XML Representation Summary

```
<ancestors>
    Content: description?
</ancestors>
```

descendents

Type: *Expression* [230], complex content

The codes that will be tested for the descendent relationship.

XML Representation Summary

```
<descendents>
    Content: description?
</descendents>
```

complexType "SimpleCodeLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    code = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The SimpleCodeLiteral expression returns a value of type CS with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **SimpleCodeLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="SimpleCodeLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="code" type="xs:string" use="required"/>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

code

Type: xs:string, predefined
Use: required

complexType "Sort"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 1 attribute, 1 element

XML Representation Summary

```
<...
   orderBy = xs:string
   >
   Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in Sort complexType) [346]

Annotation

The Sort operator returns a list with all the elements in source, sorted by the given orderBy.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Sort**

XML Source (w/o annotations (1))

```
<xs:complexType name="Sort">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element name="source" type="Expression"/>
      </xs:sequence>
      <xs:attribute name="orderBy" type="xs:string" use="optional"/>
    </xs:extension>
    <!-- TODO: Model ordering more explicitly? Multi-column sort? Ascending/Descending? -->
```

```
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

orderBy

Type: `xs:string`, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 1/2)

source

Type: *Expression* [230], complex content

XML Representation Summary

```
<source>
    Content: description?
</source>
```

complexType "Split"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description?, stringToSplit, separator?
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], stringToSplit [347]

separator (defined in Split complexType) [347],

Annotation

The Split operator splits a string into a list of strings using a separator.

If the stringToSplit argument is null, the result is null.

If the stringToSplit argument does not contain any appearances of the separator, the result is a list of strings containing one element that is the value of the stringToSplit argument.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Split**

XML Source (w/o annotations (1))

```
<xs:complexType name="Split">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="stringToSplit" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="separator" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

stringToSplit

Type: *Expression* [230], complex content

XML Representation Summary

```
<stringToSplit>
    Content: description?
</stringToSplit>
```

separator

Type: *Expression* [230], complex content

XML Representation Summary

```
<separator>
    Content: description?
</separator>
```

complexType "StdDev"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
    path = xs:string
    >
    Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The StdDev operator returns the statistical standard deviation of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *AggregateExpression* [167] (extension)
  └ **StdDev**

XML Source (w/o annotations (1))

```
<xs:complexType name="StdDev">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "StringLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in Expression complexType) [230]

Annotation

The StringLiteral expression returns a value of type ST with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **StringLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="StringLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

value

Type: xs:string, predefined
Use: required

complexType "Substring"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 elements

XML Representation Summary

```
<...>
  Content: description?, stringToSub, startIndex, length?
</...>
```

Content Model Elements (4):

description (defined in Expression complexType) [230], startIndex [350], length [350], stringToSub [350]

Annotation

The Substring operator returns the string within stringToSub, beginning at the 1-based index startIndex, and consisting of length characters.

If length is ommitted, the substring returned starts at startIndex and continues to the end of stringToSub.

If stringToSub or startIndex is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Substring**

XML Source (w/o annotations (1))

```
<xs:complexType name="Substring">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="stringToSub"/>
        <xs:element maxOccurs="1" minOccurs="1" name="startIndex"/>
        <xs:element maxOccurs="1" minOccurs="0" name="length"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 3/4)

stringToSub

Type: `xs:anyType`, any content

XML Representation Summary

```
<stringToSub>
    ...
</stringToSub>
```

startIndex

Type: `xs:anyType`, any content

XML Representation Summary

```
<startIndex>
    ...
</startIndex>
```

length

Type: `xs:anyType`, any content

XML Representation Summary

```
<length>
    ...
</length>
```

complexType "Subsumes"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
    Content: description?, ancestor?, descendent?
</...>
```

Content Model Elements (3):

ancestor [352], description (defined in Expression complexType) [230]
descendent [352],

Annotation

The Subsumes operator returns true if the operands were of the same code system, and the ancestor operand subsumed the descendant operand in the hierarchy of the code system. If the codes are the same code, the operator returns true.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **Subsumes**

XML Source (w/o annotations (3))

```
<xs:complexType name="Subsumes">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="ancestor" type="Expression"/>
        <xs:element maxOccurs="1" minOccurs="0" name="descendent" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/3)

ancestor

Type: Expression [230], complex content

The code that will be tested for the ancestor relationship.

XML Representation Summary

```
<ancestor>
    Content: description?
</ancestor>
```

descendent

Type: *Expression* [230], complex content

The code that will be tested for the descendent relationship.

XML Representation Summary

```
<descendent>
    Content: description?
</descendent>
```

complexType "Subtract"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The Subtract operator performs numeric subtraction of its arguments.

If either argument is null, the result is null.

The Subtract operator is defined for the Integer and Real types.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
- BinaryExpression [179] (extension)
  - **Subtract**

XML Source (w/o annotations (1))

```
<xs:complexType name="Subtract">
  <xs:complexContent>
```

```
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Succ"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Succ operator returns the successor of the argument. For example, the successor of 1 is 2.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

```
Expression [230] (extension)
  └ UnaryExpression [364] (extension)
      └ Succ
```

XML Source (w/o annotations (1))

```
<xs:complexType name="Succ">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Sum"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
    path = xs:string
    >
    Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Sum operator returns the sum of non-null elements in the source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *AggregateExpression* [167] (extension)
  └ **Sum**

XML Source (w/o annotations (1))

```
<xs:complexType name="Sum">
  <xs:complexContent>
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "SupportingEvidence"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: evidence+
</...>
```

Content Model Elements (1):

evidence [355]

Annotation

The evidence grade and the sources of evidence associated with this artifact.

XML Source (w/o annotations (2))

```
<xs:complexType name="SupportingEvidence">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" name="evidence" type="Evidence"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

evidence

Type: Evidence [227], complex content

A single evidence reference.

XML Representation Summary

```
<evidence>
    Content: qualityOfEvidence?, strengthOfRecommendation?, resources?
</evidence>
```

complexType "SupportingResource"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: resource+
</...>
```

Content Model Elements (1):

resource (defined in SupportingResource complexType) [356]

Annotation

Didactic or other informational resources associated with the artifact that can be provided to the CDS recipient. Information resources can include inline text commentary and links to web resources.Note, supporting resources excludes supporting evidence. For supporting evidence, use SupportingEvidence.

XML Source (w/o annotations (2))

```
<xs:complexType name="SupportingResource">
  <xs:sequence>
```

```
    <xs:element maxOccurs="unbounded" name="resource" type="KnowledgeResource"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

resource

Type: KnowledgeResource [265], complex content

An individual supporting resource.

XML Representation Summary

```
<resource>
    Content: identifiers?, templateIds?, title?, location?, description?, citation?
</resource>
```

complexType "TernaryExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: description?, operand[3..3]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in TernaryExpression complexType) [357]

Annotation

The TernaryExpression type defines the abstract base type for all expressions that take three arguments.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **TernaryExpression**

XML Source (w/o annotations (1))

```
<xs:complexType name="TernaryExpression">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="3" minOccurs="3" name="operand" type="Expression"/>
      </xs:sequence>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

operand

Type: *Expression* [230], complex content

XML Representation Summary

```
<operand>
    Content: description?
</operand>
```

complexType "TimeOf"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The TimeOf operator returns the time (with no date component) of the argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
- *UnaryExpression* [364] (extension)
  - **TimeOf**

XML Source (w/o annotations (1))

```
<xs:complexType name="TimeOf">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "TimestampIntervalLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 attributes, 2 elements

XML Representation Summary

```
<...
    lowClosed  = xs:boolean
    highClosed = xs:boolean
    >
    Content: description?, low?, high?
</...>
```

Content Model Elements (3):

description (defined in *Expression* complexType) [230], low (defined in TimestampIntervalLiteral complexType) [359]
high (defined in TimestampIntervalLiteral complexType) [359],

Annotation

The TimestampIntervalLiteral expression returns a value of type IVL_TS with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ **TimestampIntervalLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="TimestampIntervalLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="low" type="dt:TS"/>
        <xs:element maxOccurs="1" minOccurs="0" name="high" type="dt:TS"/>
      </xs:sequence>
      <xs:attribute name="lowClosed" type="xs:boolean" use="optional"/>
      <xs:attribute name="highClosed" type="xs:boolean" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 2/2)

lowClosed

Type: xs:boolean, predefined

Use: optional

highClosed

Type: `xs:boolean`, predefined
Use: optional

Content Element Detail (all declarations; defined within this component only; 2/3)

low

Type: `dt:TS [126]`, empty content

XML Representation Summary

```
<low
    value = xs:string
/>
```

high

Type: `dt:TS [126]`, empty content

XML Representation Summary

```
<high
    value = xs:string
/>
```

complexType "TimestampLiteral"

Namespace: `urn:hl7-org:knowledgeartifact:r1`
Includes: definition of 1 attribute

XML Representation Summary

```
<...
    value = xs:string
    >
    Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The TimestampLiteral expression returns a value of type TS with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **TimestampLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="TimestampLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="dt:TimeStamp" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/1)

- value

Type: dt:TimeStamp [146]
Use: required

Attribute Value

xs:string

Pattern: [1-2][0-9]{3,3}(((0[1-9])|(1[0-2]))(((0[1-9])|([1-2][0-9])|(3[0-1]))((([0-1][0-9])|(2[0-3]))(([0-5][0-9])(([0-5][0-9])(\.[0-9]{1,4})?)?)?)?)?([+\-](0[0-9]|1[0-3])([0-5][0-9]))?

complexType "Today"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

<...>

Content: description?

</...>

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The Today operator returns the date (with no time component) of the start timestamp associated with the evaluation request. See the Now operator for more information on the rationale for defining the Today operator in this way.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **Today**

XML Source (w/o annotations (1))

```
<xs:complexType name="Today">
  <xs:complexContent>
    <xs:extension base="Expression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "Trigger"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 2 elements

XML Representation Summary

```
<...>
  Content: eventType, expression?
</...>
```

Content Model Elements (2):

eventType (in trigger) [362], expression (in trigger) [362]

XML Source (w/o annotations (2))

```
<xs:complexType name="Trigger">
  <xs:sequence>
    <xs:element name="eventType" type="EventType"/>
    <xs:element minOccurs="0" name="expression" type="Expression"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 2/2)

eventType

Type: EventType [390], simple content

The event type can be one of either a DataEvent or a PeriodicEvent. A DataEvent is an event that is triggered by the value, presence, or absence of a particular data item. For instance, a data event may be triggered by a new serum potassium result below 3.5, a new appointment event to a Primary Care Clinic, or a new susbstance administration proposal for dabigatran. A periodic event is an event that is triggered on a regular basis. For instance, every day at midnight.

XML Representation Summary

```
<eventType>
    Content:        { ("DataEvent" | "PeriodicEvent") | ("DataEvent" | "PeriodicEvent") }
</eventType>
```

expression

Type: Expression [230], complex content

The event expression that must be met for the trigger to activate the record.

XML Representation Summary

```
<expression>
    Content: description?
</expression>
```

complexType "Triggers"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
    Content: trigger+
</...>
```

Content Model Elements (1):

trigger [362]

XML Source (w/o annotations (1))

```
<xs:complexType name="Triggers">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" name="trigger" type="Trigger"/>
  </xs:sequence>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/1)

trigger

Type: Trigger [361], complex content

The trigger element represents an event that 'triggers' the knowledge artifact. For instance, 'evaluate this artifact whenever a new Problem with a Diabetes Diagnosis code is added to the patient's record' or 'Evaluate this artifact every week day at 10:00 PM'. A trigger can model either a data event or a periodic event.

XML Representation Summary

```
<trigger>
    Content: eventType, expression?
</trigger>
```

complexType "Truncate"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Truncate operator returns the integer component of its argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *UnaryExpression* [364] (extension)
  └ **Truncate**

XML Source (w/o annotations (1))

```
<xs:complexType name="Truncate">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "TruncatedDivide"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
   Content: description?, operand[2..2]
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in BinaryExpression complexType) [180]

Annotation

The TruncatedDivide operator performs integer division of its arguments.

If either argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ BinaryExpression [179] (extension)

└ **TruncatedDivide**

XML Source (w/o annotations (1))

```
<xs:complexType name="TruncatedDivide">
  <xs:complexContent>
    <xs:extension base="BinaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "UnaryExpression"

Namespace: urn:hl7-org:knowledgeartifact:r1
Abstract: *(cannot be assigned directly to elements used in instance XML documents)*
Includes: definition of 1 element

XML Representation Summary

```
<...>
  Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The UnaryExpression type defines the abstract base type for all expressions that take a single argument.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ ***UnaryExpression***

XML Source (w/o annotations (1))

```
<xs:complexType abstract="true" name="UnaryExpression">
```

```
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="operand" type="Expression"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

operand

Type: *Expression* [230], complex content

XML Representation Summary

```
<operand>
    Content: description?
</operand>
```

complexType "Union"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand*
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *NaryExpression* complexType) [294]

Annotation

The Union operator returns the union of the operands.

This operator has two overloads:
List
Interval

For the list overload, this operator returns a list with all elements from all operands.

For the interval overload, this operator returns the interval that starts at the earliest starting point in the arguments, and ends at the latest starting point in the arguments. If the arguments do not overlap, this operator returns null.

If any of the arguments are null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *NaryExpression* [293] (extension)

└ **Union**

XML Source (w/o annotations (1))

```
<xs:complexType name="Union">
  <xs:complexContent>
    <xs:extension base="NaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "UpdateAction"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
  Content: (actionId?, supportingEvidence?, supportingResources?, actors?, behaviors?, conditions?)?,
           textEquivalent?, actionSentence
</...>
```

Content Model Elements (8):

actionId [160],

actionSentence (defined in UpdateAction complexType) [367],

actors [160],

behaviors (defined in *ActionBase* complexType) [160],

conditions (defined in *ActionBase* complexType) [160],

supportingEvidence (defined in *ActionBase* complexType) [160],

supportingResources [160],

textEquivalent [174]

Annotation

This action changing the value of another existing
action. The action being modified may be a proposed action (e.g., an
medication being prescribed by a clinician) or be an ongoing action
(e.g., an existing prescription). In these cases, a modification can
be used to change the dose of the medication. It may also be used to
discontinue a medication by changing the stop date.

Type Definition Detail

Type Derivation Tree

ActionBase [158] (extension)

└ AtomicAction [173] (extension)

└ **UpdateAction**

XML Source (w/o annotations (2))

```
<xs:complexType name="UpdateAction">
  <xs:complexContent>
    <xs:extension base="AtomicAction">
      <xs:sequence>
        <xs:element name="actionSentence" type="ObjectRedefine"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/8)

actionSentence

Type: ObjectRedefine [301], complex content

The modification to the action. This is specified by modifying the property of an existing action using the ObjectRedefine expression.

XML Representation Summary

```
<actionSentence
    scope = xs:string
    >
    Content: description?, source?, property+
</actionSentence>
```

complexType "Upper"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...>
    Content: description?, operand
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], operand (defined in *UnaryExpression* complexType) [365]

Annotation

The Upper operator returns the upper case of its argument.

If the argument is null, the result is null.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ *UnaryExpression* [364] (extension)

└ **Upper**

XML Source (w/o annotations (1))

```
<xs:complexType name="Upper">
  <xs:complexContent>
    <xs:extension base="UnaryExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "UrlLiteral"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 attributes

XML Representation Summary

```
<...
   value        = xs:anyURI
   use          = list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" |
                  "MC" | "PG")
   capabilities = list of ("voice" | "fax" | "data" | "tty" | "sms")
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The UrlLiteral expression returns a value of type TEL with the given attributes.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **UrlLiteral**

XML Source (w/o annotations (1))

```
<xs:complexType name="UrlLiteral">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="value" type="xs:anyURI" use="required"/>
      <xs:attribute name="use" type="dt:set_TelecommunicationAddressUse" use="optional"/>
      <xs:attribute name="capabilities" type="dt:set_TelecommunicationCapability" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 3/3)

value

Type: xs:anyURI, predefined
Use: required

use

Type: dt:set_TelecommunicationAddressUse [143]
Use: optional

Attribute Value

```
list of ("H" | "HP" | "HV" | "WP" | "DIR" | "PUB" | "BAD" | "TMP" | "AS" | "EC" | "MC" | "PG")
```

capabilities

Type: dt:set_TelecommunicationCapability [144]
Use: optional

Attribute Value

```
list of ("voice" | "fax" | "data" | "tty" | "sms")
```

complexType "ValueSet"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definitions of 3 attributes

XML Representation Summary

```
<...
   id        = xs:string
   version   = xs:string
   authority = xs:string
   >
   Content: description?
</...>
```

Content Model Elements (1):

description (defined in *Expression* complexType) [230]

Annotation

The ValueSet operator returns a list of codes whose elements are defined by the given value set authority for the given value set id and version. If version is an asterisk (*), the current version of the value set is returned.

Note that the id, version, and authority for the value set are specified by attributes, rather than as expression elements. This is deliberately done to ensure that the value sets involved in an artifact can be determined by static analysis.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)

└ **ValueSet**

XML Source (w/o annotations (4))

```
<xs:complexType name="ValueSet">
  <xs:complexContent>
    <xs:extension base="Expression">
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="version" type="xs:string" use="optional"/>
      <xs:attribute name="authority" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 3/3)

id

Type: xs:string, predefined
Use: required

The unique identifier of the value set to be retrieved. This identifier is evaluated within the context of the authority.

version

Type: xs:string, predefined
Use: optional

The version of the value set to be retrieved. If an asterisk (*) is used, the current version of the value set is returned.

authority

Type: xs:string, predefined
Use: optional

The authority for the value set to be retrieved. This value represents a conceptual identifier for the steward of the value set. An integration would need to be able to map this to a specific service providing the value sets maintained by this authority.

complexType "ValueSetConstraint"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 element

XML Representation Summary

```
<...>
  Content: constraintType, valueSet
</...>
```

Content Model Elements (2):

constraintType [329], valueSet [372]

Annotation

A constraint specifying that the value is an item from the value set specified. This constraint applies to list type constraints only. The response data type for this DocumentationItem will be Code.

Type Definition Detail

Type Derivation Tree

RangeConstraint [328] (extension)
└ **ValueSetConstraint**

XML Source (w/o annotations (1))

```
<xs:complexType name="ValueSetConstraint">
  <xs:complexContent>
    <xs:extension base="RangeConstraint">
      <xs:sequence>
        <xs:element name="valueSet" type="ValueSet"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Content Element Detail (all declarations; defined within this component only; 1/2)

valueSet

Type: ValueSet [369], complex content

XML Representation Summary

```
<valueSet
    authority = xs:string
    id        = xs:string
    version   = xs:string
    >
    Content: description?
</valueSet>
```

complexType "Variance"

Namespace: urn:hl7-org:knowledgeartifact:r1

XML Representation Summary

```
<...
    path = xs:string
    >
    Content: description?, source
</...>
```

Content Model Elements (2):

description (defined in *Expression* complexType) [230], source (defined in *AggregateExpression* complexType) [168]

Annotation

The Variance operator returns the statistical variance of the elements in source.

If a path is specified, elements with no value for the property specified by the path are ignored.

If source contains no non-null elements, null is returned.

Type Definition Detail

Type Derivation Tree

Expression [230] (extension)
└ *AggregateExpression* [167] (extension)
  └ **Variance**

XML Source (w/o annotations (1))

```
<xs:complexType name="Variance">
  <xs:complexContent>
```

```
    <xs:extension base="AggregateExpression"/>
  </xs:complexContent>
</xs:complexType>
```

complexType "VersionedIdentifier"

Namespace: urn:hl7-org:knowledgeartifact:r1
Includes: definition of 1 attribute

XML Representation Summary

```
<...
   root           = xs:string
   extension      = xs:string
   identifierName = xs:string
   version        = xs:string
/>
```

Annotation

VersionedIdentifier is composed of two parts: (1) an II identifier which identifies the set of all versions of a given resource. (2) the actual version of the instance of interest in this set. The VersionedIdentifier therefore points to an individual 'versioned' instance of a resource such as the third version of a reminder rule.

Type Definition Detail

Type Derivation Tree

```
dt:HXIT [99] (extension)
  └ dt:ANY [84] (extension)
      └ dt:II [100] (extension)
          └ VersionedIdentifier
```

XML Source (w/o annotations (1))

```
<xs:complexType name="VersionedIdentifier">
  <xs:complexContent>
    <xs:extension base="dt:II">
      <xs:attribute name="version" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Attribute Detail (all declarations; defined within this component only; 1/4)

- version

Type: xs:string, predefined
Use: optional

simpleType "ArtifactLifeCycleEventType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

("Created" | "Pre-published" | "Published" | "Reviewed" | "Withdrawn" | "Superseded") | ("Created" | "Pre-published" | "Published" | "Reviewed" | "Withdrawn" | "Superseded")

Annotation

A version of an artifact may have different actions performed on it during the course of its life cycle. Each action is considered an artifact life cycle event, and may be recorded in the history of that version of the artifact.

See the Implementation Guide for a state-transition diagram showing the legal transitions from each state; each transition is equivalent to an event.

Type Definition Detail

Type Derivation Tree

union of (ArtifactLifeCycleEventTypeCore | ArtifactLifeCycleEventTypeExt)

└ **ArtifactLifeCycleEventType**

Derivation: by union

Member Types

1. ArtifactLifeCycleEventTypeCore
2. ArtifactLifeCycleEventTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="ArtifactLifeCycleEventType">
  <xs:union memberTypes="ArtifactLifeCycleEventTypeCore ArtifactLifeCycleEventTypeExt"/>
</xs:simpleType>
```

simpleType "ArtifactLifeCycleEventTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Created" – The date/time when a particular version of an artifact is created may be recorded in the history of the artifact-version.

"Pre- – An artifact may be made available to consumers in a pre-publication state for public commentary, etc., or may simply

published" "Published"	be made available for review internal to the publisher. This event can be recorded in the artifact-version's history.
"Reviewed"	– A specific version of artifact may be reviewed by parties internal or external to the publisher, and this can be recorded in the lifecycle of the artifact-version.
"Withdrawn"	– A specific version of an artifact may be withdrawn by a publisher for various reasons.
"Superseded"	– A specific version of an artifact may be superseded by another version of that same artifact, or by another artifact altogether.

Annotation

A lifecycle event is applicable to a specific version of an artifact.

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`ArtifactLifeCycleEventTypeCore`**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

"Created"	– The date/time when a particular version of an artifact is created may be recorded in the history of the artifact-version.
"Pre-published"	– An artifact may be made available to consumers in a pre-publication state for public commentary, etc., or may simply be made available for review internal to the publisher. This event can be recorded in the artifact-version's history.
"Published"	
"Reviewed"	– A specific version of artifact may be reviewed by parties internal or external to the publisher, and this can be recorded in the lifecycle of the artifact-version.
"Withdrawn"	– A specific version of an artifact may be withdrawn by a publisher for various reasons.
"Superseded"	– A specific version of an artifact may be superseded by another version of that same artifact, or by another artifact altogether.

XML Source (w/o annotations (6))

```
<xs:simpleType name="ArtifactLifeCycleEventTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Created"/>
    <xs:enumeration value="Pre-published"/>
    <xs:enumeration value="Published"/>
    <xs:enumeration value="Reviewed"/>
    <xs:enumeration value="Withdrawn"/>
    <xs:enumeration value="Superseded"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "ArtifactLifeCycleEventTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration: "Created" – The date/time when a particular version of an artifact is created may be recorded in the history of the artifact-version.

"Pre-published"	– An artifact may be made available to consumers in a pre-publication state for public commentary, etc., or may simply be made available for review internal to the publisher. This event can be recorded in the artifact-version's history.
"Published"	
"Reviewed"	– A specific version of artifact may be reviewed by parties internal or external to the publisher, and this can be recorded in the lifecycle of the artifact-version.
"Withdrawn"	– A specific version of an artifact may be withdrawn by a publisher for various reasons.
"Superseded"	– A specific version of an artifact may be superseded by another version of that same artifact, or by another artifact altogether.

Annotation

To add new items to the enumeration ArtifactLifeCycleEvent, comment out or delete the restriction on ArtifactLifeCycleEventTypeCore.(Optionally, also remove
the directive above to include the artifactlifecycleeventtypecore.xsd).
Now, add a new restriction such as on string and define new enumerated types. See the commented out example below

Type Definition Detail

Type Derivation Tree

```
xs:string (restriction)
  └ ArtifactLifeCycleEventTypeCore [374] (restriction)
      └ ArtifactLifeCycleEventTypeExt
```

Derivation: restriction of `ArtifactLifeCycleEventTypeCore`

XML Source (w/o annotations (1))

```
<xs:simpleType name="ArtifactLifeCycleEventTypeExt">
  <xs:restriction base="ArtifactLifeCycleEventTypeCore"/>
  <!--
  <xs:restriction base="xs:string">
  <xs:enumeration value="Erased"/>
  <xs:enumeration value="Restored"/>
  </xs:restriction>
  -->
</xs:simpleType>
```

simpleType "ArtifactStatusType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

```
("Draft" | "InTest" | "Active" | "Inactive") | ("Draft" | "InTest" | "Active" | "Inactive")
```

Annotation

A specific status is associated with each version of an artifact.

See the Implementation Guide for a state-transition diagram showing the legal transitions from each state; each state is equivalent to particular status.

Type Definition Detail

Type Derivation Tree

union of (ArtifactStatusTypeCore | ArtifactStatusTypeExt)

└ **ArtifactStatusType**

Derivation: by union

Member Types

1. ArtifactStatusTypeCore
2. ArtifactStatusTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="ArtifactStatusType">
  <xs:union memberTypes="ArtifactStatusTypeCore ArtifactStatusTypeExt"/>
</xs:simpleType>
```

simpleType "ArtifactStatusTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Draft"
"InTest"
"Active"
"Inactive"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **ArtifactStatusTypeCore**

Derivation: restriction of xs:string
Facets: **enumeration:** "Draft"
"InTest"
"Active"
"Inactive"

XML Source (w/o annotations (1))

```
<xs:simpleType name="ArtifactStatusTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Draft"/>
    <xs:enumeration value="InTest"/>
    <xs:enumeration value="Active"/>
    <xs:enumeration value="Inactive"/>
  </xs:restriction>
```

```
</xs:simpleType>
```

simpleType "ArtifactStatusTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Draft"
"InTest"
"Active"
"Inactive"

Annotation

To add new items to the enumeration ArtifactStatus, comment out or delete the restriction on ArtifactStatusTypeCore.(Optionally, also remove
the directive above to include the artifactstatustypecore.xsd).
Now, add a new restriction such as on string and define new enumerated types. See the commented out example below. !!!
BKM get example from ArtifactLifeCycleEventTypeExt

Type Definition Detail

Type Derivation Tree

xs:string (restriction)
└ ArtifactStatusTypeCore [377] (restriction)
  └ **ArtifactStatusTypeExt**

Derivation: restriction of ArtifactStatusTypeCore

XML Source (w/o annotations (1))

```
<xs:simpleType name="ArtifactStatusTypeExt">
  <xs:restriction base="ArtifactStatusTypeCore"/>
</xs:simpleType>
```

simpleType "ArtifactType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

("Rule" | "Order Set" | "Documentation Template") | ("Rule" | "Order Set" | "Documentation Template")

Annotation

Three types of artifacts are in scope for Health eDecisions Use Case #1: Event-condition-action rules, order sets and documentation templates.

Type Definition Detail

Type Derivation Tree

union of (ArtifactTypeCore | ArtifactTypeExt)

└ **ArtifactType**

Derivation: by union

Member Types

1. ArtifactTypeCore
2. ArtifactTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="ArtifactType">
  <xs:union memberTypes="ArtifactTypeCore ArtifactTypeExt"/>
</xs:simpleType>
```

simpleType "ArtifactTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"Rule"	– Denotes an Event-Condition-Action Rule. An ECA rule has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to ECA rules.
"Order Set"	– Denotes an Order Set for use by physicians at the point of care. An Order Set has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to Order Sets.
"Documentation Template"	– Denotes a Documentation Template for use by care providers. A Documentation Template has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to Documentation Templates.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **ArtifactTypeCore**

Derivation: restriction of xs:string

Facets: **enumeration:**

"Rule"	– Denotes an Event-Condition-Action Rule. An ECA rule has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to ECA rules.
"Order Set"	– Denotes an Order Set for use by physicians at the point of care. An Order Set has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to Order Sets.
"Documentation Template"	– Denotes a Documentation Template for use by care providers. A Documentation Template has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to Documentation Templates.

XML Source (w/o annotations (3))

```
<xs:simpleType name="ArtifactTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Rule"/>
    <xs:enumeration value="Order Set"/>
    <xs:enumeration value="Documentation Template"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "ArtifactTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"Rule"	– Denotes an Event-Condition-Action Rule. An ECA rule has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to ECA rules.
"Order Set"	– Denotes an Order Set for use by physicians at the point of care. An Order Set has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to Order Sets.
"Documentation Template"	– Denotes a Documentation Template for use by care providers. A Documentation Template has a specific structure that is a restriction on the general structure of a CDS Knowledge Artifact. See the Implementation Guide for the constraints applicable to Documentation Templates.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ ArtifactTypeCore [379] (restriction)

└ **ArtifactTypeExt**

Derivation: restriction of ArtifactTypeCore

XML Source

```
<xs:simpleType name="ArtifactTypeExt">
  <xs:restriction base="ArtifactTypeCore"/>
</xs:simpleType>
```

simpleType "Cardinality"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: `"Single", "Multiple"`

Annotation

Cardinality defines the expected cardinality of an element, single or multiple.

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **Cardinality**

Derivation: restriction of `xs:string`
Facets: **enumeration:** `"Single", "Multiple"`

XML Source (w/o annotations (1))

```
<xs:simpleType name="Cardinality">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Single"/>
    <xs:enumeration value="Multiple"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "ConditionRoleType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

`"ApplicableScenario" | "ApplicableScenario"`

Annotation

The roles that a condition plays in the execution of a component. Currently, only one role type is defined. Additional role types may be defined in the future (e.g., inclusion criteria, exclusion criteria)

Type Definition Detail

Type Derivation Tree

union of (`ConditionRoleTypeCore` | `ConditionRoleTypeExt`)

└ **ConditionRoleType**

Derivation: by union

Member Types

1. `ConditionRoleTypeCore`
2. `ConditionRoleTypeExt`

XML Source (w/o annotations (1))

```
<xs:simpleType name="ConditionRoleType">
  <xs:union memberTypes="ConditionRoleTypeCore ConditionRoleTypeExt"/>
</xs:simpleType>
```

simpleType "ConditionRoleTypeCore"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration: `"ApplicableScenario"` – This role type specifies that a condition is used to determine whether or not a particular knowledge component should be executed. If the expression evaluates to true, then the component is executed.

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **`ConditionRoleTypeCore`**

Derivation: restriction of `xs:string`

Facets: **enumeration:** `"ApplicableScenario"` – This role type specifies that a condition is used to determine whether or not a particular knowledge component should be executed. If the expression evaluates to true, then the component is executed.

XML Source (w/o annotations (1))

```
<xs:simpleType name="ConditionRoleTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ApplicableScenario"/>
    <!-- <xs:enumeration value="Inclusion"></xs:enumeration>
    <xs:enumeration value="Exclusion"></xs:enumeration>
    <xs:enumeration value="Precondition"></xs:enumeration>
    <xs:enumeration value="Postcondition"></xs:enumeration> -->
  </xs:restriction>
</xs:simpleType>
```

simpleType "ConditionRoleTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration: `"ApplicableScenario"` – This role type specifies that a condition is used to determine whether or not a particular knowledge component should be executed. If the expression evaluates to true, then the component is executed.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ ConditionRoleTypeCore [382] (restriction)

└ **ConditionRoleTypeExt**

Derivation: restriction of ConditionRoleTypeCore

XML Source

```
<xs:simpleType name="ConditionRoleTypeExt">
  <xs:restriction base="ConditionRoleTypeCore"/>
</xs:simpleType>
```

simpleType "ContributorType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

("Author" | "Editor" | "Endorser" | "Reviewer") | ("Author" | "Editor" | "Endorser" | "Reviewer")

Annotation

Enumeration of roles that contribute to the development and maintenance of a knowledge artifact.

Type Definition Detail

Type Derivation Tree

union of (ContributorTypeCore | ContributorTypeExt)

└ **ContributorType**

Derivation: by union

Member Types

1. ContributorTypeCore
2. ContributorTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="ContributorType">
  <xs:union memberTypes="ContributorTypeCore ContributorTypeExt"/>
</xs:simpleType>
```

simpleType "ContributorTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Author", "Editor", "Endorser", "Reviewer"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **ContributorTypeCore**

Derivation: restriction of xs:string
Facets: **enumeration:** "Author", "Editor", "Endorser", "Reviewer"

XML Source

```
<xs:simpleType name="ContributorTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Author"/>
    <xs:enumeration value="Editor"/>
    <xs:enumeration value="Endorser"/>
    <xs:enumeration value="Reviewer"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "ContributorTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Author", "Editor", "Endorser", "Reviewer"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ ContributorTypeCore [383] (restriction)

└ **ContributorTypeExt**

Derivation: restriction of ContributorTypeCore

XML Source

```
<xs:simpleType name="ContributorTypeExt">
```

```
  <xs:restriction base="ContributorTypeCore"/>
</xs:simpleType>
```

simpleType "CoverageType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

```
("PatientGender" | "PatientAgeGroup" | "ClinicalFocus" | "TargetUser" | "WorkflowSetting" |
"WorkflowTask" | "ClinicalVenue") | ("PatientGender" | "PatientAgeGroup" | "ClinicalFocus" |
"TargetUser" | "WorkflowSetting" | "WorkflowTask" | "ClinicalVenue")
```

Annotation

Specifies clinical metadata that can be used to retrieve, index and/or categorize the knowledge artifact. This metadata can either be specific to the applicable population (e.g., age category, DRG) or the specific context of care (e.g., venue, care setting, provider of care).

Type Definition Detail

Type Derivation Tree

union of (CoverageTypeCore | CoverageTypeExt)

└ **CoverageType**

Derivation: by union

Member Types

1. CoverageTypeCore
2. CoverageTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="CoverageType">
  <xs:union memberTypes="CoverageTypeCore CoverageTypeExt"/>
</xs:simpleType>
```

simpleType "CoverageTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

- "PatientGender" – The gender of the patient. For this item type, use HL7 administrative gender codes (OID: 2.16.840.1.113883.1.11.1)
- "PatientAgeGroup" – A patient demographic category for which this artifact is applicable. Allows specification of age groups using coded values originating from the MeSH Code system (OID: 2.16.840.1.113883.6.177). More specifically, only codes from the AgeGroupObservationValue value set are valid for this field [2.16.840.1.113883.11.75]
- "ClinicalFocus" – The clinical concept(s) addressed by the artifact. For example, disease, diagnostic test interpretation, medication ordering. Please refer to the implementation guide on which code system and codes to use.
- "TargetUser" – The user types to which an artifact is targeted. For example, PCP, Patient, Cardiologist, Behavioral Professional, Oral Health Professional, Prescriber, etc... taken from the NUCC Health Care provider taxonomyCode system (OID: 2.16.840.1.113883.6.101)

"WorkflowSetting" – The settings in which the artifact is intended for use. For example, admission, pre-op, etc.

"WorkflowTask" – The context for the clinical task(s) represented by this artifact. Can be any task context represented by the HL7 ActTaskCode value set (OID: 2.16.840.1.113883.1.11.19846). General categories include: order entry, patient documentation and patient information review

"ClinicalVenue" – The venue in which an artifact could be used. For example, Outpatient, Inpatient, Home, Nursing home. The code value may originate from either the HL7 ActEncounter (OID: 2.16.840.1.113883.1.11.13955) or NUCC non-individual provider codes OID: 2.16.840.1.113883.1.11.19465

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **CoverageTypeCore**

Derivation: restriction of xs:string

Facets: **enumeration:**

"PatientGender" – The gender of the patient. For this item type, use HL7 administrative gender codes (OID: 2.16.840.1.113883.1.11.1)

"PatientAgeGroup" – A patient demographic category for which this artifact is applicable. Allows specification of age groups using coded values originating from the MeSH Code system (OID: 2.16.840.1.113883.6.177). More specifically, only codes from the AgeGroupObservationValue value set are valid for this field [2.16.840.1.113883.11.75]

"ClinicalFocus" – The clinical concept(s) addressed by the artifact. For example, disease, diagnostic test interpretation, medication ordering. Please refer to the implementation guide on which code system and codes to use.

"TargetUser" – The user types to which an artifact is targeted. For example, PCP, Patient, Cardiologist, Behavioral Professional, Oral Health Professional, Prescriber, etc... taken from the NUCC Health Care provider taxonomyCode system (OID: 2.16.840.1.113883.6.101)

"WorkflowSetting" – The settings in which the artifact is intended for use. For example, admission, pre-op, etc.

"WorkflowTask" – The context for the clinical task(s) represented by this artifact. Can be any task context represented by the HL7 ActTaskCode value set (OID: 2.16.840.1.113883.1.11.19846). General categories include: order entry, patient documentation and patient information review

"ClinicalVenue" – The venue in which an artifact could be used. For example, Outpatient, Inpatient, Home, Nursing home. The code value may originate from either the HL7 ActEncounter (OID: 2.16.840.1.113883.1.11.13955) or NUCC non-individual provider codes OID: 2.16.840.1.113883.1.11.19465

XML Source (w/o annotations (7))

```
<xs:simpleType name="CoverageTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="PatientGender"/>
    <xs:enumeration value="PatientAgeGroup"/>
    <xs:enumeration value="ClinicalFocus"/>
    <xs:enumeration value="TargetUser"/>
    <xs:enumeration value="WorkflowSetting"/>
    <xs:enumeration value="WorkflowTask"/>
    <xs:enumeration value="ClinicalVenue"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "CoverageTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

- `"PatientGender"` – The gender of the patient. For this item type, use HL7 administrative gender codes (OID: 2.16.840.1.113883.1.11.1)
- `"PatientAgeGroup"` – A patient demographic category for which this artifact is applicable. Allows specification of age groups using coded values originating from the MeSH Code system (OID: 2.16.840.1.113883.6.177). More specifically, only codes from the AgeGroupObservationValue value set are valid for this field [2.16.840.1.113883.11.75]
- `"ClinicalFocus"` – The clinical concept(s) addressed by the artifact. For example, disease, diagnostic test interpretation, medication ordering. Please refer to the implementation guide on which code system and codes to use.
- `"TargetUser"` – The user types to which an artifact is targeted. For example, PCP, Patient, Cardiologist, Behavioral Professional, Oral Health Professional, Prescriber, etc... taken from the NUCC Health Care provider taxonomyCode system (OID: 2.16.840.1.113883.6.101)
- `"WorkflowSetting"` – The settings in which the artifact is intended for use. For example, admission, pre-op, etc.
- `"WorkflowTask"` – The context for the clinical task(s) represented by this artifact. Can be any task context represented by the HL7 ActTaskCode value set (OID: 2.16.840.1.113883.1.11.19846). General categories include: order entry, patient documentation and patient information review
- `"ClinicalVenue"` – The venue in which an artifact could be used. For example, Outpatient, Inpatient, Home, Nursing home. The code value may originate from either the HL7 ActEncounter (OID: 2.16.840.1.113883.1.11.13955) or NUCC non-individual provider codes OID: 2.16.840.1.113883.1.11.19465

Type Definition Detail

Type Derivation Tree

```
xs:string (restriction)
  └ CoverageTypeCore [385] (restriction)
      └ CoverageTypeExt
```

Derivation: restriction of `CoverageTypeCore`

XML Source

```
<xs:simpleType name="CoverageTypeExt">
  <xs:restriction base="CoverageTypeCore"/>
</xs:simpleType>
```

simpleType "DataEventType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

```
("DataElementAdded" | "DataElementModified" | "DataElementRemoved" | "DataElementAccessed" |
"DataElementAccessEnded") | ("DataElementAdded" | "DataElementModified" | "DataElementRemoved" |
"DataElementAccessed" | "DataElementAccessEnded")
```

Annotation

Enumeration of types of events related to access, creation, removal, or update of data.

Type Definition Detail

Type Derivation Tree

```
union of (DataEventTypeCore | DataEventTypeExt)
  └ DataEventType
```

Derivation: by union

Member Types

1. DataEventTypeCore
2. DataEventTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="DataEventType">
  <xs:union memberTypes="DataEventTypeCore DataEventTypeExt"/>
</xs:simpleType>
```

simpleType "DataEventTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "DataElementAdded", "DataElementModified", "DataElementRemoved", "DataElementAccessed", "DataElementAccessEnded"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **DataEventTypeCore**

Derivation: restriction of xs:string
Facets: **enumeration:** "DataElementAdded", "DataElementModified", "DataElementRemoved", "DataElementAccessed", "DataElementAccessEnded"

XML Source

```
<xs:simpleType name="DataEventTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="DataElementAdded"/>
    <xs:enumeration value="DataElementModified"/>
    <xs:enumeration value="DataElementRemoved"/>
    <xs:enumeration value="DataElementAccessed"/>
    <xs:enumeration value="DataElementAccessEnded"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "DataEventTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "DataElementAdded", "DataElementModified", "DataElementRemoved", "DataElementAccessed", "DataElementAccessEnded"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ DataEventTypeCore [388] (restriction)

└ **DataEventTypeExt**

Derivation: restriction of DataEventTypeCore

XML Source

```
<xs:simpleType name="DataEventTypeExt">
  <xs:restriction base="DataEventTypeCore"/>
</xs:simpleType>
```

simpleType "DateGranularity"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Year", "Month", "Week", "Day", "Hour", "Minute", "Second", "Millisecond"

Annotation

The DateGranularity type specifies the granularities available for temporal operations such as DateAdd, DateDiff, and DatePart.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **DateGranularity**

Derivation: restriction of xs:string
Facets: **enumeration:** "Year", "Month", "Week", "Day", "Hour", "Minute", "Second", "Millisecond"

XML Source (w/o annotations (1))

```
<xs:simpleType name="DateGranularity">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Year"/>
    <xs:enumeration value="Month"/>
```

```
    <xs:enumeration value="Week"/>
    <xs:enumeration value="Day"/>
    <xs:enumeration value="Hour"/>
    <xs:enumeration value="Minute"/>
    <xs:enumeration value="Second"/>
    <xs:enumeration value="Millisecond"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "EventType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

("DataEvent" | "PeriodicEvent") | ("DataEvent" | "PeriodicEvent")

Annotation

An enumeration of event types. Events occur external to the artifact that can be used as a trigger to the artifact.

Type Definition Detail

Type Derivation Tree

union of (EventTypeCore | EventTypeExt)

└ **EventType**

Derivation: by union

Member Types

1. EventTypeCore
2. EventTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="EventType">
  <xs:union memberTypes="EventTypeCore EventTypeExt"/>
</xs:simpleType>
```

simpleType "EventTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "DataEvent" – An event in which a data item is created, removed, updated, or accessed. Expression is expected to be an ExpressionRef that references an ExpressionDef in ExternalData that contains a Request with a triggerType attribute specified.

"PeriodicEvent" – A time-based event which occurs at the specified period. Expression is expected to be a Period literal expression specifying the period on which the event should be repeated

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **EventTypeCore**

Derivation: restriction of xs:string

Facets: **enumeration:** "DataEvent" – An event in which a data item is created, removed, updated, or accessed. Expression is expected to be an ExpressionRef that references an ExpressionDef in ExternalData that contains a Request with a triggerType attribute specified.

"PeriodicEvent" – A time-based event which occurs at the specified period. Expression is expected to be a Period literal expression specifying the period on which the event should be repeated

XML Source (w/o annotations (2))

```
<xs:simpleType name="EventTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="DataEvent"/>
    <xs:enumeration value="PeriodicEvent"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "EventTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "DataEvent" – An event in which a data item is created, removed, updated, or accessed. Expression is expected to be an ExpressionRef that references an ExpressionDef in ExternalData that contains a Request with a triggerType attribute specified.

"PeriodicEvent" – A time-based event which occurs at the specified period. Expression is expected to be a Period literal expression specifying the period on which the event should be repeated

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ EventTypeCore [390] (restriction)

└ **EventTypeExt**

Derivation: restriction of EventTypeCore

XML Source

```
<xs:simpleType name="EventTypeExt">
  <xs:restriction base="EventTypeCore"/>
</xs:simpleType>
```

simpleType "GroupOrganizationBehaviorType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

`("VisualGroup" | "LogicalGroup" | "SentenceGroup") | ("VisualGroup" | "LogicalGroup" | "SentenceGroup")`

Annotation

Defines organization behavior of a group: gives the reason why the items are grouped together.

Type Definition Detail

Type Derivation Tree

union of (`GroupOrganizationBehaviorTypeCore` | `GroupOrganizationBehaviorTypeExt`)

└ **`GroupOrganizationBehaviorType`**

Derivation: by union

Member Types

1. `GroupOrganizationBehaviorTypeCore`
2. `GroupOrganizationBehaviorTypeExt`

XML Source (w/o annotations (1))

```
<xs:simpleType name="GroupOrganizationBehaviorType">
  <xs:union memberTypes="GroupOrganizationBehaviorTypeCore GroupOrganizationBehaviorTypeExt"/>
</xs:simpleType>
```

simpleType "GroupOrganizationBehaviorTypeCore"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration:

- `"VisualGroup"` – Any group marked with this behavior should be displayed as a visual group to the end user.
- `"LogicalGroup"` – A group with this behavior logically groups its sub-elements, and may be shown as a visual group to the end user, but it is not required to do so.
- `"SentenceGroup"` – A group of related alternative actions is a sentence group if the item referenced by the action is the same in all the actions, and each action simply constitutes a different variation on how to specify the details for that item. For example, two actions that could be in a SentenceGroup are "aspirin, 500 mg, 2 times per day" and "aspirin, 300 mg, 3 times per day". In both cases, aspirin is the item referenced by the action, and the two actions represent two different options for how aspirin might be ordered for the patient. Note that a SentenceGroup would almost always have an associated selection behavior of "AtMostOne", unless it's a required action, in which case, it would be "ExactlyOne".

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **GroupOrganizationBehaviorTypeCore**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

- `"VisualGroup"` – Any group marked with this behavior should be displayed as a visual group to the end user.
- `"LogicalGroup"` – A group with this behavior logically groups its sub-elements, and may be shown as a visual group to the end user, but it is not required to do so.
- `"SentenceGroup"` – A group of related alternative actions is a sentence group if the item referenced by the action is the same in all the actions, and each action simply constitutes a different variation on how to specify the details for that item. For example, two actions that could be in a SentenceGroup are "aspirin, 500 mg, 2 times per day" and "aspirin, 300 mg, 3 times per day". In both cases, aspirin is the item referenced by the action, and the two actions represent two different options for how aspirin might be ordered for the patient. Note that a SentenceGroup would almost always have an associated selection behavior of "AtMostOne", unless it's a required action, in which case, it would be "ExactlyOne".

XML Source (w/o annotations (3))

```
<xs:simpleType name="GroupOrganizationBehaviorTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="VisualGroup"/>
    <xs:enumeration value="LogicalGroup"/>
    <xs:enumeration value="SentenceGroup"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "GroupOrganizationBehaviorTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration:

- `"VisualGroup"` – Any group marked with this behavior should be displayed as a visual group to the end user.
- `"LogicalGroup"` – A group with this behavior logically groups its sub-elements, and may be shown as a visual group to the end user, but it is not required to do so.
- `"SentenceGroup"` – A group of related alternative actions is a sentence group if the item referenced by the action is the same in all the actions, and each action simply constitutes a different variation on how to specify the details for that item. For example, two actions that could be in a SentenceGroup are "aspirin, 500 mg, 2 times per day" and "aspirin, 300 mg, 3 times per day". In both cases, aspirin is the item referenced by the action, and the two actions represent two different options for how aspirin might be ordered for the patient. Note that a SentenceGroup would almost always have an associated selection behavior of "AtMostOne", unless it's a required action, in which case, it would be "ExactlyOne".

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ `GroupOrganizationBehaviorTypeCore` [392] (restriction)

└ **GroupOrganizationBehaviorTypeExt**

Derivation: restriction of `GroupOrganizationBehaviorTypeCore`

XML Source

```
<xs:simpleType name="GroupOrganizationBehaviorTypeExt">
  <xs:restriction base="GroupOrganizationBehaviorTypeCore"/>
</xs:simpleType>
```

simpleType "GroupSelectionBehaviorType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

```
("Any" | "All" | "AllOrNone" | "ExactlyOne" | "AtMostOne" | "OneOrMore") | ("Any" | "All" | "AllOrNone"
| "ExactlyOne" | "AtMostOne" | "OneOrMore")
```

Annotation

Defines selection behavior of a group: specifies the number of selectable items in the group that may be selected by the end user when the items of the group are displayed.

Type Definition Detail

Type Derivation Tree

union of (`GroupSelectionBehaviorTypeCore` | `GroupSelectionBehaviorTypeExt`)

└ **GroupSelectionBehaviorType**

Derivation: by union

Member Types

1. `GroupSelectionBehaviorTypeCore`
2. `GroupSelectionBehaviorTypeExt`

XML Source (w/o annotations (1))

```
<xs:simpleType name="GroupSelectionBehaviorType">
  <xs:union memberTypes="GroupSelectionBehaviorTypeCore GroupSelectionBehaviorTypeExt"/>
</xs:simpleType>
```

simpleType "GroupSelectionBehaviorTypeCore"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration:

`"Any"`	– Any number of the items in the group may be chosen, from zero to all.
`"All"`	– All the items in the group must be selected as a single unit.
`"AllOrNone"`	– All the items in the group are meant to be chosen as a single unit: either all must be selected by the end user, or none may

be selected.
"ExactlyOne" – The end user must choose one and only one of the selectable items in the group. The user may not choose none of the items in the group.
"AtMostOne" – The end user may choose zero or at most one of the items in the group.

"OneOrMore" – The end user must choose a minimum of one, and as many additional as desired.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **GroupSelectionBehaviorTypeCore**

Derivation: restriction of xs:string
Facets: **enumeration:** "Any" – Any number of the items in the group may be chosen, from zero to all.

"All" – All the items in the group must be selected as a single unit.

"AllOrNone" – All the items in the group are meant to be chosen as a single unit: either all must be selected by the end user, or none may be selected.
"ExactlyOne" – The end user must choose one and only one of the selectable items in the group. The user may not choose none of the items in the group.
"AtMostOne" – The end user may choose zero or at most one of the items in the group.

"OneOrMore" – The end user must choose a minimum of one, and as many additional as desired.

XML Source (w/o annotations (6))

```
<xs:simpleType name="GroupSelectionBehaviorTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Any"/>
    <xs:enumeration value="All"/>
    <xs:enumeration value="AllOrNone"/>
    <xs:enumeration value="ExactlyOne"/>
    <xs:enumeration value="AtMostOne"/>
    <xs:enumeration value="OneOrMore"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "GroupSelectionBehaviorTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Any" – Any number of the items in the group may be chosen, from zero to all.

"All" – All the items in the group must be selected as a single unit.

"AllOrNone" – All the items in the group are meant to be chosen as a single unit: either all must be selected by the end user, or none may be selected.
"ExactlyOne" – The end user must choose one and only one of the selectable items in the group. The user may not choose none of the items in the group.

"AtMostOne" – The end user may choose zero or at most one of the items in the group.

"OneOrMore" – The end user must choose a minimum of one, and as many additional as desired.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ GroupSelectionBehaviorTypeCore [394] (restriction)

└ **GroupSelectionBehaviorTypeExt**

Derivation: restriction of GroupSelectionBehaviorTypeCore

XML Source

```
<xs:simpleType name="GroupSelectionBehaviorTypeExt">
  <xs:restriction base="GroupSelectionBehaviorTypeCore"/>
</xs:simpleType>
```

simpleType "PrecheckBehaviorType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

("Yes" | "No") | ("Yes" | "No")

Annotation

Defines selection frequency behavior for an action or group; i.e., for most frequently selected items, the end-user system may provide convenience options in the UI (such as pre-selection) in order to (1) communicate to the end user what the most frequently selected item is, or should, be in a particular context, and (2) save the end user time.

Type Definition Detail

Type Derivation Tree

union of (PrecheckBehaviorTypeCore | PrecheckBehaviorTypeExt)

└ **PrecheckBehaviorType**

Derivation: by union

Member Types

1. PrecheckBehaviorTypeCore
2. PrecheckBehaviorTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="PrecheckBehaviorType">
  <xs:union memberTypes="PrecheckBehaviorTypeCore PrecheckBehaviorTypeExt"/>
</xs:simpleType>
```

simpleType "PrecheckBehaviorTypeCore"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration: `"Yes"` – An action with this behavior is one of the most frequent actions that is, or should be, included by an end user, for the particular context in which the action occurs. The system displaying the action to the end user should consider "pre-checking" such an action as a convenience for the user.

`"No"` – An action with this behavior is one of the less frequent actions included by the end user, for the particular context in which the action occurs. The system displaying the actions to the end user would typically not "pre-check" such an action.

Type Definition Detail

Type Derivation Tree

`xs:string` (restriction)

└ **PrecheckBehaviorTypeCore**

Derivation: restriction of `xs:string`

Facets: **enumeration:** `"Yes"` – An action with this behavior is one of the most frequent actions that is, or should be, included by an end user, for the particular context in which the action occurs. The system displaying the action to the end user should consider "pre-checking" such an action as a convenience for the user.

`"No"` – An action with this behavior is one of the less frequent actions included by the end user, for the particular context in which the action occurs. The system displaying the actions to the end user would typically not "pre-check" such an action.

XML Source (w/o annotations (2))

```
<xs:simpleType name="PrecheckBehaviorTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Yes"/>
    <xs:enumeration value="No"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "PrecheckBehaviorTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration: `"Yes"` – An action with this behavior is one of the most frequent actions that is, or should be, included by an end user, for the particular context in which the action occurs. The system displaying the action to the end user should consider "pre-checking" such an action as a convenience for the user.

`"No"` – An action with this behavior is one of the less frequent actions included by the end user, for the particular context in which the action occurs. The system displaying the actions to the end user would typically not "pre-check" such an action.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ PrecheckBehaviorTypeCore [397] (restriction)

└ **PrecheckBehaviorTypeExt**

Derivation: restriction of PrecheckBehaviorTypeCore

XML Source

```
<xs:simpleType name="PrecheckBehaviorTypeExt">
  <xs:restriction base="PrecheckBehaviorTypeCore"/>
</xs:simpleType>
```

simpleType "RangeConstraintType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

("Minimum" | "Maximum" | "List" | "Component") | ("Minimum" | "Maximum" | "List" | "Component")

Annotation

The enumeration of different types of range constraints on values.

Type Definition Detail

Type Derivation Tree

union of (RangeConstraintTypeCore | RangeConstraintTypeExt)

└ **RangeConstraintType**

Derivation: by union

Member Types

1. RangeConstraintTypeCore
2. RangeConstraintTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="RangeConstraintType">
  <xs:union memberTypes="RangeConstraintTypeCore RangeConstraintTypeExt"/>
</xs:simpleType>
```

simpleType "RangeConstraintTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Minimum" – The constraint specifies the lower bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only.
"Maximum" – The constraint specifies the upper bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only.
"List" – The constraint restricts the value to items from a list.

"Component"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **RangeConstraintTypeCore**

Derivation: restriction of xs:string
Facets: **enumeration:** "Minimum" – The constraint specifies the lower bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only.
"Maximum" – The constraint specifies the upper bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only.
"List" – The constraint restricts the value to items from a list.

"Component"

XML Source (w/o annotations (3))

```
<xs:simpleType name="RangeConstraintTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Minimum"/>
    <xs:enumeration value="Maximum"/>
    <xs:enumeration value="List"/>
    <xs:enumeration value="Component"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "RangeConstraintTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Minimum" – The constraint specifies the lower bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only.
"Maximum" – The constraint specifies the upper bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only.
"List" – The constraint restricts the value to items from a list.

"Component"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ RangeConstraintTypeCore [398] (restriction)

└ **RangeConstraintTypeExt**

Derivation: restriction of RangeConstraintTypeCore

XML Source

```
<xs:simpleType name="RangeConstraintTypeExt">
  <xs:restriction base="RangeConstraint TypeCore"/>
</xs:simpleType>
```

simpleType "RequestCardinality"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Single", "Multiple"

Annotation

RequestCardinality defines the expected cardinality of the request, single or multiple.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **RequestCardinality**

Derivation: restriction of xs:string
Facets: **enumeration:** "Single", "Multiple"

XML Source (w/o annotations (1))

```
<xs:simpleType name="RequestCardinality">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Single"/>
    <xs:enumeration value="Multiple"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "RequiredBehaviorType"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

("Must" | "Could" | "MustUnlessDocumented") | ("Must" | "Could" | "MustUnlessDocumented")

Annotation

Defines requiredness behavior for selecting an action or an action group; i.e., whether the action or action group is required or optional.

Type Definition Detail

Type Derivation Tree

union of (RequiredBehaviorTypeCore | RequiredBehaviorTypeExt)

└ **RequiredBehaviorType**

Derivation: by union

Member Types

1. RequiredBehaviorTypeCore
2. RequiredBehaviorTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="RequiredBehaviorType">
  <xs:union memberTypes="RequiredBehaviorTypeCore RequiredBehaviorTypeExt"/>
</xs:simpleType>
```

simpleType "RequiredBehaviorTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration:

"Must"	– An action with this behavior must be included in the actions processed by the end user; the end user may not choose not to include this action.
"Could"	– An action with this behavior may be included in the set of actions processed by the end user.
"MustUnlessDocumented"	– An action with this behavior must be included in the set of actions processed by the end user, unless the end user provides documentation as to why the action was not included.

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **RequiredBehaviorTypeCore**

Derivation: restriction of `xs:string`

Facets: **enumeration:**

- `"Must"` – An action with this behavior must be included in the actions processed by the end user; the end user may not choose not to include this action.
- `"Could"` – An action with this behavior may be included in the set of actions processed by the end user.
- `"MustUnlessDocumented"` – An action with this behavior must be included in the set of actions processed by the end user, unless the end user provides documentation as to why the action was not included.

XML Source (w/o annotations (3))

```
<xs:simpleType name="RequiredBehaviorTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Must"/>
    <xs:enumeration value="Could"/>
    <xs:enumeration value="MustUnlessDocumented"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "RequiredBehaviorTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration:

- `"Must"` – An action with this behavior must be included in the actions processed by the end user; the end user may not choose not to include this action.
- `"Could"` – An action with this behavior may be included in the set of actions processed by the end user.
- `"MustUnlessDocumented"` – An action with this behavior must be included in the set of actions processed by the end user, unless the end user provides documentation as to why the action was not included.

Type Definition Detail

Type Derivation Tree

```
xs:string (restriction)
  └ RequiredBehaviorTypeCore [401] (restriction)
      └ RequiredBehaviorTypeExt
```

Derivation: restriction of `RequiredBehaviorTypeCore`

XML Source

```
<xs:simpleType name="RequiredBehaviorTypeExt">
  <xs:restriction base="RequiredBehaviorTypeCore"/>
</xs:simpleType>
```

simpleType "ResourceRelationshipType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

```
("AdaptedFrom" | "AssociatedResource" | "DependsOn" | "DerivedFrom" | "SimilarTo" | "VersionOf") |
("AdaptedFrom" | "AssociatedResource" | "DependsOn" | "DerivedFrom" | "SimilarTo" | "VersionOf")
```

Annotation

A specific status is associated with each version of an artifact.

See the Implementation Guide for a state-transition diagram showing the legal transitions from each state; each state is equivalent to particular status.

Type Definition Detail

Type Derivation Tree

union of (ResourceRelationshipTypeCore | ResourceRelationshipTypeExt)

└ **ResourceRelationshipType**

Derivation: by union

Member Types

1. ResourceRelationshipTypeCore
2. ResourceRelationshipTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="ResourceRelationshipType">
  <xs:union memberTypes="ResourceRelationshipTypeCore ResourceRelationshipTypeExt"/>
</xs:simpleType>
```

simpleType "ResourceRelationshipTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "AdaptedFrom", "AssociatedResource", "DependsOn", "DerivedFrom", "SimilarTo", "VersionOf"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **ResourceRelationshipTypeCore**

Derivation: restriction of xs:string
Facets: **enumeration:** "AdaptedFrom", "AssociatedResource", "DependsOn", "DerivedFrom", "SimilarTo", "VersionOf"

XML Source

```
<xs:simpleType name="ResourceRelationshipTypeCore">
```

```
  <xs:restriction base="xs:string">
    <xs:enumeration value="AdaptedFrom"/>
    <xs:enumeration value="AssociatedResource"/>
    <xs:enumeration value="DependsOn"/>
    <xs:enumeration value="DerivedFrom"/>
    <xs:enumeration value="SimilarTo"/>
    <xs:enumeration value="VersionOf"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "ResourceRelationshipTypeExt"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

enumeration of `xs:string`

Simple Content Restrictions:

Enumeration: `"AdaptedFrom", "AssociatedResource", "DependsOn", "DerivedFrom", "SimilarTo", "VersionOf"`

Annotation

To add new items to the enumeration ArtifactStatus, comment out or delete the restriction on ArtifactStatusTypeCore.(Optionally, also remove
the directive above to include the artifactstatustypecore.xsd).
Now, add a new restriction such as on string and define new enumerated types. See the commented out example below. !!!
BKM get example from ArtifactLifeCycleEventTypeExt

Type Definition Detail

Type Derivation Tree

```
xs:string (restriction)
  └ ResourceRelationshipTypeCore [403] (restriction)
      └ ResourceRelationshipTypeExt
```

Derivation: restriction of `ResourceRelationshipTypeCore`

XML Source (w/o annotations (1))

```
<xs:simpleType name="ResourceRelationshipTypeExt">
  <xs:restriction base="ResourceRelationshipTypeCore"/>
</xs:simpleType>
```

simpleType "ValueType"

Namespace: `urn:hl7-org:knowledgeartifact:r1`

Simple Content Model

```
("Boolean" | "Code" | "CodedOrdinal" | "EntityName" | "Identifier" | "Integer" | "IntegerInterval" |
"Period" | "PhysicalQuantity" | "PhysicalQuantityInterval" | "QuantityInterval" | "Ratio" | "Real" |
"RealInterval" | "SimpleCode" | "String" | "Timestamp" | "TimestampInterval" | "URL") | ("Boolean" |
"Code" | "CodedOrdinal" | "EntityName" | "Identifier" | "Integer" | "IntegerInterval" | "Period" |
"PhysicalQuantity" | "PhysicalQuantityInterval" | "QuantityInterval" | "Ratio" | "Real" | "RealInterval"
| "SimpleCode" | "String" | "Timestamp" | "TimestampInterval" | "URL")
```

Annotation

A specification of a constraint on the range of values for an item.

Type Definition Detail

Type Derivation Tree

union of (ValueTypeCore | ValueTypeExt)

└ **ValueType**

Derivation: by union

Member Types

1. ValueTypeCore
2. ValueTypeExt

XML Source (w/o annotations (1))

```
<xs:simpleType name="ValueType">
  <xs:union memberTypes="ValueTypeCore ValueTypeExt"/>
</xs:simpleType>
```

simpleType "ValueTypeCore"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Boolean", "Code", "CodedOrdinal", "EntityName", "Identifier", "Integer", "IntegerInterval", "Period", "PhysicalQuantity", "PhysicalQuantityInterval", "QuantityInterval", "Ratio", "Real", "RealInterval", "SimpleCode", "String", "Timestamp", "TimestampInterval", "URL"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ **ValueTypeCore**

Derivation: restriction of xs:string

Facets: **enumeration:** "Boolean", "Code", "CodedOrdinal", "EntityName", "Identifier", "Integer", "IntegerInterval", "Period", "PhysicalQuantity", "PhysicalQuantityInterval", "QuantityInterval", "Ratio", "Real", "RealInterval", "SimpleCode", "String", "Timestamp", "TimestampInterval", "URL"

XML Source

```
<xs:simpleType name="ValueTypeCore">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Boolean"/>
    <xs:enumeration value="Code"/>
    <xs:enumeration value="CodedOrdinal"/>
    <xs:enumeration value="EntityName"/>
    <xs:enumeration value="Identifier"/>
    <xs:enumeration value="Integer"/>
    <xs:enumeration value="IntegerInterval"/>
    <xs:enumeration value="Period"/>
    <xs:enumeration value="PhysicalQuantity"/>
    <xs:enumeration value="PhysicalQuantityInterval"/>
    <xs:enumeration value="QuantityInterval"/>
    <xs:enumeration value="Ratio"/>
    <xs:enumeration value="Real"/>
    <xs:enumeration value="RealInterval"/>
    <xs:enumeration value="SimpleCode"/>
    <xs:enumeration value="String"/>
    <xs:enumeration value="Timestamp"/>
    <xs:enumeration value="TimestampInterval"/>
    <xs:enumeration value="URL"/>
  </xs:restriction>
</xs:simpleType>
```

simpleType "ValueTypeExt"

Namespace: urn:hl7-org:knowledgeartifact:r1

Simple Content Model

enumeration of xs:string

Simple Content Restrictions:

Enumeration: "Boolean", "Code", "CodedOrdinal", "EntityName", "Identifier", "Integer", "IntegerInterval", "Period", "PhysicalQuantity", "PhysicalQuantityInterval", "QuantityInterval", "Ratio", "Real", "RealInterval", "SimpleCode", "String", "Timestamp", "TimestampInterval", "URL"

Type Definition Detail

Type Derivation Tree

xs:string (restriction)

└ ValueTypeCore [405] (restriction)

└ **ValueTypeExt**

Derivation: restriction of ValueTypeCore

XML Source

```
<xs:simpleType name="ValueTypeExt">
  <xs:restriction base="ValueTypeCore"/>
</xs:simpleType>
```

Schema "action.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (24 local), complexTypes (10)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\action.xsd

Imports Schemas (1):
datatypes.xsd

Includes Schemas (10):
actor.xsd, base.xsd, behavior.xsd, behaviortype.xsd, catalogitem.xsd, clinicalexpression.xsd, condition.xsd, eventtype.xsd, expression.xsd, literalexpression.xsd

Included in Schemas (1):
knowledgedocument.xsd

Annotation

This file defines the Action types used to describe guidance within a knowledge artifact.

Schema "actor.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (1 local), complexTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: qualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\actor.xsd

Includes Schemas (1):
expression.xsd

Included in Schemas (1):
action.xsd

Annotation

This file defines the Actor type.

Schema "artifactlifecycleeventtype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\artifactlifecycleeventtype.xsd

Includes Schemas (2):
artifactlifecycleeventtypecore.xsd, artifactlifecycleeventtypeext.xsd

Included in Schemas (1):
metadata.xsd

Annotation

This file defines the ArtifactLifeCycleEventType enumeration.

Schema "artifactlifecycleeventtypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\artifactlifecycleeventtypecore.xsd

Included in Schemas (2):
artifactlifecycleeventtype.xsd, artifactlifecycleeventtypeext.xsd

Annotation

This file defines the core values available within the ArtifactLifeCycleEventType enumeration.

Schema "artifactlifecycleeventtypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\artifactlifecycleeventtypeext.xsd

Includes Schemas (1):
artifactlifecycleeventtypecore.xsd

Included in Schemas (1):
artifactlifecycleeventtype.xsd

Annotation

This file allows organizations to extend the enumeration ArtifactLifeCycleEventType.

Schema "artifactstatustype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\artifactstatustype.xsd

Includes Schemas (2):
artifactstatustypecore.xsd, artifactstatustypeext.xsd

Included in Schemas (1):
metadata.xsd

Annotation

This file defines the ArtifactStatusType enumeration.

Schema "artifactstatustypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\artifactstatustypecore.xsd

Included in Schemas (2):
artifactstatustype.xsd, artifactstatustypeext.xsd

Annotation

This file defines the core values available within the ArtifactStatusType enumeration.

Schema "artifactstatustypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\artifactstatustypeext.xsd

Includes Schemas (1):
artifactstatustypecore.xsd

Included in Schemas (1):
artifactstatustype.xsd

Annotation

This file allows organizations to extend the enumeration ArtifactStatusType.

Schema "artifacttype.xsd"

Target Namespace:

urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\artifacttype.xsd

Includes Schemas (2):
artifacttypecore.xsd, artifacttypeext.xsd

Included in Schemas (1):
metadata.xsd

Annotation

This file defines the ArtifactType enumeration.

Schema "artifacttypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\artifacttypecore.xsd

Included in Schemas (2):
artifacttype.xsd, artifacttypeext.xsd

Annotation

This file defines the core values available within the ArtifactType enumeration.

Schema "artifacttypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\artifacttypeext.xsd

Includes Schemas (1):
artifacttypecore.xsd

Included in Schemas (1):
artifacttype.xsd

Annotation

This file allows organizations to extend the enumeration ArtifactType.

Schema "base.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (24 local), complexTypes (10)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\base.xsd

Imports Schemas (1):
datatypes.xsd

Includes Schemas (1):
resourcerelationshiptype.xsd

Included in Schemas (3):
action.xsd, catalogitem.xsd, metadata.xsd

Annotation

This file defines the base types used in defining knowledge artifacts.

Schema "behavior.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (1 local), complexTypes (6)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\behavior.xsd

Includes Schemas (2):
behaviortype.xsd, expression.xsd

Included in Schemas (1):
action.xsd

Annotation

This file defines the behaviors used within knowledge documents.

Schema "behaviortype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (4)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\behaviortype.xsd

Includes Schemas (2):
behaviortypecore.xsd, behaviortypeext.xsd

Included in Schemas (2):
action.xsd, behavior.xsd

Annotation

This file defines the BehaviorType enumeration.

Schema "behaviortypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (4)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\behaviortypecore.xsd

Included in Schemas (2):
behaviortype.xsd, behaviortypeext.xsd

Annotation

This file defines the core values available within the BehaviorType enumeration.

Schema "behaviortypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (4)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\behaviortypeext.xsd

Includes Schemas (1):
behaviortypecore.xsd

Included in Schemas (1):
behaviortype.xsd

Annotation

This file allows organizations to extend the enumeration BehaviorType.

Schema "cardinalitytype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\cardinalitytype.xsd

Included in Schemas (1):
catalogitem.xsd

Annotation

This file defines the Cardinality enumeration.

Schema "catalogitem.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (1 global + 17 local), complexTypes (6)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\catalogitem.xsd

Imports Schemas (1):
datatypes.xsd

Includes Schemas (6):
base.xsd, cardinalitytype.xsd, clinicalexpression.xsd, literalexpression.xsd, rangeconstrainttype.xsd, valuetype.xsd

Included in Schemas (1):
action.xsd

Annotation

This file defines catalog items used to describe questions and responses for the documentation template knowledge artifact type.

Schema "clinicalexpression.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (8 local), complexTypes (7), simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\clinicalexpression.xsd

Imports Schemas (1):
datatypes.xsd

Includes Schemas (4):
dataeventtype.xsd, eventtype.xsd, expression.xsd, literalexpression.xsd

Included in Schemas (3):
action.xsd, catalogitem.xsd, knowledgedocument.xsd

Annotation

This file defines the RequestBase extensions that introduce clinically relevant dependencies such as terminology, relevant time, and value set considerations.

Schema "condition.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (3 local), complexTypes (2)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\condition.xsd

Includes Schemas (2):
conditionroletype.xsd, expression.xsd

Included in Schemas (2):
action.xsd, knowledgedocument.xsd

Annotation

This file defines the condition components used in knowledge artifacts.

Schema "conditionroletype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\conditionroletype.xsd

Includes Schemas (2):
conditionroletypecore.xsd, conditionroletypeext.xsd

Included in Schemas (1):
condition.xsd

Annotation

This file defines the ConditionRoleType enumeration.

Schema "conditionroletypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\conditionroletypecore.xsd

Included in Schemas (2):
conditionroletype.xsd, conditionroletypeext.xsd

Annotation

This file defines the core values available within the ConditionRoleType enumeration.

Schema "conditionroletypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:

simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\conditionroletypeext.xsd

Includes Schemas (1):
conditionroletypecore.xsd

Included in Schemas (1):
conditionroletype.xsd

Annotation

This file allows organizations to extend the enumeration ConditionRoleType.

Schema "contributortype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\contributortype.xsd

Includes Schemas (2):
contributortypecore.xsd, contributortypeext.xsd

Included in Schemas (1):
metadata.xsd

Annotation

This file defines the ContributorType enumeration.

Schema "contributortypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\contributortypecore.xsd

Included in Schemas (2):
contributortype.xsd, contributortypeext.xsd

Annotation

This file defines the core values available within the ContributorType enumeration.

Schema "contributortypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\contributortypeext.xsd

Includes Schemas (1):
contributortypecore.xsd

Included in Schemas (1):
contributortype.xsd

Annotation

This file allows organizations to extend the enumeration ContributorType.

Schema "coveragetype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\coveragetype.xsd

Includes Schemas (2):
coveragetypecore.xsd, coveragetypeext.xsd

Included in Schemas (1):
metadata.xsd

Annotation

This file defines the CoverageType enumeration.

Schema "coveragetypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\coveragetypecore.xsd

Included in Schemas (2):
coveragetype.xsd, coveragetypeext.xsd

Annotation

This file defines the core values available within the CoverageType enumeration.

Schema "coveragetypeext.xsd"

Target Namespace:

urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\coveragetypeext.xsd

Includes Schemas (1):
coveragetypecore.xsd

Included in Schemas (1):
coveragetype.xsd

Annotation

This file allows organizations to extend the enumeration CoverageType.

Schema "dataeventtype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\dataeventtype.xsd

Includes Schemas (2):
dataeventtypecore.xsd, dataeventtypeext.xsd

Included in Schemas (1):
clinicalexpression.xsd

Annotation

This file defines the DataEventType enumeration.

Schema "dataeventtypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\dataeventtypecore.xsd

Included in Schemas (2):
dataeventtype.xsd, dataeventtypeext.xsd

Annotation

This file defines the core values available within the DataEventType enumeration.

Schema "dataeventtypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\dataeventtypeext.xsd

Includes Schemas (1):
dataeventtypecore.xsd

Included in Schemas (1):
dataeventtype.xsd

Annotation

This file allows organizations to extend the enumeration DataEventType.

Schema "datatypes.xsd"

Target Namespace:
urn:hl7-org:cdsdt:r2

Defined Components:
elements (29 local), complexTypes (30), simpleTypes (20)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\common\datatypes.xsd

Imported by Schemas (6):
action.xsd, base.xsd, catalogitem.xsd, clinicalexpression.xsd, literalexpression.xsd, metadata.xsd

Annotation

Specifies data types used. The data types are a simplified/constrained version of the HL7 version 3 datatypes specification, release 2, which is itself based on the implementable specification of ISO 21090 data types.

Schema "eventtype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\eventtype.xsd

Includes Schemas (2):
eventtypecore.xsd, eventtypeext.xsd

Included in Schemas (2):
action.xsd, clinicalexpression.xsd

Annotation

This file defines the EventType enumeration.

Schema "eventtypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\eventtypecore.xsd

Included in Schemas (2):
eventtype.xsd, eventtypeext.xsd

Annotation

This file defines the core values available within the EventType enumeration.

Schema "eventtypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\eventtypeext.xsd

Includes Schemas (1):
eventtypecore.xsd

Included in Schemas (1):
eventtype.xsd

Annotation

This file allows organizations to extend the enumeration EventType.

Schema "expression.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (64 local), complexTypes (118), simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\expression.xsd

Included in Schemas (7):
action.xsd, actor.xsd, behavior.xsd, clinicalexpression.xsd, condition.xsd, knowledgedocument.xsd, literalexpression.xsd

Annotation

This file defines the core expression components used throughout knowledge artifacts. The core expression components defined here are intended to be general purpose, and beyond the datatypes in the cdsdt namespace, do not reference anything specific to the clinical decision support domain. This layering is intended to isolate the syntactic elements of a general purpose expression language from the more frequently varying elements of clinical decision support.

Schema "knowledgedocument.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (1 global + 13 local), complexTypes (3)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\knowledgedocument.xsd

Includes Schemas (6):
action.xsd, clinicalexpression.xsd, condition.xsd, expression.xsd, literalexpression.xsd, metadata.xsd

Annotation

This file defines the root knowledge document type and element.

Schema "literalexpression.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (14 local), complexTypes (20)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\literalexpression.xsd

Imports Schemas (1):
datatypes.xsd

Includes Schemas (1):
expression.xsd

Included in Schemas (4):
action.xsd, catalogitem.xsd, clinicalexpression.xsd, knowledgedocument.xsd

Annotation

This file defines additional expressions that provide syntactic short-hands for literals for each of the base data types.

Schema "metadata.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
elements (43 local), complexTypes (6)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:

C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\metadata.xsd

Imports Schemas (1):
datatypes.xsd

Includes Schemas (6):
artifactlifecycleeventtype.xsd, artifactstatustype.xsd, artifacttype.xsd, base.xsd, contributortype.xsd, coveragetype.xsd

Included in Schemas (1):
knowledgedocument.xsd

Annotation

This file defines the meta data components used within a knowledge artifact.

Schema "rangeconstrainttype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\rangeconstrainttype.xsd

Includes Schemas (2):
rangeconstrainttypecore.xsd, rangeconstrainttypeext.xsd

Included in Schemas (1):
catalogitem.xsd

Annotation

This file defines the RangeConstraintType enumeration.

Schema "rangeconstrainttypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\rangeconstrainttypecore.xsd

Included in Schemas (2):
rangeconstrainttype.xsd, rangeconstrainttypeext.xsd

Annotation

This file defines the core values available within the RangeCosntraintType enumeration.

Schema "rangeconstrainttypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\rangeconstrainttypeext.xsd

Includes Schemas (1):
rangeconstrainttypecore.xsd

Included in Schemas (1):
rangeconstrainttype.xsd

Annotation

This file allows organizations to extend the enumeration RangeConstraintType.

Schema "resourcerelationshiptype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\resourcerelationshiptype.xsd

Includes Schemas (2):
resourcerelationshiptypecore.xsd, resourcerelationshiptypeext.xsd

Included in Schemas (1):
base.xsd

Annotation

This file defines the ResourceRelationshipType enumeration.

Schema "resourcerelationshiptypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\resourcerelationshiptypecore.xsd

Included in Schemas (2):
resourcerelationshiptype.xsd, resourcerelationshiptypeext.xsd

Annotation

This file defines the core values available within the ResourceRelationshipType enumeration.

Schema "resourcerelationshiptypeext.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\resourcerelationshiptypeext.xsd

Includes Schemas (1):
resourcerelationshiptypecore.xsd

Included in Schemas (1):
resourcerelationshiptype.xsd

Annotation

This file allows organizations to extend the enumeration ResourceRelationshipType.

Schema "valuetype.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\valuetype.xsd

Includes Schemas (2):
valuetypecore.xsd, valuetypeext.xsd

Included in Schemas (1):
catalogitem.xsd

Annotation

This file defines the ValueType enumeration.

Schema "valuetypecore.xsd"

Target Namespace:
urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\enum\valuetypecore.xsd

Included in Schemas (2):
valuetype.xsd, valuetypeext.xsd

Annotation

This file defines the core values available within the ValueType enumeration.

Schema "valuetypeext.xsd"

Target Namespace:

urn:hl7-org:knowledgeartifact:r1

Defined Components:
simpleTypes (1)

Default Namespace-Qualified Form:
Local Elements: qualified; Local Attributes: unqualified

Schema Location:
C:\Users\Bryn\Documents\Src\HeD\Source\src\main\schema\knowledgeartifact\ext\valuetypeext.xsd

Includes Schemas (1):
valuetypecore.xsd

Included in Schemas (1):
valuetype.xsd

Annotation

This file allows organizations to extend the enumeration ValueType.

Namespace Bindings

Prefix	Namespace URI / Binding Location
-	urn:hl7-org:cdsdt:r2 **File:** datatypes.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** action.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** actor.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifactlifecycleeventtype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifactlifecycleeventtypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifactlifecycleeventtypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifactstatustype.xsd **Element:** <xs:schema ... >

-	urn:hl7-org:knowledgeartifact:r1 **File:** artifactstatustypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifactstatustypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifacttype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifacttypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** artifacttypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** base.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** behavior.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** behaviortype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** behaviortypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** behaviortypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** cardinalitytype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** catalogitem.xsd **Element:** <xs:schema ... >

-	urn:hl7-org:knowledgeartifact:r1 **File:** clinicalexpression.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** condition.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** conditionroletype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** conditionroletypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** conditionroletypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** contributortype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** contributortypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** contributortypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** coveragetype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** coveragetypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** coveragetypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** dataeventtype.xsd **Element:** <xs:schema ... >

-	urn:hl7-org:knowledgeartifact:r1 **File:** dataeventtypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** dataeventtypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** eventtype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** eventtypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** eventtypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** expression.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** knowledgedocument.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** literalexpression.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** metadata.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** rangeconstrainttype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** rangeconstrainttypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** rangeconstrainttypeext.xsd **Element:** <xs:schema ... >

-	urn:hl7-org:knowledgeartifact:r1 **File:** resourcerelationshiptype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** resourcerelationshiptypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** resourcerelationshiptypeext.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** valuetype.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** valuetypecore.xsd **Element:** <xs:schema ... >
-	urn:hl7-org:knowledgeartifact:r1 **File:** valuetypeext.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** action.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** base.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** behavior.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** catalogitem.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** clinicalexpression.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** condition.xsd **Element:** <xs:schema ... >

dt	urn:hl7-org:cdsdt:r2 **File:** datatypes.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** expression.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** knowledgedocument.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** literalexpression.xsd **Element:** <xs:schema ... >
dt	urn:hl7-org:cdsdt:r2 **File:** metadata.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** action.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** base.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** behavior.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** catalogitem.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** clinicalexpression.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** condition.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** expression.xsd **Element:** <xs:schema ... >

sch	http://purl.oclc.org/dsdl/schematron **File:** knowledgedocument.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** literalexpression.xsd **Element:** <xs:schema ... >
sch	http://purl.oclc.org/dsdl/schematron **File:** metadata.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifactlifecycleeventtype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifactlifecycleeventtypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifactlifecycleeventtypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifactstatustype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifactstatustypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifactstatustypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifacttype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifacttypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** artifacttypeext.xsd **Element:** <xs:schema ... >

xhtml	http://www.w3.org/1999/xhtml **File:** base.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** behavior.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** behaviortype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** behaviortypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** behaviortypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** cardinalitytype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** catalogitem.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** clinicalexpression.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** condition.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** conditionroletype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** conditionroletypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** conditionroletypeext.xsd **Element:** <xs:schema ... >

xhtml	http://www.w3.org/1999/xhtml **File:** contributortype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** contributortypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** contributortypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** coveragetype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** coveragetypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** coveragetypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** dataeventtype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** dataeventtypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** dataeventtypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** eventtype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** eventtypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** eventtypeext.xsd **Element:** <xs:schema ... >

xhtml	http://www.w3.org/1999/xhtml **File:** knowledgedocument.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** metadata.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** rangeconstrainttype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** rangeconstrainttypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** rangeconstrainttypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** resourcerelationshiptype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** resourcerelationshiptypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** resourcerelationshiptypeext.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** valuetype.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** valuetypecore.xsd **Element:** <xs:schema ... >
xhtml	http://www.w3.org/1999/xhtml **File:** valuetypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** action.xsd **Element:** <xs:schema ... >

xs	http://www.w3.org/2001/XMLSchema **File:** actor.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifactlifecycleeventtype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifactlifecycleeventtypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifactlifecycleeventtypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifactstatustype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifactstatustypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifactstatustypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifacttype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifacttypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** artifacttypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** base.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** behavior.xsd **Element:** <xs:schema ... >

xs	http://www.w3.org/2001/XMLSchema **File:** behaviortype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** behaviortypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** behaviortypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** cardinalitytype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** catalogitem.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** clinicalexpression.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** condition.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** conditionroletype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** conditionroletypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** conditionroletypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** contributortype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** contributortypecore.xsd **Element:** <xs:schema ... >

xs	http://www.w3.org/2001/XMLSchema **File:** contributortypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** coveragetype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** coveragetypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** coveragetypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** dataeventtype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** dataeventtypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** dataeventtypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** datatypes.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** eventtype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** eventtypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** eventtypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** expression.xsd **Element:** <xs:schema ... >

xs	http://www.w3.org/2001/XMLSchema **File:** knowledgedocument.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** literalexpression.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** metadata.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** rangeconstrainttype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** rangeconstrainttypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** rangeconstrainttypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** resourcerelationshiptype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** resourcerelationshiptypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** resourcerelationshiptypeext.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** valuetype.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** valuetypecore.xsd **Element:** <xs:schema ... >
xs	http://www.w3.org/2001/XMLSchema **File:** valuetypeext.xsd **Element:** <xs:schema ... >

6 APPENDIX A – REFERENCED DOCUMENTS

The reference documents specific to the CDS Knowledge Artifact are presented in the table below. This implementation guide is built based on the premise of reusing as much existing material as possible, and as such, the list of references is fairly extensive, and may be expanded upon as the implementation guide is developed and balloted.

Reference Name	Location	How reference was used in development of CDS Knowledge Artifact Implementation Guide
S&I Framework Health eDecisions - CDS Artifact Sharing Use Case	http://wiki.siframework.org/file/view/SIFramework_HeD_UC1_CDSArtifactSharing_v1.0.docx	The use case serves as the baseline for all functional and system requirements associated with the CDS knowledge artifact, and provides scoping for the implementation guide
The Arden Syntax for Medical Logic Systems Version 2.7	http://www.hl7.org/documentcenter/private/standards/Arden/v27/Arden%20Syntax%202.7_PDF.zip	The Arden Syntax Version 2.8 is used as the primary reference source for the Expression section of this document. It also provides additional reference material on datatypes associated with the CDS Knowledge Artifact.
HL7 Virtual Medical Record for Clinical Decision Support (vMR-CDS) Logical Model, Release 2, Version 3.0	http://wiki.hl7.org/index.php?title=HL7_CDS_Standards	The Logical Model for the Virtual Medical Record (vMR) serves as the primary resource for the structure of the CDS Knowledge Artifact.
HL7 Virtual Medical Record for Clinical Decision Support (vMR-CDS) Templates, Release 1, Version 2.0	http://wiki.hl7.org/index.php?title=HL7_CDS_Standards	The Templates provide terminology and other constraints, applicable for the US Realm, on the use of the vMR in the CDS Knowledge Artifact.
HL7 Version 3 Implementation Guide: Virtual Medical Record for Clinical Decision Support (vMR-CDS) for GELLO, Release 1 Draft Standard for Trial Use	http://www.hl7.org/documentcenter/public/standards/dstu/V3IG_CDS_VMR_GELLO_DSTU_R1_2012APR.pdf	GELLO is included in this list of references due to the structure of the vMR/GELLO implementation guide that was balloted by HL7. The HeD initiative wishes to emulate several elements of that structure to assist implementers of the CDS Knowledge Artifact
Guidelines Element Model (GEM)	http://gem.med.yale.edu/default.htm	The Guidelines Element Model (GEM) is included within this implementation guide as one of the schemas that has been harmonized within the new CDS Knowledge Artifact schema.

© 2014 Health Level Seven International. All rights reserved

Reference Name	Location	How reference was used in development of CDS Knowledge Artifact Implementation Guide
Digital Infrastructure for the Learning Health System: The Foundation for Continuous Improvement in Health and Health Care: Workshop Series Summary. Institute of Medicine.	This PDF is available from The National Academies Press at: http://www.nap.edu/catalog.php?record_id=12912 www,nap.edu October 2011. ISBN 978-0-309-15416-1"	Cited as reference for learning health system in Executive Summary
AHRQ eRecs – Structuring Care Recommendations for Clinical Decision Support	http://images.ahrq.gov/publishedimages/communities/a_e/ahrq_funded_projects/projects/calendaryearupdateshtmlpages/2011_2900900022i2_osheroff_pdf_3.pdf	The AHRQ eRecs specification will be harmonized to facilitate representation of data elements and logical expressions in a structured, codified format, enabling further local processing into CDS rules.
Clinical Decision Support Consortium Level 3 XML examples	http://cdsportal.partners.org/CDSCSearch.aspx	As part of promoting greater CDS adoption with a wider community of CDS vendors, the Harmonized HeD schema also maps to the CDSC L3 schema. While this schema is not considered a formally balloted standard through a Standards Development Organization (SDO), it nevertheless contains valuable best practices and implementation experience from the field. L3 serves as the "container" for the harmonized schema.

TABLE 11 – APPENDIX A – LIST OF CDS REFERENCES FOR IMPLEMENTERS

© 2014 Health Level Seven International. All rights reserved

7 APPENDIX B – ACRONYMS

The following acronyms are referenced in this implementation guide:

Acronym	Definition/Description
CDS	Clinical Decision Support
CDSC L3	Clinical Decision Support Consortium Level 3
CREF	Allscripts Common Rule Engine Format (CREF) specification
DAM	Domain Analysis Model
EHR	Electronic Health Record
EMR	Electronic Medical Record
eRecs	AHRQ Electronic Recommendations
GEM	Guidelines Element Model
HIE	Health Information Exchange
HIT	Health Information Technology
HITECH Act	Health Information Technology for Economic and Clinical Health Act
HIPAA	Health Insurance Portability and Accountability Act
HeD	Health eDecisions
HITSP	Health Information Technology Standards Panel
HL7	Health Level 7
MU	Meaningful Use
ONC	Office of the National Coordinator for Health IT
S&I Framework	Standards & Interoperability Framework
vMR	HL7 Virtual Medical Record
XML	Extensible Markup Language
XSD	XML Schema

TABLE 12 – APPENDIX B – LIST OF ACRONYMS USED IN THIS GUIDE

© 2014 Health Level Seven International. All rights reserved

8 APPENDIX C – DEFINITIONS

Because the CDS Knowledge Artifact introduces new concepts and terms, an additional appendix of definitions is provided to support implementer understanding of terms that may be used in various sections of this document, specifically documentation and examples in Sections 4-6. This appendix DOES NOT include terms and definitions already provided in the S&I Framework HeD CDS Artifact Sharing Use Case

Term	Definition/Description
Abstract	An abstract element or complex type cannot used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.
Complex Type	A complex type is an XML element that contains other elements and/or attributes.
Enumeration	Used to limit an element to a set of constrained values – enumerations are used in the CDS Knowledge Artifact to define “value sets”
Simple Type	A simple type contains constraints and information about the values of attributes or text-only elements.
Value Set	A uniquely identifiable set of valid concept identifiers, where any concept identifier in a coded element can be tested to determine whether it is a member of the Value Set.

TABLE 13 – APPENDIX C – LIST OF DEFINITIONS

© 2014 Health Level Seven International. All rights reserved

9 APPENDIX D – HED SCHEMA FRAMEWORK

9.1 Overview

The HeD Artifact Utility is a simple command-line executable used to verify HeD Artifacts, and optionally translate them to another target format. The utility is built using the HeD Schema Framework, which provides a basis for the implementation of technologies that process and manipulate artifacts expressed in the HeD Schema. The framework was built as an open source project as part of the S&I Framework Health eDecisions initiative as a tool to help implement and verify the HeD Schema pilot phases of the initiative. The framework is freely distributed under an open source license, which can be found in the license.txt file at the root of the project. The project source code is hosted in a Google code repository found at the following URL:

http://code.google.com/p/health-e-decisions/

This section will discuss the architecture, design, and implementation of the various components of the HeD Schema framework. The content is intended to be useful for anyone interested in a concrete implementation of the schema defined in this implementation guide, and as such assumes a level of technical familiarity with software development concepts in general.

9.1.1 Technology

The HeD Schema Framework is built on the Microsoft .NET Framework, and is most conveniently built using Microsoft Visual Studio. However, there is no requirement that Visual Studio be used; any development environment capable of building .NET solutions may be used, such as Sharp Develop.

The framework targets the latest version of the .NET Framework, version 4.5, but it does not take advantage of any functionality that is exclusive to the 4.5 version, and should be backwards compatible with at least .NET 3.5.

The code is written entirely in C#, and makes extensive use of XML processing functionality. Specifically, the XDocument classes, as well as LINQ. It is worth noting, however, that the concepts used are based on traditional compiler and systems design principles and should translate straightforwardly to any technological environment.

9.1.2 Solution Structure

The overall solution is organized in the following assemblies:

Assembly	Description
HeDEngine	Provides the core infrastructure and functionality for dealing with HeD Schema, and registering the type handlers used to provide specific functionality such as semantic validation, translation, and evaluation.
vMR.Model	Contains the C# class definitions corresponding the vMR XML Schema, as well as the type resolver for the vMR model.
CREF.Translation	Contains the implementation of a translation module used to translate an HeD Schema Artifact to an Allscripts CREF representation.
HeDArtifactUtility	An executable that exposes the functionality built up in the HeDEngine and related assemblies.

© 2014 Health Level Seven International. All rights reserved

The overall solution is layered, just as the HeD Schema is, to ensure that there is no static dependency on the various models—such as vMR and CREF—that are used by the artifact utility. The purpose of the HeDEngine assembly is to provide the core infrastructure in such a way that it can be dynamically extended to support other data models and functionality.

9.1.3 Design Goals

The HeD Schema Framework is intended to serve as a platform for the implementation of language processing tools for the HeD Schema. The overall design is based on the traditional design of a compiler, with several important differences.

The following diagram depicts the data flow through a traditional compiler:

As shown, a string of the language is provided as input to the compiler. The first phase of processing is Lexical Analysis, which is responsible for converting the raw string of characters into discrete tokens.

Parsing then takes that stream of tokens and, using the grammar, or syntax rules, of the language, converts the tokens into an abstract syntax tree, a representation of the string that is syntactically valid, but not necessarily semantically valid.

Note that in the example, order of operations is being determined by the parser by arranging the resulting abstract syntax tree such that the multiply operation is invoked before the add operation.

The next phase is semantic analysis, which is responsible for ensuring that all the tokens referenced within the expression appear in valid contexts. Because the HeD Schema Framework is an abstract syntax tree representation of the logic involved, this is the stage at which processing

© 2014 Health Level Seven International. All rights reserved

of an HeD Artifact can begin. There is no need to perform lexical analysis or parsing, avoiding the difficulties associated with those tasks, and simplifying the overall language processing tasks.

The final phase is compilation, which is responsible for converting the semantically valid statement into an equivalent representation in the target language, usually machine code for a traditional compiler.

The goal of the HeD Schema Framework then is to provide an infrastructure for building translators and interpreters for the HeD Schema language, and to provide a complete semantic verifier that can be used to verify HeD Schema Artifact semantics.

9.2 Components

The following diagram depicts the major functional components provided by the HeD Engine assembly:

9.2.1 Maps

One of the core services provided by the HeD Schema Framework is the ability to define maps that allow functionality to be specified and loaded dynamically. This is the key feature that enables the decoupling of the core implementation from higher level functionality, and all the core pieces are defined in terms of maps in some way. In the abstract, a map is a dictionary that associates a string key with a specific class implementation, called a handler. There are several types of handlers introduced by the core infrastructure, and each map specifies a type of handler. Each handler type is associated with a specific interface, and each class in the map is required to implement that interface. The following table lists the types of handlers and their associated interfaces:

Handler Type	Interface	Description
Module Registration	IModuleRegistrar	Provides module registration functionality
Type Resolution	ITypeResolver	Resolves a type based on the XML qualified type name.
Verification	INodeVerifier	Verifies a specific type of node.

© 2014 Health Level Seven International. All rights reserved

Handler Type	Interface	Description
Evaluation		Not yet defined.
Translation	IArtifactTranslator	Translates a specific type of artifact.
Node Translation	INodeTranslator	Translates a specific type of node.
Model Translation	IModelTranslator	Translates a specific type of model.
Writing	IArtifactWriter	Writes a specific type of artifact.

For example, the ModuleMap, shown here:

```
<map xmlns="urn:hl7-org:v3/hed/engine"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:hl7-org:v3/hed/engine HandlerMap.xsd"
    handlerType="ModuleRegistration">
        <entry key="urn:hl7-org:v3/hed" value="HeD.Engine.Verification.BaseModuleRegistrar"/>
        <entry key="urn:hl7-org:v3/vmr" value="vMR.Model.vMRModuleRegistrar, vMR.Model"/>
</map>
```

defines the handlers responsible for loading modules into the base framework.

9.2.2 Model

The model component defines the model classes that are used to represent an HeD Schema Artifact within the HeD Schema Framework. These classes correspond roughly to the HeD Schema as defined in the various schema files in the HeDSchema folder. However, to simplify processing, there is not a one-to-one correspondence. For example, all expression schema types are represented simply by the ASTNode.

Note that the Framework could have used the direct representation of an artifact as XML (using the XDocument classes in the .NET Framework), but the ASTNode representation normalizes the expression classes defined in the HeD Schema, making abstract processing at later stages much simpler.

9.2.3 Reading

The Reading component defines the classes that read an HeD Schema Artifact and return the resulting equivalent representation in terms of the classes defined in the Model component. This component is a simple passthrough from the XML representation of the artifact to the Model classes.

9.2.4 Writing

The Writing component defines the handler interface for writing an artifact definition to an output stream. This component is used by the Translation component to write the in-memory representation of a translated artifact to the target output format, usually an XML representation.

9.3 Verification

The Verification component provides semantic verification of an HeD Schema Artifact. This requires several types of handler interfaces, as well as infrastructure functionality for tracking state during the verification process such as symbol tables and parameter definitions.

In order to verify the semantics of an HeD Artifact, the verification component must build a representation of the logic involved, and determine whether each expression in the artifact references valid types within the HeD type system, and valid operations defined by the language.

In general, this involves modeling the following concepts within the verification component:

Concept	Description

© 2014 Health Level Seven International. All rights reserved

Concept	Description
DataType	The type of a value within the HeD model. As defined by the model, data types can be Scalars, Intervals, Lists, and Objects. Each of these categories of types is represented by a different class within the HeD Model, all descended from the abstract base class DataType.
Operator	The definition of an operator, involving the name of the operator, as well as its signature (the types of its arguments), and its return type. Because HeD supports overloading, operators must be registered explicitly.
Parameter	A parameter represents the definition of an input parameter as part of an artifact definition.
Expression	An expression represents a named expression within the artifact definition that can be referenced by other expressions.
Symbol	A symbol represents a named symbol used within scopes such as filter and foreach operations to provide context. As in a traditional compiler, a symbol table is used to track a stack of symbols, allowing these types of operations to be nested.

The verification process then uses these concepts to construct a VerificationContext, and perform verification of each expression defined in the artifact. The following sections discuss the specific steps involved in verification.

9.3.1 Type Resolution

Type resolution is performed with classes that implement the ITypeResolver interface. This interface takes the string representation of a type, and returns the DataType representation of the type. The following classes model the HeD Schema type system:

Class	Description
ScalarType	Only the name of the type is represented.
IntervalType	The point type, as well as the begin and end properties are represented.
ListType	The element type is represented.
ObjectType	The properties of the object are represented.

9.3.2 Operator Resolution

Operator resolution is performed using the ResolveCall method of the VerificationContext. An operator within the HeD Schema is defined as the name of the operator, together with the types of all the arguments, and its result type. Because the HeD Schema supports operator overloading, the specific operator being invoked must be determined based on the resolved types of the arguments. Operator overloads are registered during the module registration process. For example, the following operator definitions specify the overloads for the Add operator:

```
new Operator("Add", new Signature(new[] { DataTypes.Integer, DataTypes.Integer }), DataTypes.Integer),
new Operator("Add", new Signature(new[] { DataTypes.Real, DataTypes.Real}), DataTypes.Real),
```

This mechanism allows module extensions to be introduced that register new overloads for existing operator definitions. For example, the vMR.Model assembly could register additional overloads of the Add operator for types that are introduced in the vMR.

© 2014 Health Level Seven International. All rights reserved

9.3.3 Node Verification

Node verification is performed with classes that implement the INodeVerifier interface. This interface takes the node representation of an expression and determines the type of the resulting expression. Any errors that are encountered through the process are thrown as exceptions.

For each node to be processed, the verification handler is looked up, using the mapping machinery provided by the core, based on the fully qualified XML name of the node type. For example, the following expression:

```
<expression xsi:type="Add">
    <operand xsi:type="IntegerLiteral" value="2"/>
    <operand xsi:type="IntegerLiteral" value="2"/>
</expression>
```

is validated by looking up the verification handler for the Add operation. This handler is defined in the VerificationHandlerMap:

```
<entry key="urn:hl7-org:v3/hed:Add" value="HeD.Engine.Verification.OperatorNodeVerifier, HeDEngine"/>
```

The entry specifies that verification for a node of this type is handled by the OperatorNodeVerifier class. The following shows the C# code for this class:

public class OperatorNodeVerifier : NodeVerifier

```
public class OperatorNodeVerifier : NodeVerifier
{
        protected virtual string GetOperatorName(ASTNode node)
        {
                return node.NodeType.GetLocalName();
        }

        protected override void InternalVerify(VerificationContext context, ASTNode node)
        {
                base.InternalVerify(context, node);

                var dataTypes = new List<DataType>();
                foreach (var child in node.Children)
                {
                        if (child.ResultType == null)
                        {
                                throw new InvalidOperationException(String.Format("Could not determine
type of '{0}' expression.", child.Name));
                        }

                        dataTypes.Add(child.ResultType);
                }

                var op = context.ResolveCall(GetOperatorName(node), new Signature(dataTypes));

                node.ResultType = op.ResultType;
        }
}
```

The OperatorNodeVerifier is a general class that provides node verification for any node that invokes an operator. The node first determines the data types of all arguments (all child nodes), by invoking the base InternalVerify. It then constructs the signature for the call by gathering the data types of each child node, and uses this signature to resolve the specific operator being invoked with the VerificationContext.ResolveCall method. If the signature does not match an existing overload, an exception is thrown. Otherwise, the ResultType of the node is set to the return type of the operator returned from the ResolveCall method.

Each type of expression within the HeD Schema is verified with a specific type of node verifier, as specified in the VerificationHandlerMap.

© 2014 Health Level Seven International. All rights reserved

9.3.4 Symbol Resolution

Several of the list processing operations defined within the HeD Schema, such as Filter, perform per-element processing. Access to each element being processed is enabled through a stack accessible via the verification context.

Semantic verification for these operations is performed by pushing a symbol on to the stack prior to verifying the per-element condition. If no name is specified for the symbol, the default name of Current is used.

Access within the per-element condition expression is allowed through the PropertyExpression, or the CurrentExpression. These expressions allow a scope name to be specified. If no name is specified, the default name of Current is used.

9.4 Translation

The translation component provides infrastructure for implementing translation from an HeD Artifact to a target artifact format. Translation functionality is built as extensions to the services provided by the core framework, so that any number of modules for translation to various formats can be provided.

9.4.1 Artifact Translation

The basic translation service is provided by implementing an IArtifactTranslator, and registering it in the TranslationHandlerMap.

Artifact translation proceeds by first verifying the artifact (so that all nodes have the ResultType specified), and then invoking the Translate method of the IArtifactTranslator interface of the appropriate translation handler.

9.4.2 Node Translation

The artifact translator first establishes a TranslationContext, and then, as with the verification steps, proceeds by instantiating the appropriate node translation handler for each node type, and invoking translate. For many node types, this involves a simple one-to-one translation. For example, the following HeD expression:

```
<operand xsi:type="GreaterOrEqual">
        <description>Patient age greater than or equal to 18</description>
        <operand xsi:type="ExpressionRef" name="PatientAge"/>
        <operand xsi:type="PhysicalQuantityLiteral" value="18" unit="a"/>
</operand>
```

Looks up the node translation handler based on the name of the expression type GreaterOrEqual:

```
<entry key="urn:hl7-org:v3/hed:GreaterOrEqual"
value="CREF.Translation.GreaterOrEqualTranslator,CREF.Translation"/>
```

9.4.3 Model Translation

Although the HeD Schema artifacts are written using the vMR as the patient data model, it is likely that different target artifact formats use an entirely different data model, and expressions within the artifact must therefore not only be translated syntactically, but semantically. The TransformModelType and TransformModelPath methods of the TranslationContext provide this functionality through the IModelTranslator interface.

Classes that implement this interface must provide the functionality to transform a model type or path reference within an HeD expression, targeting the data model for the artifact (vMR in most cases), to an equivalent model type or path reference as an expression of the target syntax.

The specific translation process is covered in more detail in the section on CREF translation.

© 2014 Health Level Seven International. All rights reserved

9.5 CREF Translation

This section describes the implementation of the CREF translation plug-in for the HeD Artifact Utility. The translator was built using two different ECA rules:

1. An ECA rule based on the NQF-0068 measure as the initial vehicle for testing and developing the translation.
2. An ECA rule based on the San Diego County Pertussis Reporting Requirement, used as a vehicle for further testing and development.

The NQF-0068 rule was selected because the HeD IG already included NQF-0068 as an example rule, and the EHR participant already had a test environment and test cases established for the rule due to its importance as an MU2 measure.

The San Diego County Pertussis Reporting Requirement was developed as an HeD ECA rule as part of the pilot efforts, and the translator was then used to demonstrate that the rule could be deployed to an existing CDS environment.

Note that the operator and expression mappings listed here do not represent a complete mapping of functionality from HeD to CREF, only that subset that was required to support translation of thse artifacts. Further work must still be done to complete the mapping and translation support.

Also note that the HeD expression language is a superset of CREF both in terms of expressive capability, as well as the types of data it is capable of representing. This means that in addition to the "Not Implemented" variety of support discussed above, there are places where the translation is simply not possible, and a "Not Supported" exception must be raised. For example, HeD supports intervals of any type, but CREF only supports DateTime intervals. As a result, interval translation is supported, but only if the interval type within HeD is over Timestamps.

9.6 Metadata

Metadata translation from the HeD format to Allscripts CREF was fairly straightforward, mostly due to the fact that the CREF format does not have as much information in the metadata section as a full HeD artifact, so the translation involved a fairly simple subset as follows:

HeD Metadata Element	CREF Equivalent
metadata\identifiers\identifier@root	@Name
metadata\title@value	@Description

For metadata that doesn't have a corresponding representation with CREF, the translation will just ignore the source metadata. This should be extended to produce warnings so that it's clear that information from the source artifact has been ignored in producing the translated output.

9.7 Syntax

Due to the shared heritage of HeD and CREF, the translation of the syntax of the logic involved in an artifact is almost a one-to-one mapping of the operators in each format. The following sections discuss the translation done for each conceptual group of operators.

9.7.1 External Data and Expressions

HeD Artifacts represent external data requirements, as well as reusable logic, using the ExpressionDef element in the HeD Schema. This element can appear in both the externalData element, and the expressions element of the HeD Artifact. Both appearances are translated to a

© 2014 Health Level Seven International. All rights reserved

NamedExpression element in CREF, and appear in the NamedExpressions section of the CREF rule.

Referencing an expression with an ExpressionRef element in HeD translates directly to an ExpressionReference in CREF.

9.7.2 Parameters

Parameters within an HeD artifact are declared using the ParameterDef element, and referenced with a ParameterRef. In CREF, only parameter reference is expressed, and is done using the ParameterExpression element. As such, the translator only deals with ParameterRef, and translates them directly to ParameterExpression elements.

Note that the ParameterExpression in CREF does support the notion of a default value for the parameter, which could be retrieved from the corresponding ParameterDef in the HeD Artifact. This is marked as a TODO in the translator code.

9.7.3 Scalar Values

Scalar values such as strings, integers, and dates, are represented within HeD using the various Literal expressions. In CREF, these values are represented using a ValueExpression. Because the range of values supported for these types maps directly between HeD and CREF, there is very little translation work that needs to be done beyond transforming the literals to an appropriate ValueExpression.

One exception to this is the representation of Timestamp value within HeD, as opposed to the DateTime representation expected by CREF. This is marked as a TODO in the translator code.

9.7.4 Complex Types

HeD allows for the representation of complex types such as PhysicalQuantity. However, the underlying model on which CREF operates, Allscripts Medical Object Model (MOM), does not have an equivalent data type. As such, physical quantity references must be implicitly converted by the translation in order to allow correct semantics to be communicated. For the artifact under consideration, one such physical quantity involved comparison of the age of a patient. To ensure correct comparison, the units of the physical quantity literal are examined and the value is converted to days according to the following table:

Physical Quantity Unit (date subset)	Conversion Factor
a (year)	value * 365.25
mo (month)	value * 30.4375
wk (week)	value * 7
d (day)	value

Note that this approach requires that references to physical quantities within the model also be converted. This issue is discussed later in the model translation section.

9.7.5 Casting and Conversion Operators

HeD Schema supports casting and conversion operator to deal with data models that support hierarchical definitions. Because CREF is an interpreted format, it does not have any direct equivalent for these operators:

Operator	CREF Equivalent
Is	Not Supported

© 2014 Health Level Seven International. All rights reserved

Operator	CREF Equivalent
As	Translated as a no-op. In other words, an As appearing in an HeD artifact is simply not output in the CREF result. Because CREF is interpreted, there is no need for a compile-time cast.
Convert	Not Supported

9.7.6 Null-Handling Operators

HeD Schema includes various operators for dealing with nulls, including a Null selector, various null testing and null conditioning operators. The following table lists these operators and their CREF equivalent:

Operator	CREF Equivalent
Null	CREF does not have a direct equivalent, however a Coalesce with no arguments is allowed and will return null, so this is used as the translation.
IsNull	CREF has no equivalent, this translation is not supported.
IfNull	Coalesce
Coalesce	Coalesce

9.7.7 Unary Operators

HeD Schema has separate representations for each supported operator, whereas CREF has a single expression type for all unary operators, one for binary operators, etc. and uses an Operator attribute to indicate which operation is being represented. The following table shows the translations for unary operators:

Operator	CREF Equivalent
Not	UnaryOperator[opNot]
Ceiling	UnaryOperator[opCeiling]
Floor	UnaryOperator[opFloor]
Negate	UnaryOperator[opNegate]
Round	UnaryOperator[opRound]

9.7.8 Binary Operators

The following table shows the translations for binary operators:

Operator	CREF Equivalent
Equal	BinaryOperator[opEqual]
NotEqual	BinaryOperator[opNotEqual]
Less	BinaryOperator[opLess]
Greater	BinaryOperator[opGreater]
LessOrEqual	BinaryOperator[opLessOrEqual]
GreaterOrEqual	BinaryOperator[opGreaterOrEqual]
Add	BinaryOperator[opAdd]
Subtract	BinaryOperator[opSubtract]
Multiply	BinaryOperator[opMultiply]
Divide	BinaryOperator[opDivide]
TruncatedDivide	BinaryOperator[opDiv]
Modulo	BinaryOperator[opMod]

© 2014 Health Level Seven International. All rights reserved

Operator	CREF Equivalent
Power	BinaryOperator[opPower]

9.7.9 Logical Operators

For the core logical operators, And and Or, in HeD, these are n-ary operators, but in CREF, they are Binary operators. However, CREF does have an n-ary flavor, called the LogicalConnective, which can represent And and Or.

In addition to basic logical operators, HeD supports a ternary conditional operator, as well as two flavors of case expressions. The CREF equivalents for these are Condition and Choice, respectively, but for the Choice specifically, the mapping is not direct. The following table illustrates these translations:

Operator	CREF Equivalent
And	LogicalConnective[opAnd]
Or	LogicalConnective[opOr]
Conditional	Condition
Case (w/ comparand)	Choice where each element is represented as Condition(BinaryOperator[opEqual](<comparand>, <whenNode>), <thenNode>, <elseNode>)
Case	Choice where each element is represented as Condition(<whenNode>, <thenNode>, <elseNode>)

9.7.10 Set/List Operators

Many of the set and list operations supported by HeD have direct equivalents within CREF. The following table shows the mappings that were used to support this translation effort:

Operator	CREF Equivalent
List(<elements>)	ListExpression(<elements>)
First(<source>)	UnaryOperator[opFirst](<source>)
Last(<source>)	UnaryOperator[opLast](<source>)
IsEmpty(<source>)	UnaryOperator[opNot](UnaryOperator[opExists](<source>))
IsNotEmpty(<source>)	UnaryOperator[opExists](<source>)
Contains(<source>, <element>) (List overload only)	UnaryOperator[opExists](FilterExpression(<source>, BinaryExpression[opEqual](PropertyExpression, <element>)))
Filter(<source>, <condition>)	FilterExpression(<source>, <condition>)
In(<element>, <collection>) (List overload only)	Same translation strategy as Contains

9.7.11 Date/Time Operators

Date and Time operators map fairly directly, as the following table shows:

Operator	CREF Equivalent
DateAdd	DateAdd
DateDiff	DateDiff (TODO)
DatePart	DatePart (TODO)

© 2014 Health Level Seven International. All rights reserved

Operator	CREF Equivalent
Today	Today
Now	Not Supported
Date	TODO

9.8 Model

In addition to translation of the syntax of the expressions involved in the artifact, the translator must transform aspects of those expressions that reference the model to the equivalent expression against the model underlying the CREF rule. For the HeD Artifact under consideration, that underlying model is the Virtual Medical Record (vMR), while the corresponding model on the CREF side is the Allscripts Medical Object Model (MOM).

As the following table shows, at a high level, the concepts map fairly directly:

vMR Type	MOM Equivalent
EvaluatedPerson	Patient(Detail)
AdverseEvent	Allergy where Status = Active
DeniedAdverseEvent	Allergy where Status = Denied
EncounterEvent	Encounter where Status = Complete
ObservationOrder	Result where Status = Complete
ObservationProposal	Result where Status = Ordered
ObservationResult	Result where Status = Active
UnconductedObservation	Result where Status = Denied
Problem	Problem where Status = Active
DeniedProblem	Problem where Status = Denied
ProcedureEvent	Procedure where Status = Complete
ProcedureOrder	Procedure where Status = Ordered
ProcedureProposal	Procedure where Status = Ordered
SubstanceAdministrationEvent	Medication where Status = Active
SubstanceAdministrationOrder	Medication where Status = Complete
SubstanceAdministrationProposal	Medication where Status = Ordered
UndeliveredSubstanceAdministration	Medication where Status = Denied

Note that because of the difference in the way that the vMR and the MOM represent the status of a specific clinical statement, the equivalent representation for most vMR references involves an additional status filter to be correctly represented in the MOM. For example, the following code snippet shows an HeD external data definition and its CREF equivalent:

```
<def name="AMI_Diagnosis">
        <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                dataType="vmr:Problem" codeProperty="problemCode" dateProperty="diagnosticEventTime.begin">
                <description>Diagnosis codes for acute myocardial infarction</description>
                <codes xsi:type="ValueSet" authority="National Committee for Quality Assurance"
                        id="2.16.840.1.113883.3.464.1003.104.12.1001" />
        </expression>
</def>


<ds:NamedExpression Name="AMI_Diagnosis">
 <ds:FilterExpression>
        <ds:RequestExpression Cardinality="Multiple" Type="Problem">
```


© 2014 Health Level Seven International. All rights reserved

```
            <ds:ValueSetExpression ValueSetID="2.16.840.1.113883.3.464.1003.104.12.1001" />
            <ds:RequestExpression.Codes />
          </ds:RequestExpression>
          <ds:BinaryExpression Operator="opEqual">
            <ds:PropertyExpression Path="Status" />
            <ds:ValueExpression Type="String" Value="Active" />
          </ds:BinaryExpression>
  </ds:FilterExpression>
</ds:NamedExpression>
```

9.8.1 Patient Age

One specific aspect of the NQF-0068 measure was the difference in the way that age is represented in the HeD version of the artifact. The vMR has a specific property of patients for representing the age, while the MOM has only the BirthDate. To capture this difference, a reference to the Age in vMR was translated as a CREF CalculateAge expression, referencing the BirthDate property of the Patient. In addition, because Age is represented as a Physical Quantity in the vMR, but as a Year value in CREF, the result of the CalculateAge expression was multiplied by 365.25 to convert the Year value to a day.

The following example shows the HeD age reference, followed by the CREF equivalent:

```
<def name="PatientAge">
          <expression xsi:type="Property" path="demographics.age">
                    <source xsi:type="ExpressionRef" name="Patient" />
          </expression>
</def>


<ds:NamedExpression Name="PatientAge">
  <ds:BinaryExpression Operator="opMultiply">
          <ds:CalculateAge>
            <ds:PropertyExpression Path="DateOfBirth">
                    <ds:ExpressionReference Name="Patient" />
            </ds:PropertyExpression>
          </ds:CalculateAge>
          <ds:ValueExpression Type="Decimal" Value="365.25" />
  </ds:BinaryExpression>
</ds:NamedExpression>
```

9.8.2 Negation Rationale

The NQF-0068 measure definition includes an exclusionary factor based on the existence of a documented reason for not prescribing an anti-thrombotic, represented in the original HeD Artifact as:

```
<def name="antithromboticNotPrescribedForDocumentedReason">
          <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                    dataType="vmr:ObservationResult" codeProperty="observationFocus"
                    dateProperty="observationEventTime.begin">
                    <description>Patient reason or other reason for not prescribing an antithrombotic</description>
                    <codes xsi:type="List">
                              <element xsi:type="CodeLiteral" code="G8697"
                                        codeSystem="2.16.840.1.113883.6.12" codeSystemName="CPT-4"
                                        displayName="" />
                    </codes>
          </expression>
</def>
```

However, the original CREF artifact interpreted this aspect of the measure as an Allergy:

```
<ds:NamedExpression Name="aspirinAdverseEvent">
  <ds:FilterExpression>
          <ds:RequestExpression Cardinality="Multiple" Type="Allergy">
            <ds:ValueSetExpression ValueSetID="2.16.840.1.113883.3.464.1003.196.12.1211" />
            <ds:RequestExpression.Codes />
          </ds:RequestExpression>
          <ds:BinaryExpression Operator="opEqual">
            <ds:PropertyExpression Path="Status" />
```


© 2014 Health Level Seven International. All rights reserved

```
            <ds:ValueExpression Type="String" Value="Active" />
          </ds:BinaryExpression>
  </ds:FilterExpression>
</ds:NamedExpression>
```

Because both these methods of exclusion would represent valid reasons for not prescribing an anti-thrombotic from a clinical perspective, the original HeD Artifact was modified to include the Allergy to Aspirin exclusionary criteria, and the Documented Negation Rationale was translated to an ObservationResult in the CREF artifact.

9.8.3 Procedures and Medications

Another difference between the original HeD Artifact and the original CREF artifact was in the way that Procedures and Medications were represented. Initially, the HeD Artifact only looked for completed Procedures, as specified in the NQF-0068 measure:

```
<def name="PCI_Procedures">
          <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                    dataType="vmr:ProcedureEvent" codeProperty="procedureCode"
                    dateProperty="procedureTime.begin">
                    <description>Procedure codes related to percutaneous coronary interventions</description>
                    <codes xsi:type="ValueSet" authority="National Committee for Quality Assurance"
                              id="2.16.840.1.113883.3.464.1003.104.12.1010" />
                    </expression>
</def>
```

However, since the original Allscripts expression of the rule was designed for use in a real-time setting, it included additional criteria. Specifically, the existence of an Encounter with a PCI code, as well as the existence of a proposal for a PCI procedure, were both considered satisfying criteria. As a result, the original HeD Artifact was modified to include those additional criteria for both procedures and medications.

9.8.4 Encounter Locations

For the San Diego County Pertussis Reporting Requirement, the HeD Artifact specifies criteria involving the patient's address, as well the address at which an encounter occurs. However, the Allscripts model does not have any mechanism to express these concepts within their patient data model. These criteria are therefore excluded from the translation by removing them from the HeD Artifact prior to executing the translation.

An implementation of this artifact using a system such as the Allscripts CDS that does not have facilities for expressing encounter or patient location, would need to use other mechanisms to determine the location criteria. For example, if the rule is deployed in a facility within San Diego, the criterion is effectively always true and can be ignored. Alternatively, the target system could be expanded to include location representation within the model, allowing translation of that aspect could be supported.

9.9 Value Sets

Although both HeD Schema and CREF allow artifacts to specify explicit lists of codes involved in a given criteria, they both also support the use of Value Sets to specify criteria. As a result, the HeD Artifact used the NQF value sets specified in the NQF-0068 eMeasure tooling. The original Allscripts rule for NQF-0068 used internal value sets, but rather than map those value sets, the Allscripts implementation was changed to use the NQF-0068 value sets. This resulted in a clean and straightforward mapping of the value sets involved.

In situations where this direct correspondence of value sets is not feasible, additional mapping will have to be undertaken, either within the translator, or by invoking calls to a terminology service.

© 2014 Health Level Seven International. All rights reserved

9.10 Guidance

Translating the guidance for an ECA rule involves converting the actions specified in the HeD artifact to the analogous structure in the target format. In the case of CREF, this structure is the Assertion. CREF supports 3 types of assertions:

Assertion Type	Description
MissingDataAssertion	Used to indicate that a specific piece of information is missing, such as a patient documentation item, or a medication.
OutOfRangeAssertion	Used to indicate that a specific value is out of an expected range, such as an A1C level.
CompositeAssertion	Used to group assertions. A composite assertion can indicate that all or any of the component assertions are recommended.

Within HeD, the following Action Types are available:

Action Type	Description
ActionGroup	Provides a container for actions, and allows various grouping behaviors such as All, Any, AtLeastOne, etc.
CreateAction	Defines an action that indicates a proposal to add patient information.
UpdateAction	Defines an action that indicates a proposal to update patient information.
RemoveAction	Defines an action that indicates a proposal to remove patient information.
FireEventAction	Indicates that an event should be fired.
MessageAction	Specifies a message should be returned.
CollectInformationAction	Defines an action to collect specific information.

9.10.1 Severity

HeD does not have an artifact-level severity indicator, which is required for CREF artifacts. Severity in HeD is specified as part of the proposals that are produced, rather than generically at the action level. This makes translation of the “severity” of the artifact quite difficult, because the severity could potentially be set differently based on input data for the artifact. As a result, the translator just sets the SeverityID for the output to MED, and issues a warning that an arbitrary value is being selected as part of the translation.

9.10.2 Constructed Guidance

The HeD Schema provides for the ability to construct proposals to be returned as guidance using object construction expressions provided as part of the HeD expression language. This results in very powerful and flexible functionality for HeD Artifacts. However, CREF has very limited facilities for that type of functionality. Specifically, it only supports token-style replacement within recommendation descriptions. As a result, the translation must place specific limitations on the extent of dynamic recommendations that are allowed in the input HeD Artifact. In this case, the recommendations returned include proposals to prescribe an anti-thrombotic, or to document that an anti-thrombotic has already been prescribed. These proposals are built with dynamic expressions within the action sentence of the HeD Artifact:

```
<simpleAction xsi:type="CreateAction">
```


© 2014 Health Level Seven International. All rights reserved

```
        <textEquivalent value="Prescribe aspirin or other antithrombotic"/>
        <actionSentence xsi:type="ObjectExpression"
                objectType="vmr:SubstanceAdministrationProposal">
                <description>Prescribe aspirin or other antithrombotic</description>
                <property name="substance.substanceCode">
                        <value xsi:type="CodeLiteral" code="2.16.840.1.113883.3.464.1003.196.12.1211"
                                codeSystem="National Committee for Quality Assurance"
                                displayName="Select a medication from this value set." />
                </property>
        </actionSentence>
</simpleAction>
```

To translate this guidance, the translator assumes a very specific pattern in the object expression, and uses that assumption to look for the code identifying the medication being prescribed. It then constructs an equivalent static representation for output in the resulting CREF:

```
<am:MissingDataAssertion Description="Prescribe aspirin or other antithrombotic"
        CodeSet="National Committee for Quality Assurance"
        Code="2.16.840.1.113883.3.464.1003.196.12.1211">
</am:MissingDataAssertion>
```

9.10.3 Dynamic Guidance

For the San Diego County Pertussis rule, the resulting guidance reports which criterion among several was actually the trigger for the report. Not only is this constructed guidance, as the NQF-0068 rule used, but is being constructed based on the patient data being evaluated. This means these is in general no way to translate the resulting guidance statically, it must be translated to an equivalent dynamic expression in the target artifact format.

With the CREF target format in particular, there are several potential options for handling this, listed in increasing order of impact on the relevant systems:

1. Change the guidance within the HeD Artifact to be static.
2. Manually translate the guidance portion of the HeD Artifact.
3. Devise a scheme to translate the dynamic references within the HeD Artifact to token replacement references within the guidance in the CREF Artifact.
4. Expand the CREF format to support dynamic construction of results the way that HeD does.

Given the time and resources available for the pilot project, the first option was selected in this case, but this has the disadvantage of losing the dynamically determined aspects of the resulting guidance. Further development in the future of the HeD to CREF translator, or the CREF format would result in an implementation that could fully support translation of dynamic guidance..

© 2014 Health Level Seven International. All rights reserved

10 APPENDIX E – EXAMPLES

10.1 FLACC Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?schematron-schema href="../main/schematron/knowledgeartifact.sch"?>
<?schematron-schema href="../main/schematron/documentationtemplates.sch"?>
<knowledgeDocument xmlns="urn:hl7-org:knowledgeartifact:r1"
        xmlns:vmr="urn:hl7-org:vmr:r2" xmlns:dt="urn:hl7-org:cdsdt:r2"
        xmlns:p1="http://www.w3.org/1999/xhtml" xmlns:xml="http://www.w3.org/XML/1998/namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:hl7-org:knowledgeartifact:r1
../schema/knowledgeartifact/knowledgedocument.xsd ">
        <!-- FLACC Pain Scale Documentation Template This example describes the
                FLACC pain scale as a documentation template. The example was chosen to illustrate
                the use of a documentation template to describe a typical practitioner administered
                patient evaluation instrument. -->
        <metadata>
                <identifiers>
                        <identifier root="FLACCExampleForHeD" version="2" />
                </identifiers>
                <artifactType value="Documentation Template" />

                <schemaIdentifier root="urn:hl7-org:knowledgeartifact:r1"
                        version="1" />

                <dataModels>
                        <modelReference>
                                <description value="Virtual Medical Record model" />
                                <referencedModel value="urn:hl7-org:vmr:r2" />
                        </modelReference>
                </dataModels>
                <title value="FLACC Pain Scale" />
                <relatedResources>
                        <relatedResource>
                                <relationship value="DerivedFrom" />
                                <resources>
                                        <resource>
                                                <title value="FLACC Scale" />
                                                <location

        value="http://painconsortium.nih.gov/pain_scales/FLACCScale.pdf" />
                                                <description value="NIH Pain Intnesity Instruments" />
                                        </resource>
                                </resources>
                        </relatedResource>
                </relatedResources>
                <supportingEvidence>
                        <evidence>
                                <resources>
                                        <resource>
                                                <citation
                                                        value="Merkel, SI, Voepel-Lewis, T., Shayevitz,
JR, &amp; Malviya, S. (1997).  The FLACC: a behavioral
```


© 2014 Health Level Seven International. All rights reserved

```
scale for scoring postoperative pain in young children.  Pediatric Nursing, 23(3): 293-297" />
                                        </resource>
                                    </resources>
                                </evidence>
                            </supportingEvidence>
                            <applicability>
                                <coverage>
                                    <focus value="PatientAgeGroup" />
                                    <description value="Population between the age of 2 months to 7 years"
/>
                                    <value code="D007223" codeSystem="2.16.840.1.113883.11.75"
                                        codeSystemName="MeSH - AgeGroupObservationValue">
                                        <dt:displayName value="Infant; 1 to 23 months" />
                                    </value>
                                </coverage>
                                <coverage>
                                    <focus value="PatientAgeGroup" />
                                    <description value="Population between the age of 2 months to 7 years"
/>
                                    <value code="D002675" codeSystem="2.16.840.1.113883.11.75"
                                        codeSystemName="MeSH - AgeGroupObservationValue">
                                        <dt:displayName value="child, preschool; 2 to 5 years" />
                                    </value>
                                </coverage>
                                <coverage>
                                    <focus value="PatientAgeGroup" />
                                    <description value="Population between the age of 2 months to 7 years"
/>
                                    <value code="D002648" codeSystem="2.16.840.1.113883.11.75"
                                        codeSystemName="MeSH - AgeGroupObservationValue">
                                        <dt:displayName value="child; 6 to 12 years" />
                                    </value>
                                </coverage>
                            </applicability>
                            <status value="Draft" />
                            <contributions>
                                <contribution>
                                    <contributor xsi:type="Person">
                                        <contacts>
                                            <contact value="mailto:aziz.boxwala@meliorix.com" />
                                        </contacts>
                                        <name use="C">
                                            <dt:part value="Aziz" type="GIV" />
                                            <dt:part value="Boxwala" type="FAM" />
                                        </name>
                                        <affiliation>
                                            <name value="Meliorix Inc." />
                                        </affiliation>
                                    </contributor>
                                    <role value="Author" />
                                </contribution>
                            </contributions>

                        </metadata>

                        <externalData>
                            <def name="Patient">
```


© 2014 Health Level Seven International. All rights reserved

```
                        <expression xsi:type="ClinicalRequest" cardinality="Single"
                                dataType="vmr:EvaluatedPerson" isInitial="true" />
                </def>
        </externalData>

        <expressions>
                <def name="PatientAgeInMonths">
                        <expression xsi:type="DateDiff">
                                <description>Number of years from patient's birth time to
today</description>
                                <startDate xsi:type="Property" path="birthTime">
                                        <source xsi:type="ExpressionRef" name="Patient" />
                                </startDate>
                                <endDate xsi:type="Today" />
                                <granularity xsi:type="Literal" valueType="DateGranularity"
                                        value="Month" />
                        </expression>
                </def>
        </expressions>

        <conditions>
                <condition>
                        <logic xsi:type="And">
                                <description>Patient is between 2 months and 7 years of
age</description>
                                <operand xsi:type="GreaterOrEqual">
                                        <description>Patient is more than 2 months old</description>
                                        <operand xsi:type="ExpressionRef"
name="PatientAgeInMonths"></operand>
                                        <!-- NOTE: A physical quantity with units of months is used here.
The
                                                intended semantics are that the implementation engine
will perform any conversions
                                                required to perform the comparison. -->
                                        <operand xsi:type="RealLiteral" value="2" />
                                </operand>
                                <operand xsi:type="LessOrEqual">
                                        <description>Patient is less than 7 years old (or 84
months)</description>
                                        <operand xsi:type="ExpressionRef"
name="PatientAgeInMonths"></operand>
                                        <operand xsi:type="RealLiteral" value="84" />
                                </operand>
                        </logic>
                        <conditionRole value="ApplicableScenario" />
                </condition>
        </conditions>

        <actionGroup>
                <title value="FLACC Scale" />
                <representedConcepts>
                        <concept code="38213-5" codeSystem="2.16.840.1.113883.6.1"
                                codeSystemName="LOINC">
                                <dt:displayName value="FLACC pain assessment panel:-:Pt:^Patient:-"
/>
                        </concept>
                </representedConcepts>
```


© 2014 Health Level Seven International. All rights reserved

```
            <subElements>
                <!-- Define the container to store responses to the documentation items. -->
                <simpleAction xsi:type="DeclareResponseAction" name="Responses" />

                <simpleAction xsi:type="CollectInformationAction">
                    <documentationConcept>
                        <displayText value="Face" />
                        <itemCodes>
                            <itemCode code="38216-8"
codeSystem="2.16.840.1.113883.6.1"
                                codeSystemName="LOINC" />
                        </itemCodes>
                        <responseDataType value="Integer" />
                        <responseCardinality value="Single" />
                        <responseRange xsi:type="ListConstraint"
                            strictSelection="true">
                            <constraintType value="List" />
                            <item>
                                <value xsi:type="IntegerLiteral" value="0" />
                                <displayText value="No particular expression or
smile" />
                            </item>
                            <item>
                                <value xsi:type="IntegerLiteral" value="1" />
                                <displayText
                                    value="Occasional grimace or frown,
withdrawn, disinterested" />
                            </item>
                            <item>
                                <value xsi:type="IntegerLiteral" value="2" />
                                <displayText value="Frequent to constant
quivering chin, clenched jaw" />
                            </item>
                        </responseRange>
                    </documentationConcept>
                    <responseBinding property="Face" />
                </simpleAction>
                <simpleAction xsi:type="CollectInformationAction">
                    <documentationConcept>
                        <displayText value="Legs" />
                        <itemCodes>
                            <itemCode code="38217-6"
codeSystem="2.16.840.1.113883.6.1"
                                codeSystemName="LOINC" />
                        </itemCodes>
                        <responseDataType value="Integer" />
                        <responseCardinality value="Single" />
                        <responseRange xsi:type="ListConstraint"
                            strictSelection="true">
                            <constraintType value="List" />
                            <item>
                                <value xsi:type="IntegerLiteral" value="0" />
                                <displayText value="Normal position or relaxed"
/>
                            </item>
                            <item>
                                <value xsi:type="IntegerLiteral" value="1" />
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <displayText value="Uneasy, restless, tense" />
                                    </item>
                                    <item>
                                        <value xsi:type="IntegerLiteral" value="2" />
                                        <displayText value="Kicking, or legs drawn up"
/>
                                    </item>
                                </responseRange>
                            </documentationConcept>
                            <responseBinding property="Legs" />
                        </simpleAction>

                        <simpleAction xsi:type="CollectInformationAction">
                            <documentationConcept>
                                <displayText value="Activity" />
                                <itemCodes>
                                    <itemCode code="38218-4"
codeSystem="2.16.840.1.113883.6.1"
                                        codeSystemName="LOINC" />
                                </itemCodes>
                                <responseDataType value="Integer" />
                                <responseCardinality value="Single" />
                                <responseRange xsi:type="ListConstraint"
                                    strictSelection="true">
                                    <constraintType value="List" />
                                    <item>
                                        <value xsi:type="IntegerLiteral" value="0" />
                                        <displayText value="Lying quietly, normal
position, moves easily" />
                                    </item>
                                    <item>
                                        <value xsi:type="IntegerLiteral" value="1" />
                                        <displayText value="Squirming, shifting back
and forth, tense" />
                                    </item>
                                    <item>
                                        <value xsi:type="IntegerLiteral" value="2" />
                                        <displayText value="Arched, rigid or jerking" />
                                    </item>
                                </responseRange>
                            </documentationConcept>
                            <responseBinding property="Activity" />
                        </simpleAction>

                        <simpleAction xsi:type="CollectInformationAction">
                            <documentationConcept>
                                <displayText value="Cry" />
                                <itemCodes>
                                    <itemCode code="38219-2"
codeSystem="2.16.840.1.113883.6.1"
                                        codeSystemName="LOINC" />
                                </itemCodes>
                                <responseDataType value="Integer" />
                                <responseCardinality value="Single" />
                                <responseRange xsi:type="ListConstraint"
                                    strictSelection="true">
                                    <constraintType value="List" />
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <item>
                                                <value xsi:type="IntegerLiteral" value="0" />
                                                <displayText value="No cry (awake or asleep)"
/>
                                        </item>
                                        <item>
                                                <value xsi:type="IntegerLiteral" value="1" />
                                                <displayText value="Moans or whimpers;
occasional complaint" />
                                        </item>
                                        <item>
                                                <value xsi:type="IntegerLiteral" value="2" />
                                                <displayText
                                                        value="Crying steadily, screams or sobs,
frequent complaints" />
                                        </item>
                                </responseRange>
                        </documentationConcept>
                        <responseBinding property="Cry" />
                </simpleAction>

                <simpleAction xsi:type="CollectInformationAction">
                        <documentationConcept>
                                <displayText value="Consolability" />
                                <itemCodes>
                                        <itemCode code="38220-0"
codeSystem="2.16.840.1.113883.6.1"
                                                codeSystemName="LOINC" />
                                </itemCodes>
                                <responseDataType value="Integer" />
                                <responseCardinality value="Single" />
                                <responseRange xsi:type="ListConstraint"
                                        strictSelection="true">
                                        <constraintType value="List" />
                                        <item>
                                                <value xsi:type="IntegerLiteral" value="0" />
                                                <displayText value="Content, relaxed" />
                                        </item>
                                        <item>
                                                <value xsi:type="IntegerLiteral" value="1" />
                                                <displayText
                                                        value="Reassured by occasional
touching, hugging or being talked to, distractible" />
                                        </item>
                                        <item>
                                                <value xsi:type="IntegerLiteral" value="2" />
                                                <displayText value="Difficult to console or
comfort" />
                                        </item>
                                </responseRange>
                        </documentationConcept>
                        <responseBinding property="Consolability" />
                </simpleAction>

                <simpleAction xsi:type="CollectInformationAction">
                        <documentationConcept>
                                <displayText value="Total Score" />
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <description
                                                value="Each of the five (5) categories is scored from 0-2,
which results in a total score between 0
        and 10." />
                                        <itemCodes>
                                                <itemCode code="38215-0"
codeSystem="2.16.840.1.113883.6.1"
                                                        codeSystemName="LOINC" />
                                        </itemCodes>
                                        <responseDataType value="Integer" />
                                        <responseCardinality value="Single" />
                                        <responseRange xsi:type="ExpressionConstraint">
                                                <constraintType value="Minimum" />
                                                <constraint xsi:type="IntegerLiteral" value="0" />
                                        </responseRange>
                                        <responseRange xsi:type="ExpressionConstraint">
                                                <constraintType value="Maximum" />
                                                <constraint xsi:type="IntegerLiteral" value="10" />
                                        </responseRange>
                                </documentationConcept>
                                <initialValue xsi:type="Sum">
                                        <source xsi:type="List">
                                                <element xsi:type="Property" path="Face">
                                                        <source xsi:type="ParameterRef"
name="Responses" />
                                                </element>
                                                <element xsi:type="Property" path="Legs">
                                                        <source xsi:type="ParameterRef"
name="Responses" />
                                                </element>
                                                <element xsi:type="Property" path="Activity">
                                                        <source xsi:type="ParameterRef"
name="Responses" />
                                                </element>
                                                <element xsi:type="Property" path="Cry">
                                                        <source xsi:type="ParameterRef"
name="Responses" />
                                                </element>
                                                <element xsi:type="Property" path="Consolability">
                                                        <source xsi:type="ParameterRef"
name="Responses" />
                                                </element>
                                        </source>
                                </initialValue>
                                <responseBinding property="TotalScore" />
                        </simpleAction>
                </subElements>
        </actionGroup>

</knowledgeDocument>
```

10.2 RespiratoryOrder Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?schematron-schema href="../main/schematron/knowledgeartifact.sch"?>
<?schematron-schema href="../main/schematron/ordersets.sch"?>
<knowledgeDocument xmlns="urn:hl7-org:knowledgeartifact:r1"
        xmlns:vmr="urn:hl7-org:vmr:r2" xmlns:dt="urn:hl7-org:cdsdt:r2"
```


© 2014 Health Level Seven International. All rights reserved

```
        xmlns:p1="http://www.w3.org/1999/xhtml" xmlns:xml="http://www.w3.org/XML/1998/namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:hl7-org:knowledgeartifact:r1
../schema/knowledgeartifact/knowledgedocument.xsd ">

        <!--
        Respiratory Order

        This example defines a typical respiratory order.

        This example was chosen to illustrate a modular order set that could be used alone or in
conjunction
        with another order set, and to illustrate the handling of Boolean selection logic within an order set.
        -->
        <metadata>
                <identifiers>
                        <identifier root="www.zynx.com/cds/orderset/RespiratoryProtocol"
extension="1234" version="2.7"/>
                </identifiers>
                <artifactType value="Order Set"/>
                <schemaIdentifier root="urn:hl7-org:knowledgeartifact:r1" version="1" />

                <dataModels>
                        <modelReference>
                                <description value="Virtual Medical Record model" />
                                <referencedModel value="urn:hl7-org:vmr:r2" />
                        </modelReference>
                </dataModels>
                <title value=" Respiratory Order Linkable Order Set"/>
                <description value="Order Set Illustrating Respiratory Orders. This order set is a linkable
order set. That is, it is intended to be modular and included as part of a larger order set."/>
                <documentation>
                        <description value="Explanation"/>
                        <content>
                                <xhtml:div xmlns:xhtml="http://www.w3.org/1999/xhtml">Order Set
Illustrating Respiratory Orders. This order set is a linkable order set. That is, it is intended to be modular
and included as part of a larger order set. It is used here to illustrate boolean relationships between
orderables in an order set.</xhtml:div>
                        </content>
                </documentation>
                <applicability>
                        <!-- Can we assume that anything subsumed under respiratory therapy is
applicable or does one need to enumerate each option? -->
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Respiratory Therapy Order - Parent Orderable"/>
                                <value code="53950000" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT">
                                        <dt:displayName value="Respiratory therapy (procedure)"/>
                                </value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Ventilator settings"/>
                                <value code="410210009" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Ventilator care management
(procedure)"/></value>
```


© 2014 Health Level Seven International. All rights reserved

```
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Pulse oximetry"/>
                                <value code="252465000" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Pulse oximetry (procedure)"/></value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Blood gas, arterial"/>
                                <value code="32564009" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Arterial specimen collection for laboratory
test (procedure)"/></value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Oxygen via nasal cannula"/>
                                <value code="371907003" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Oxygen administration by nasal cannula
(procedure)"/></value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Oxygen via nonrebreather face mask"/> <!-- No
exact match in SNOMED -->
                                <value code="371908008" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Oxygen administration by mask
(procedure)"/></value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Oxygen via simple face mask"/>  <!-- No exact
match in SNOMED -->
                                <value code="371908008" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Oxygen administration by mask
(procedure)"/></value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus"/>
                                <description value="Oxygen via Venturi mask"/>
                                <value code="429253002" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Oxygen administration by Venturi mask
(procedure)"/></value>
                        </coverage>
                </applicability>
                <status value="Draft"/>
                <eventHistory>
                        <!-- How do we handle change tracking -->
                        <artifactLifeCycleEvent>
                                <eventType value="Created"/>
                                <eventDateTime value="20121130"/>
                        </artifactLifeCycleEvent>
                </eventHistory>
                <contributions>
                        <contribution>
                                <contributor  xsi:type="Organization">
                                        <addresses>
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <address>
                                                <dt:part type="SAL" value="10880 Wilshire
Boulevard"/>
                                                <dt:part type="CTY" value="Los Angeles"/>
                                                <dt:part type="ZIP" value="90024"/>
                                                <dt:part type="STA" value="CA"/>
                                                <dt:part type="CNT" value="USA"/>
                                        </address>
                                </addresses>
                                <contacts>
                                        <contact value="310-825-3333" use="WP"/>
                                </contacts>
                                <name value="Zynx Health"/>
                        </contributor>
                        <role/>
                </contribution>
        </contributions>
        <publishers>
                <publisher xsi:type="Organization">
                        <addresses>
                                <address>
                                        <dt:part type="SAL" value="10880 Wilshire Boulevard"/>
                                        <dt:part type="CTY" value="Los Angeles"/>
                                        <dt:part type="ZIP" value="90024"/>
                                        <dt:part type="STA" value="CA"/>
                                        <dt:part type="CNT" value="USA"/>
                                </address>
                        </addresses>
                        <contacts>
                                <contact value="310-825-3333" use="WP"/>
                        </contacts>
                        <name value="Zynx Health"/>
                </publisher>
        </publishers>
    </metadata>
    <expressions/>
    <actionGroup> <!-- Respiratory Order -->
        <behaviors>
                <behavior xsi:type="GroupSelectionBehavior" value="AtMostOne"/>
        </behaviors>
        <representedConcepts>
                <concept code="53950000" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"><dt:displayName value="Respiratory therapy
(procedure)"/></concept>
        </representedConcepts>
        <subElements>
                <actionGroup> <!-- Ventilator group -->
                        <behaviors>
                                <behavior xsi:type="GroupSelectionBehavior" value="All"/>
                        </behaviors>
                        <subElements>
                                <simpleAction xsi:type="CreateAction">  <!-- Ventilator Settings -
->
                                        <textEquivalent value="Ventilator Settings"/>
                                        <actionSentence xsi:type="ObjectExpression"
objectType="vmr:ProcedureProposal">
                                                <property name="procedureCode">
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                <value xsi:type="CodeLiteral"
code="410210009" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT"
displayName="Ventilator care management (procedure)"/>
                                                        </property>
                                                </actionSentence>
                                        </simpleAction>
                                        <actionGroup>  <!-- Oxygenation Assessment group -->
                                                <behaviors>
                                                        <behavior xsi:type="GroupSelectionBehavior"
value="Any"/>
                                                </behaviors>
                                                <subElements>
                                                        <simpleAction xsi:type="CreateAction"> <!--
Pulse Oxymetry -->
                                                                <textEquivalent value="Pulse
Oxymetry"/>
                                                                <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                        <property
name="procedureCode">
                                                                                <value
xsi:type="CodeLiteral" code="252465000" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Pulse oximetry (procedure)"/>
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                        <simpleAction xsi:type="CreateAction">  <!--
Blood gas, arterial -->
                                                                <textEquivalent value="Blood gas,
arterial"/>
                                                                <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                        <property
name="procedureCode">
                                                                                <value
xsi:type="CodeLiteral" code="32564009" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Arterial specimen collection for laboratory test
(procedure)"/>
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                </subElements>
                                        </actionGroup>
                                </subElements>
                        </actionGroup>
                        <actionGroup> <!-- Supplemental oxygen group -->
                                <behaviors>
                                        <behavior xsi:type="GroupSelectionBehavior" value="All"/>
                                </behaviors>
                                <subElements>
                                        <actionGroup> <!-- Supplemental oxygen -->
                                                <behaviors>
                                                        <behavior xsi:type="GroupSelectionBehavior"
value="ExactlyOne"/>
                                                </behaviors>
                                                <subElements>
```


© 2014 Health Level Seven International. All rights reserved

```
                                                <simpleAction xsi:type="CreateAction">  <!--
        Oxygen via nasal canula -->
                                                        <textEquivalent value="Oxygen via nasal
canula"/>
                                                        <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                <property
name="procedureCode">
                                                                        <value
xsi:type="CodeLiteral" code="371907003" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Oxygen administration by nasal cannula (procedure)"/>
                                                                </property>
                                                        </actionSentence>
                                                </simpleAction>
                                                <simpleAction xsi:type="CreateAction"> <!--
        Oxygen via nonrebreather face mask. Note that this is not an exact match with SNOMED CT. -->
                                                        <textEquivalent value="Oxygen via
nonrebreather face mask"/>
                                                        <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                <property
name="procedureCode">
                                                                        <value
xsi:type="CodeLiteral" code="371908008" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Oxygen administration by mask (procedure)"/>
                                                                </property>
                                                        </actionSentence>
                                                </simpleAction>
                                                <simpleAction xsi:type="CreateAction">  <!--
        Oxygen via simple face mask. Note that this is not an exact match with SNOMED CT. -->
                                                        <textEquivalent value="Oxygen via
simple face mask"/>
                                                        <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                <property
name="procedureCode">
                                                                        <value
xsi:type="CodeLiteral" code="371908008" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Oxygen administration by mask (procedure)"/>
                                                                </property>
                                                        </actionSentence>
                                                </simpleAction>
                                                <simpleAction xsi:type="CreateAction">  <!--
        Oxygen via venturi mask -->
                                                        <textEquivalent value="Oxygen via
venturi mask"/>
                                                        <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                <property
name="procedureCode">
                                                                        <value
xsi:type="CodeLiteral" code="429253002" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Oxygen administration by Venturi mask (procedure)"/>
                                                                </property>
                                                        </actionSentence>
                                                </simpleAction>
                                        </subElements>
```


© 2014 Health Level Seven International. All rights reserved

```
                                        </actionGroup>
                                        <actionGroup> <!-- Oxygenation assessment group -->
                                                <behaviors>
                                                        <behavior xsi:type="GroupSelectionBehavior"
value="OneOrMore"/>
                                                </behaviors>
                                                <subElements>
                                                        <simpleAction xsi:type="CreateAction"> <!--
Pulse Oxymetry -->
                                                                <textEquivalent value="Pulse
Oxymetry"/>
                                                                <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                        <property
name="procedureCode">
                                                                                <value
xsi:type="CodeLiteral" code="252465000" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Pulse oximetry (procedure)"/>
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                        <simpleAction xsi:type="CreateAction">  <!--
Blood gas, arterial -->
                                                                <textEquivalent value="Blood gas,
arterial"/>
                                                                <actionSentence
xsi:type="ObjectExpression" objectType="vmr:ProcedureProposal">
                                                                        <property
name="procedureCode">
                                                                                <value
xsi:type="CodeLiteral" code="32564009" codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT" displayName="Arterial specimen collection for laboratory test
(procedure)"/>
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                </subElements>
                                        </actionGroup>
                                </subElements>
                        </actionGroup>
                </subElements>
        </actionGroup>
</knowledgeDocument>
```

10.3 DopamineComplexIVOrderWithComplexLiteral Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?schematron-schema href="../main/schematron/knowledgeartifact.sch"?>
<?schematron-schema href="../main/schematron/ordersets.sch"?>
<knowledgeDocument xmlns="urn:hl7-org:knowledgeartifact:r1"
        xmlns:vmr="urn:hl7-org:vmr:r2" xmlns:dt="urn:hl7-org:cdsdt:r2"
        xmlns:p1="http://www.w3.org/1999/xhtml" xmlns:xml="http://www.w3.org/XML/1998/namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:hl7-org:knowledgeartifact:r1
../schema/knowledgeartifact/knowledgedocument.xsd
        urn:hl7-org:vmr:r2 ../schema/vmr/vmr.xsd">
        <!-- Dopamine Complex IV Order This example describes a complex IV order.
                This example was chosen to illustrate a modular order set that could be used
```


© 2014 Health Level Seven International. All rights reserved

```
                alone or in conjunction with another order set, to illustrate a composite
                order that contains both a diluent (base solution) and an additive, and to
                illustrate the proposed vMR extension mechanism. -->
        <!-- Please note that all terminology bindings are for illustrative purposes
                only. At this time, the HeD community has not defined the terminologies and
                value sets that will be bound to specific clinical attribute types -->
        <metadata>
                <identifiers>
                        <identifier root="www.zynx.com/cds/orderset/ivsets"
                                extension="42364" version="0.9" />
                </identifiers>
                <artifactType value="Order Set" />
                <schemaIdentifier root="urn:hl7-org:knowledgeartifact:r1"
                        version="1" />
                <templateIds>
                        <!-- An IV Set would require a template ID since it makes use of the vMR
                                extension scheme -->
                        <templateId root="http://www.zynx.com/cds/template/"
                                extension="1237" />
                        <templateId
                                root="http://www.exampleURI.com/HeD/templates/attributeExtension" />
                        <templateId
root="http://www.exampleURI.com/HeD/templates/ComplexIVOrders" />
                </templateIds>
                <dataModels>
                        <modelReference>
                                <description value="Virtual Medical Record model" />
                                <referencedModel value="urn:hl7-org:vmr:r2" />
                        </modelReference>
                </dataModels>
                <title value="IV Set Order" />
                <description value="Order set illustrating IV Sets" />
                <documentation>
                        <description value="Explanation" />
                        <content>
                                <xhtml:div xmlns:xhtml="http://www.w3.org/1999/xhtml">IV Sets are
examples of complex
                                        medications commonly ordered in inpatient order sets.
                                </xhtml:div>
                        </content>
                </documentation>
                <status value="Draft" />
                <eventHistory>
                        <artifactLifeCycleEvent>
                                <eventType value="Created" />
                                <eventDateTime value="20121130" />
                        </artifactLifeCycleEvent>
                </eventHistory>
                <contributions>
                        <contribution>
                                <contributor xsi:type="Organization">
                                        <addresses>
                                                <address>
                                                        <dt:part type="SAL" value="10880 Wilshire
Boulevard" />
                                                        <dt:part type="CTY" value="Los Angeles" />
                                                        <dt:part type="ZIP" value="90024" />
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <dt:part type="STA" value="CA" />
                                        <dt:part type="CNT" value="USA" />
                                    </address>
                                </addresses>
                                <contacts>
                                    <contact value="310-825-3333" use="WP" />
                                </contacts>
                                <name value="Zynx Health" />
                            </contributor>
                            <role />
                        </contribution>
                    </contributions>
                    <publishers>
                        <publisher xsi:type="Organization">
                            <addresses>
                                <address>
                                    <dt:part type="SAL" value="10880 Wilshire Boulevard" />
                                    <dt:part type="CTY" value="Los Angeles" />
                                    <dt:part type="ZIP" value="90024" />
                                    <dt:part type="STA" value="CA" />
                                    <dt:part type="CNT" value="USA" />
                                </address>
                            </addresses>
                            <contacts>
                                <contact value="310-825-3333" use="WP" />
                            </contacts>
                            <name value="Zynx Health" />
                        </publisher>
                    </publishers>
                </metadata>
                <actionGroup>
                    <behaviors>
                        <behavior xsi:type="GroupSelectionBehavior" value="All" />
                    </behaviors>
                    <subElements>
                        <!-- Here we build the IV Set container order. All attributes here pertain
                            to the IV Set as a whole. -->
                        <simpleAction xsi:type="CreateAction">
                            <textEquivalent value="DOPamine drip 800mg/500mL D5W" />
                            <actionSentence xsi:type="ObjectRedefine">
                                <description>
                                                DOPamine drip 800mg/500mL D5W
                                                Starting Dose: 2 mcg/kg/min,
                                                Titrate Increment: 0.5 mcg/kg/min every
5 minutes,
                                                Keep MAP Greater Than: 60,
                                                Hold if HR Greater than: 120,
                                                Priority: Routine,
                                                Start Time Offset: now,
                                                Special Inst: maximum dose of
20mcg/kg/min, titrate,
                                                Comments: Ordered as: DOPamine drip
800mg/500mL D5W,
                                                Diluent: D5W Titratable Base 500 mL,
IV,
                                                Additive: DOPamine (for infusion) 800
mg
```


© 2014 Health Level Seven International. All rights reserved

```
                                                    </description>
                                        <source xsi:type="ComplexLiteral">
                                                <!-- AB: Changed type from SubstanceAdminProposal --
>
                                                <value xsi:type="vmr:CompositeSubstanceProposal">
                                                    <vmr:id root="12345" /> <!-- TODO: id should not
be required -->
                                                    <!-- Dose Restriction: "maximum dose of
20mcg/kg/min, titrate" -->

                                                    <!-- AB:added -->
                                                    <!-- Starting Dose: 2 mcg/kg/min -->
                                                    <vmr:attribute>
                                                        <vmr:name value="Starting Dose"/>
                                                        <vmr:value xsi:type="vmr:Value">
                                                            <vmr:value xsi:type="dt:PQ"
value="2" unit="mcg/kg/min"></vmr:value>
                                                        </vmr:value>
                                                    </vmr:attribute>

                                                    <!-- AB:added -->
                                                    <!-- Titrate Increment: 0.5 mcg/kg/min -->
                                                    <vmr:attribute>
                                                        <vmr:name value="Titrate Increment"/>
                                                        <vmr:value xsi:type="vmr:Value">
                                                            <vmr:value xsi:type="dt:PQ"
value="0.5" unit="mcg/kg/min"></vmr:value>
                                                        </vmr:value>
                                                    </vmr:attribute>

                                                    <!-- AB: Added -->
                                                    <!-- Nursing Instruction: "Hold if HR Greater
than: 120" -->
                                                    <vmr:comment>
                                                        <vmr:type><dt:displayName
value="Instruction"/></vmr:type>
                                                        <vmr:content value="Hold if HR Greater
than: 120"/>
                                                    </vmr:comment>

                                                    <!-- AB: Added -->
                                                    <!-- Nursing Instruction: "Keep MAP Greater
Than: 60" -->
                                                    <vmr:comment>
                                                        <vmr:type><dt:displayName
value="Instruction"/></vmr:type>
                                                        <vmr:content value="Keep MAP Greater
Than: 60"/>
                                                    </vmr:comment>

                                                    <!-- AB: cannot specify this with atttribute since
frequency is not
                                                        a sub-type of any -->
                                                    <!-- Titrate Frequency: q5min -->
                                                    <vmr:relatedClinicalStatement>
                                                        <vmr:targetRole code="???"
codeSystem="???"
```


© 2014 Health Level Seven International. All rights reserved

```
        codeSystemName="???"><dt:displayName value="extendClassWithAttribute"/></vmr:targetRole>
                                                                <vmr:clinicalStatement

        xsi:type="vmr:ObservationResult">
                                                                        <vmr:id root="12345" />
                                                                        <vmr:observationFocus
code="???"
                                                                                codeSystem="???"
codeSystemName="???"><dt:displayName value="Titrate Frequency"/>
                                                                        </vmr:observationFocus>
                                                                        <!-- AB: In the new VMR
Frequency is its own type. So this is incorrect. -->
                                                                        <vmr:observationValue
xsi:type="vmr:Value">
                                                                                <vmr:value
xsi:type="dt:CD"
                                                                                        code="???"
codeSystem="???" codeSystemName="???">

        <dt:displayName value="q5min"/>
                                                                                </vmr:value>
                                                                        </vmr:observationValue>
                                                                <!-- Specify the drug additive for this Complex IV
-->
                                                                <vmr:substance>
                                                                        <vmr:id root="12345" />
                                                                        <vmr:substanceCode code="1160755"

        codeSystem="2.16.840.1.113883.6.88" codeSystemName="RxNorm">
                                                                        <dt:displayName
value="Dopamine Injectable Product"/>
                                                                        </vmr:substanceCode>
                                                                        <vmr:strength>
                                                                        <dt:numerator xsi:type="dt:PQ"
value="800" unit="mg" />
                                                                        <dt:denominator
xsi:type="dt:PQ" value="500" unit="ml" />
                                                                        </vmr:strength>
                                                                </vmr:substance>

                                                                <vmr:dose>
                                                                        <vmr:doseRestriction>
                                                                        <vmr:maxDoseForInterval
value="20" unit="mcg/kg" /> <!-- <originalText>maximum dose of 20mcg/kg/min, titrate</originalText> -->
                                                                        <vmr:timeInterval value="1"
unit="min" />
                                                                        </vmr:doseRestriction>
                                                                </vmr:dose>

                                                                <!-- AB: Added -->
                                                                <vmr:urgency><dt:displayName
value="routine"/></vmr:urgency>
```


© 2014 Health Level Seven International. All rights reserved

```
                                                <vmr:constituent>
                                                        <vmr:constituentType code="Diluent"/>
                                                        <vmr:substance>
                                                                <vmr:id root="12345" />
                                                                <vmr:substanceCode
code="400420008"

        codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT">
                                                                        <dt:displayName
value="Dextrose 5g/100mL (5%) injection solution 500mL vial"/>
                                                                <vmr:substanceCode>
                                                        </vmr:substance>
                                                        <vmr:dose></vmr:dose>
                                                </vmr:constituent>

                                                <!-- AB: Added -->
                                                <vmr:constituent>
                                                        <vmr:id root="12345" />
                                                        <vmr:constituentType code="Additive"/>
                                                        <vmr:substance>
                                                                <vmr:id root="12345" />
                                                                <vmr:substanceCode
code="1160755"

        codeSystem="2.16.840.1.113883.6.88" codeSystemName="RxNorm">
                                                                        <dt:displayName
value="Dopamine Injectable Product"/>
                                                                <vmr:substanceCode>
                                                        </vmr:substance>
                                                        <vmr:dose>
                                                                <vmr:doseQuantity>
                                                                        <dt:low value="800"
unit="mg" />
                                                                        <dt:high value="800"
unit="mg" /> <!-- TODO: highValue and highUnit are required? -->
                                                                <vmr:doseQuantity>
                                                        <vmr:dose>
                                                </vmr:constituent>
                                        </value>
                                </source>
                                <!-- Start Time Offset: Now -->
                                <property name="proposedAdministrationTimeInterval"> <!--
IVL_TS -->
                                        <value xsi:type="Interval">
                                                <begin xsi:type="Now" />
                                        </value>
                                </property>
                        </actionSentence>
                </simpleAction>
        </subElements>
    </actionGroup>
</knowledgeDocument>
```

10.4 HeartFailureAdmissionToMedSurgOrderSet Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?schematron-schema href="../main/schematron/knowledgeartifact.sch"?>
<?schematron-schema href="../main/schematron/ordersets.sch"?>
```


© 2014 Health Level Seven International. All rights reserved

```
<!-- Rationale: -->
<knowledgeDocument xmlns="urn:hl7-org:knowledgeartifact:r1"
        xmlns:vmr="urn:hl7-org:vmr:r2" xmlns:dt="urn:hl7-org:cdsdt:r2"
        xmlns:p1="http://www.w3.org/1999/xhtml" xmlns:xml="http://www.w3.org/XML/1998/namespace"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:hl7-org:knowledgeartifact:r1
../schema/knowledgeartifact/knowledgedocument.xsd
         urn:hl7-org:vmr:r2 ../schema/vmr/vmr.xsd">
        <!-- Heart Failure Admission to Med/Surg Unit This is a partial order set
                for Heart Failure, Admission to the Med/Surg unit of a hospital. This example
                was chosen to illustrate a routine order set and how to represent sections,
                reminders, orders, order details, performance measures, evidence links, selection
                (and other) types of behaviors, and links to modular order sets. -->
        <!-- Note that all coding systems and codes in this example are for illustrative
                purposes only. The Health eDecisions Terminologies and Value Sets sub-working
                group is working to finalize a set of terminologies and value sets for all
                the coded values in a Health eDecisions artifact. Once that set of terminologies
                and value sets is finalized, this example will be updated to include the
                correct references. For example, LOINC is used in this example as the terminology
                for laboratory orders, but this is subject to change, depending on the results
                of the sub-working group. -->
        <metadata>
                <!-- This section contains all the metadata for the artifact which can
                        be used to support searches -->
                <identifiers>
                        <identifier
                                root="www.zynx.com/cds/orderset/HeartFailureAdmissionToMedSurg"
                                extension="1234" version="3.0" />
                </identifiers>
                <artifactType value="Order Set" />
                <schemaIdentifier root="urn:hl7-org:knowledgeartifact:r1"
                        version="1" />
                <templateIds>
                        <templateId root="http://www.zynx.com/cds/template/"
                                extension="1234" version="3.0" />
                </templateIds>
                <dataModels>
                        <modelReference>
                                <description value="Virtual Medical Record model" />
                                <referencedModel value="urn:hl7-org:vmr:r2" />
                        </modelReference>
                </dataModels>

                <title value="Heart Failure Admission to Med/Surg" />
                <description
                        value="The Heart Failure module addresses the medical inpatient management
of adult patients with new-onset or acute exacerbations of heart failure. The emphasis of this module is on
medical management. Surgical and other nonmedical interventions are not covered in depth. This module
addresses acute cardiogenic pulmonary edema due to heart failure with reduced left ventricular ejection
fraction. This module does not fully address management of acute myocardial infarction or unstable
angina." />
                <documentation>
                        <title value="Zynx Heart Failure Module" />
                        <location value="https://www.zynx.com/Reference/Content.aspx?ItemID=216945"
/>
                        <description
```


© 2014 Health Level Seven International. All rights reserved

```
                                value="Addresses the medical inpatient management of adult patients
with new-onset or acute exacerbations of heart failure" />
                </documentation>
                <applicability>
                        <!-- These coverage codes are to help users who are searching for an applicable
                                Order Set -->
                        <coverage>
                                <focus value="ClinicalFocus" />
                                <description value="Heart Failure - SNOMED CT" />
                                <value codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED CT"
                                        code="84114007">
                                        <dt:displayName value="Heart Failure (disorder)" />
                                </value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalFocus" />
                                <description value="Heart Failure - ICD-9" />
                                <value codeSystem="2.16.840.1.113883.3.7.1.6.1"
                                        codeSystemName="ICD-9" code="428.9">
                                        <dt:displayName value="Heart failure, unspecified" />
                                </value>
                        </coverage>
                        <coverage>
                                <focus value="PatientAgeGroup" />
                                <description value="Adult" />
                                <value code="133936004" codeSystem="2.16.840.1.113883.6.96"
                                        codeSystemName="SNOMED-CT">
                                        <dt:displayName value="Adult" />
                                </value>
                        </coverage>
                        <coverage>
                                <focus value="ClinicalVenue" />
                                <description value="Inpatient" />
                                <value codeSystem="2.16.840.1.113883.12.4"
codeSystemName="Patient class (HL7)"
                                        code="I">
                                        <dt:displayName value="Inpatient" />
                                </value>
                        </coverage>
                </applicability>
                <status value="Active" />
                <eventHistory>
                        <artifactLifeCycleEvent>
                                <eventType value="Created" />
                                <eventDateTime value="20110125" />
                        </artifactLifeCycleEvent>
                        <artifactLifeCycleEvent>
                                <eventType value="Pre-published" />
                                <eventDateTime value="20110911" />
                        </artifactLifeCycleEvent>
                        <artifactLifeCycleEvent>
                                <eventType value="Published" />
                                <eventDateTime value="20120125" />
                        </artifactLifeCycleEvent>
                </eventHistory>
                <publishers>
```


© 2014 Health Level Seven International. All rights reserved

```
                        <publisher xsi:type="Organization">
                                <addresses>
                                        <address>
                                                <dt:part type="SAL" value="10880 Wilshire Boulevard" />
                                                <dt:part type="CTY" value="Los Angeles" />
                                                <dt:part type="ZIP" value="90024" />
                                                <dt:part type="STA" value="CA" />
                                                <dt:part type="CNT" value="USA" />
                                        </address>
                                </addresses>
                                <contacts>
                                        <contact value="310-825-3333" use="WP" />
                                </contacts>
                                <name value="Zynx Health" />
                        </publisher>
                </publishers>
        </metadata>
        <externalData>
                <def name="Patient">
                        <expression xsi:type="ClinicalRequest" cardinality="Single"
                                dataType="vmr:EvaluatedPerson" isInitial="true" />
                </def>
                <def name="AdverseReactionToACEInhibitors">
                        <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                                dataType="vmr:AdverseEvent" isInitial="true">
                                <codes xsi:type="List">
                                        <element xsi:type="CodeLiteral" code="293500009"
                                                codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"
                                                displayName="Angiotensin-converting-enzyme inhibitor
adverse reaction (disorder)" />
                                        <element xsi:type="CodeLiteral" code="295036000"
                                                codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"
                                                displayName="Angiotensin-converting-enzyme inhibitor
allergy (disorder)" />
                                        <element xsi:type="CodeLiteral" code="407579007"
                                                codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"
                                                displayName="History of - angiotensin II receptor
antagonist allergy (situation)" />
                                        <element xsi:type="CodeLiteral" code="407590002"
                                                codeSystem="2.16.840.1.113883.6.96"
codeSystemName="SNOMED-CT"
                                                displayName="Angiotensin II receptor antagonist
adverse reaction " />
                                </codes>
                        </expression>
                </def>
                <def name="AdministeredACEInhibitors">
                        <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                                dataType="vmr:SubstanceAdministrationEvent"
codeProperty="substanceAdministrationGeneralPurpose"
                                dateProperty="administrationTimeInterval.low" useValueSets="true">
                                <description>ACE inhibitor administered to patient</description>
                                <codes xsi:type="List">
                                        <element xsi:type="CodeLiteral" code="N0000000181"
```


© 2014 Health Level Seven International. All rights reserved

```
                                        codeSystem="2.16.840.1.113883.3.26.1.5"
codeSystemName="NDF-RT"
                                        displayName="Angiotensin-Converting Enzyme
Inhibitors" />
                        </codes>
                    </expression>
                </def>
                <def name="PrescribedACEInhibitors">
                    <expression xsi:type="ClinicalRequest" cardinality="Multiple"
                        dataType="vmr:SubstanceAdministrationProposal"
codeProperty="substanceAdministrationGeneralPurpose"
                        dateProperty="proposedAdministrationTimeInterval.low"
useValueSets="true">
                        <description>Patient prescribed ACE inhibitor medication</description>
                        <codes xsi:type="List">
                            <element xsi:type="CodeLiteral" code="N0000000181"
                                        codeSystem="2.16.840.1.113883.3.26.1.5"
codeSystemName="NDF-RT"
                                        displayName="Angiotensin-Converting Enzyme
Inhibitors" />
                        </codes>
                    </expression>
                </def>
            </externalData>
            <expressions>
                <def name="PatientAgeInYears">
                    <expression xsi:type="DateDiff">
                        <description>Number of years from patient's birth time to
today</description>
                        <startDate xsi:type="Property" path="birthTime">
                            <source xsi:type="ExpressionRef" name="Patient" />
                        </startDate>
                        <endDate xsi:type="Today" />
                        <granularity xsi:type="Literal" valueType="DateGranularity"
                            value="Year" />
                    </expression>
                </def>
            </expressions>
            <actionGroup>
                <subElements>
                    <actionGroup>
                        <!-- General -->
                        <title value="General" />
                        <subElements>
                            <!-- Note that both of these "reminder" elements are modelled as
MessageActions,
                                        but we need to revisit the semantics of MessageAction
so we can distinguish
                                        message actions that might be ordered by a provider
(e.g., "send email to
                                        patient about xxx") vs. messages that are simply
included in-line with a
                                        list of orders in an Order Set. Note that group selection
behaviours related
                                        to selection should not apply to the latter type of
message, but should apply
                                        to the former type. -->
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <simpleAction xsi:type="CreateAction">
                                                <supportingEvidence>
                                                        <evidence>
                                                                <resources>
                                                                        <!-- shows both a content-
provider evidence link (Zynx) as well
                                                                                as a link to a third-party
source -->
                                                                        <resource>
                                                                                <title value="Zynx
Evidence" />
                                                                                <location

        value="https://www.zynx.com/Reference/Content.aspx?ItemID=216984&amp;ver=1" />
                                                                        </resource>
                                                                        <resource>
                                                                                <title value="European
Journal of Heart Failure 2009" />
                                                                                <location

        value="http://eurjhf.oxfordjournals.org/content/11/12/1208.full.pdf" />
                                                                        </resource>
                                                                </resources>
                                                        </evidence>
                                                </supportingEvidence>
                                                <actionSentence xsi:type="ObjectExpression"
                                                        objectType="vmr:CommunicationProposal">
                                                        <property name="message">
                                                                <value xsi:type="ComplexLiteral">
                                                                        <value xsi:type="dt:ED"
value="ADHERE Risk Model" />
                                                                </value>
                                                        </property>
                                                </actionSentence>
                                        </simpleAction>
                                        <simpleAction xsi:type="CreateAction">
                                                <supportingEvidence>
                                                        <evidence>
                                                                <resources>
                                                                        <resource>
                                                                                <title value="Zynx
Evidence" />
                                                                                <location

        value="https://www.zynx.com/Reference/Content.aspx?ItemID=216985&amp;ver=1" />
                                                                        </resource>
                                                                </resources>
                                                        </evidence>
                                                </supportingEvidence>
                                                <actionSentence xsi:type="ObjectExpression"
                                                        objectType="vmr:CommunicationProposal">
                                                        <property name="message">
                                                                <value xsi:type="ComplexLiteral">
                                                                        <value xsi:type="dt:ED"
value="Heart Failure Survival Score" />
                                                                </value>
                                                        </property>
```


© 2014 Health Level Seven International. All rights reserved

```
                                        </actionSentence>
                                    </simpleAction>
                                </subElements>
                            </actionGroup>
                            <actionGroup>
                                <!-- Activity -->
                                <behaviors>
                                    <!-- Indicate that the physician must choose exactly one of the
items
                                        in this group as they are mutually exclusive. -->
                                    <behavior xsi:type="GroupSelectionBehavior"
value="ExactlyOne" />
                                </behaviors>
                                <title value="Activity" />
                                <subElements>
                                    <!-- Create some proposed procedures to be presented to the
physician
                                        at CPOE time -->
                                    <simpleAction xsi:type="CreateAction">
                                        <!-- Create a proposed procedure to be presented to the
physician at
                                            CPOE time -->
                                        <textEquivalent value="Ambulate" />
                                        <actionSentence xsi:type="ObjectExpression"
                                            objectType="vmr:ProcedureProposal">
                                            <property name="procedureCode">
                                                <value xsi:type="CodeLiteral"
codeSystem="2.16.840.1.113883.6.96"
                                                    code="62013009"
codeSystemName="SNOMED-CT" displayName="Ambulating Patient" />
                                            </property>
                                        </actionSentence>
                                    </simpleAction>
                                    <simpleAction xsi:type="CreateAction">
                                        <textEquivalent value="Bed rest" />
                                        <actionSentence xsi:type="ObjectExpression"
                                            objectType="vmr:ProcedureProposal">
                                            <property name="procedureCode">
                                                <value xsi:type="CodeLiteral"
codeSystem="2.16.840.1.113883.6.96"
                                                    code="183074009"
codeSystemName="SNOMED-CT" displayName="Recommendation to rest in bed" />
                                                <!-- Terminology note: this is supposed
to be an order, not a recommendation,
                                                    so SNOMED-CT term is not a
good fit for this -->
                                            </property>
                                        </actionSentence>
                                    </simpleAction>
                                </subElements>
                            </actionGroup>
                            <actionGroup>
                                <!-- Nursing Orders -->
                                <title value="Nursing Orders" />
                                <subElements>
                                    <actionGroup>
                                        <behaviors>
```


© 2014 Health Level Seven International. All rights reserved

```
                                                <!-- Indicate that the physician may choose any
items in the group.
                                                        "Any" includes choosing none. -->
                                                <behavior xsi:type="GroupSelectionBehavior"
value="Any" />
                                        </behaviors>
                                        <title value="Assessments" />
                                        <subElements>
                                                <simpleAction xsi:type="CreateAction">
                                                        <!-- Cardiac monitor -->
                                                        <behaviors>
                                                                <behavior
xsi:type="PrecheckBehavior" value="Yes" />
                                                        </behaviors>
                                                        <textEquivalent value="Cardiac Monitor"
/>
                                                        <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:ProcedureProposal">
                                                                <property
name="procedureCode">
                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.96"

        code="88140007" codeSystemName="SNOMED-CT" displayName="Cardiac monitor
surveillance" />
                                                                </property>
                                                        </actionSentence>
                                                </simpleAction>
                                                <simpleAction xsi:type="CreateAction">
                                                        <!-- Measure blood pressure, orthostatic
-->
                                                        <textEquivalent value="Measure blood
pressure, orthostatic" />
                                                        <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:ProcedureProposal">
                                                                <property
name="procedureCode">
                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.96"

        code="425058005" codeSystemName="SNOMED-CT"

        displayName="Taking orthostatic vital signs" />
                                                                        <!-- Terminology
comment: SNOMED Term is an indirect match for
                                                                                the desired
order -->
                                                                </property>
                                                        </actionSentence>
                                                </simpleAction>
                                        </subElements>
                                </actionGroup>
                                <actionGroup>
```


© 2014 Health Level Seven International. All rights reserved

```
                                                <behaviors>
                                                        <behavior xsi:type="GroupSelectionBehavior"
value="Any" />
                                                </behaviors>
                                                <title value="Interventions" />
                                                <subElements>
                                                        <simpleAction xsi:type="CreateAction">
                                                                <textEquivalent value="Elevate head of
bed" />
                                                                <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:ProcedureProposal">
                                                                        <property
name="procedureCode">
                                                                                <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.96"

        code="423171007" codeSystemName="SNOMED-CT" displayName="Elevation of head of bed"
/>
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                        <simpleAction xsi:type="CreateAction">
                                                                <textEquivalent value="Urinary catheter
initiation/management" />
                                                                <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:ProcedureProposal">
                                                                        <property
name="procedureCode">
                                                                                <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.96"

        code="429723008" codeSystemName="SNOMED-CT"

        displayName="Procedure involving urinary catheter" />
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                </subElements>
                                        </actionGroup>
                                </subElements>
                        </actionGroup>
                        <actionGroup>
                                <!-- Medications -->
                                <behaviors>
                                        <!-- Indicate that this group should be visually separated from
other
                                                groups in the order set, perhaps as indented section with
the group title
                                                as the title, etc. -->
                                        <behavior xsi:type="GroupOrganizationBehavior"
value="VisualGroup" />
                                </behaviors>
                                <title value="Medications" />
```


© 2014 Health Level Seven International. All rights reserved

```
                                <subElements>
                                        <actionGroup>
                                                <subElements>
                                                        <simpleAction xsi:type="CreateAction">
                                                                <!-- if patient meets certain criteria, then
remind the physician
                                                                        to consider ordering an ACE
inhibitor. -->
                                                                <conditions>
                                                                        <condition>
                                                                                <logic xsi:type="And">
                                                                                        <operand
xsi:type="GreaterOrEqual">

        <operand xsi:type="ExpressionRef" name="PatientAgeInYears" />

        <operand xsi:type="RealLiteral" value="18" />
                                                                                        </operand>
                                                                                        <operand
xsi:type="IsEmpty">

        <operand xsi:type="ExpressionRef" name="AdverseReactionToACEInhibitors" />
                                                                                        </operand>
                                                                                </logic>
                                                                                <conditionRole
value="ApplicableScenario" />
                                                                        </condition>
                                                                </conditions>
                                                                <textEquivalent
                                                                        value="If the patient is over 18
years old and not allergic to an ACE inhibitor and not receiving an ACE inhibitor and LVEF is less than
40%, then consider prescribing an ACE inhibitor." />
                                                                <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:CcommunicationProposal">
                                                                        <property name="message">
                                                                                <value
xsi:type="ComplexLiteral">
                                                                                        <value
xsi:type="dt:ED"

        value="If the patient is over 18 years old and not allergic to an ACE inhibitor and not receiving an
ACE inhibitor and LVEF is less than 40%, then consider prescribing an ACE inhibitor." />
                                                                                </value>
                                                                        </property>
                                                                </actionSentence>
                                                        </simpleAction>
                                                </subElements>
                                        </actionGroup>
                                        <actionGroup>
                                                <!-- ACE inhibitors -->
                                                <supportingResources>
                                                        <!-- Quality Measures -->
                                                        <resource>
                                                                <!-- Note that no identifier is given for
this quality measure, but
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                one could be if desired. -->
                                                        <title
                                                                value="ARRA EHR Stage 2
Meaningful Use Quality Measure by the Centers for Medicare and Medicaid Services (2012)" />
                                                        <location value="TBD" />
                                                        <description
                                                                value="ARRA EHR Stage 2
Meaningful Use Quality Measure by the Centers for Medicare and Medicaid Services (2012)" />
                                                </resource>
                                                <resource>
                                                        <title
                                                                value="Physician Consortium for
Performance Improvement Performance Measure by the American Medical Association (2012)" />
                                                        <location value="TBD" />
                                                        <description
                                                                value="Physician Consortium for
Performance Improvement Performance Measure by the American Medical Association (2012)" />
                                                </resource>
                                        </supportingResources>
                                        <behaviors>
                                                <!-- Note interaction between group selection
behavior and required
                                                        behavior. This says that the physician is
required to choose exactly one
                                                        of these medications unless
documentation is provided explaining why it was
                                                        not chosen. -->
                                                <behavior xsi:type="GroupSelectionBehavior"
value="ExactlyOne" />
                                                <behavior xsi:type="RequiredBehavior"
value="MustUnlessDocumented" />
                                        </behaviors>
                                        <title value="Angiotensin-Converting Enzyme Inhibitors"
/>
                                        <description
                                                value="For patients with diastolic heart failure
who are intolerant to an ACE inhibitor, consider the use of an ARB.  For patients with diastolic heart
failure, consider the use of an ACE inhibitor.  For patients with systolic heart failure who are
hemodynamically stable and are intolerant to an ACE inhibitor due to cough, use an ARB" />
                                        <representedConcepts>
                                                <!-- Coded this as an ACE inhibitors group -->
                                                <concept code="N0000000181"
codeSystem="2.16.840.1.113883.3.26.1.5"
                                                        codeSystemName="NDF-RT">
                                                        <dt:displayName value="Angiotensin-
Converting Enzyme Inhibitors" />
                                                </concept>
                                        </representedConcepts>
                                        <subElements>
                                                <actionGroup>
                                                        <!-- This is a group of orders in which
captopril is the orderable
                                                                substance for each. We use the
representedConcept to specify this explicitly,
                                                                and we use SentenceGroup
behavior to indicate that each item in the group
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
```

```


© 2014 Health Level Seven International. All rights reserved
```

```
                    code="001" codeSystemName="FDA Routes of Administration"
                    displayName="ORAL" />
        </property>
        <property name="frequency">
            <value xsi:type="ComplexLiteral">
                <value xsi:type="vmr:Schedule">
                    <vmr:cycle>
                        <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
                            <vmr:repeatCode code="??">
                                <dt:displayName value="3 times per day" />
                            </vmr:repeatCode>
                        </vmr:cycleTiming>
                    </vmr:cycle>
                </value>
            </value>
        </property>
        </element>
        </value>
                                                                    </property>
                                                                    <!-- vMR does
not have a way to say "3 times per day". Instead,                        a
dosing period must be specified. Since 3 times per day would be every 8  hours
for a fixed dosing period, we set 8 hours as the dosing period. The      value of
dosingPeriodIntervalIsImportant is used to determine whether the         interval
itself is important or not. If not, then a dosing period of "every       8 hours"
is interpreted as "3 times per day". -->                            <!-- Since
dosing frequency is typically a coded value in EMR's,                    perhaps
we should consider adding a coded frequency field to vMR? Otherwise,     this may
be very difficult for current EMR's to process. -->
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                                <!-- vmr r2
allows frequency as codes. -->

                                                                        </actionSentence>
                                                                </simpleAction>
                                                                <simpleAction
xsi:type="CreateAction">
                                                                        <textEquivalent
                                                                                value="captopril
12.5 milligram orally 3 times a day" />
                                                                        <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:SubstanceAdministrationProposal">
                                                                                <property
name="substance.substanceCode">
                                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"

        code="1998" codeSystemName="RxNorm" displayName="captopril" />
                                                                                </property>
                                                                                <property
name="dose">
                                                                                        <value
xsi:type="List">

        <element xsi:type="ObjectExpression" objectType="vmr:Dose">

        <property name="doseQuantity">

                <value xsi:type="PhysicalQuantityIntervalLiteral"

                        lowClosed="true" highClosed="true">

                        <low value="12.5" unit="mg" />

                        <high value="12.5" unit="mg" />

                </value>

        </property>

        <property name="deliveryRoute">

                <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"

                        code="001" codeSystemName="FDA Routes of Administration"

                        displayName="ORAL" />

        </property>

        <property name="frequency">

                <value xsi:type="ComplexLiteral">

                        <value xsi:type="vmr:Schedule">
```


© 2014 Health Level Seven International. All rights reserved

```
                    <vmr:cycle>
                        <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
                            <vmr:repeatCode code="??">
                                <dt:displayName value="3 times per day" />
                            </vmr:repeatCode>
                        </vmr:cycleTiming>
                    </vmr:cycle>
                </value>
            </value>
        </property>
        </element>
        </value>
                                                                    </property>
                                                                </actionSentence>
                                                            </simpleAction>
                                                            <simpleAction
xsi:type="CreateAction">
                                                                <textEquivalent
value="captopril 25 milligram orally 3 times a day" />
                                                                <actionSentence
xsi:type="ObjectExpression"
        objectType="vmr:SubstanceAdministrationProposal">
                                                                    <property
name="substance.substanceCode">
                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"
        code="1998" codeSystemName="RxNorm" displayName="captopril" />
                                                                    </property>
                                                                    <property
name="dose">
                                                                        <value
xsi:type="List">
        <element xsi:type="ObjectExpression" objectType="vmr:Dose">
        <property name="doseQuantity">
            <value xsi:type="PhysicalQuantityIntervalLiteral"
                lowClosed="true" highClosed="true">
                <low value="25" unit="mg" />
```


© 2014 Health Level Seven International. All rights reserved

```
                <high value="25" unit="mg" />
            </value>
        </property>
        <property name="deliveryRoute">
            <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"
                code="001" codeSystemName="FDA Routes of Administration"
                displayName="ORAL" />
        </property>
        <property name="frequency">
            <value xsi:type="ComplexLiteral">
                <value xsi:type="vmr:Schedule">
                    <vmr:cycle>
                        <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
                            <vmr:repeatCode code="??">
                                <dt:displayName value="3 times per day" />
                            </vmr:repeatCode>
                        </vmr:cycleTiming>
                    </vmr:cycle>
                </value>
            </value>
        </property>
        </element>
        </value>
                                                                    </property>
                                                                </actionSentence>
                                                            </simpleAction>
                                                        </subElements>
                                                    </actionGroup>
                                                    <simpleAction xsi:type="CreateAction">
                                                        <textEquivalent value="lisinopril 2.5
milligram orally once a day" />
                                                        <actionSentence
xsi:type="ObjectExpression"
```


© 2014 Health Level Seven International. All rights reserved

```
        objectType="vmr:SubstanceAdministrationProposal">
                                                                <property
name="substance.substanceCode">
                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"
                                                                                code="29046"
codeSystemName="RxNorm" displayName="Lisinopril" />
                                                                </property>
                                                                <property name="dose">
                                                                        <value xsi:type="List">
                                                                                <element
xsi:type="ObjectExpression" objectType="vmr:Dose">
        <property name="doseQuantity">
        <!-- units should be coded value, but ISO datatypes do not support
        a coded value for the units of meausre. We will work with the vMR group to
        address this. In the meantime, we are using coded values for all UOM from
        UCUM (https://phinvads.cdc.gov/vads/ViewCodeSystem.action?id=2.16.840.1.113883.6.8)
        until this issue is addressed ... -->
        <value xsi:type="PhysicalQuantityIntervalLiteral"
        lowClosed="true" highClosed="true">
        <low value="2.5" unit="mg" />
        <high value="2.5" unit="mg" />
        </value>
        </property>
        <property name="deliveryRoute">
        <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"
        code="001" codeSystemName="FDA Routes of Administration"
        displayName="ORAL" />
        </property>
        <property name="frequency">
        <value xsi:type="ComplexLiteral">
        <value xsi:type="vmr:Schedule">
                <vmr:cycle>
                        <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
```


© 2014 Health Level Seven International. All rights reserved

```
                    <vmr:repeatCode code="??">
                        <dt:displayName value="3 times per day" />
                    </vmr:repeatCode>
                </vmr:cycleTiming>
            </vmr:cycle>
        </value>
        </value>
        </property>
                                                                                    </element>
                                                                                </value>
                                                                            </property>
                                                                        </actionSentence>
                                                                    </simpleAction>
                                                                </subElements>
                                                            </actionGroup>
                                                            <actionGroup>
                                                                <!-- Beta Blockers -->
                                                                <supportingResources>
                                                                    <!-- Quality Measures, none with IDs -->
                                                                    <resource>
                                                                        <title
                                                                            value="ARRA EHR Stage 2
Meaningful Use Quality Measure by the Centers for Medicare and Medicaid Services (2012)" />
                                                                        <location
        value="https://www.zynx.com/Reference/Content.aspx?ItemID=216994" />
                                                                        <description
                                                                            value="ARRA EHR Stage 2
Meaningful Use Quality Measure by the Centers for Medicare and Medicaid Services (2012)" />
                                                                    </resource>
                                                                    <resource>
                                                                        <title
                                                                            value="Physician Consortium for
Performance Improvement Performance Measure by the American Medical Association (2012)" />
                                                                        <location
        value="https://www.zynx.com/Reference/Content.aspx?ItemID=216994" />
                                                                        <description
                                                                            value="Physician Consortium for
Performance Improvement Performance Measure by the American Medical Association (2012)" />
                                                                    </resource>
                                                                    <resource>
                                                                        <title
                                                                            value="Get With The Guidelines
Achievement Measure by the American Heart Association/American Stroke Association (2012)" />
                                                                        <location
        value="https://www.zynx.com/Reference/Content.aspx?ItemID=216994" />
                                                                        <description
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                    value="Get With The Guidelines
Achievement Measure by the American Heart Association/American Stroke Association (2012)" />
                                                        </resource>
                                                        <resource>
                                                                <title
                                                                    value="Target Measure by the
American Heart Association/American Stroke Association (2012)" />
                                                                <location

        value="https://www.zynx.com/Reference/Content.aspx?ItemID=216994" />
                                                                <description
                                                                    value="Target Measure by the
American Heart Association/American Stroke Association (2012)" />
                                                        </resource>
                                                </supportingResources>
                                                <behaviors>
                                                        <!-- Note interaction between group selection
behavior and required
                                                                behavior. -->
                                                        <behavior xsi:type="GroupSelectionBehavior"
value="ExactlyOne" />
                                                        <behavior xsi:type="RequiredBehavior"
value="MustUnlessDocumented" />
                                                </behaviors>
                                                <title value="Beta-Blockers" />
                                                <description
                                                        value="For patients with diastolic heart failure
and a previous MI, use a beta-blocker.  For patients with diastolic heart failure, consider the use of a beta-
blocker.  For patients with systolic heart failure who are hemodynamically stable, use beta-blocker therapy
(eg, bisoprolol, carvedilol, metoprolol extended release)." />
                                                <representedConcepts>
                                                        <!-- Coded this as a beta blockers group -->
                                                        <concept code="N0000000161"
codeSystem="2.16.840.1.113883.3.26.1.5"
                                                                codeSystemName="NDF-RT">
                                                                <dt:displayName value="Adrenergic
beta-Antagonists" />
                                                        </concept>
                                                </representedConcepts>
                                                <subElements>
                                                        <simpleAction xsi:type="CreateAction">
                                                                <textEquivalent
                                                                    value="carvedilol 3.125
milligram orally 2 times a day" />
                                                                <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:SubstanceAdministrationProposal">
                                                                    <property
name="substance.substanceCode">
                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"
                                                                            code="20352"
codeSystemName="RxNorm" displayName="carvedilol" />
                                                                    </property>
                                                                    <property name="dose">
                                                                        <value xsi:type="List">
```


© 2014 Health Level Seven International. All rights reserved

```
<element xsi:type="ObjectExpression" objectType="vmr:Dose">
    <property name="doseQuantity">
    <value xsi:type="PhysicalQuantityIntervalLiteral"
    lowClosed="true" highClosed="true">
    <low value="3.125" unit="mg" />
    <high value="3.125" unit="mg" />
    </value>
    </property>
    <property name="deliveryRoute">
    <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"
    code="001" codeSystemName="FDA Routes of Administration"
    displayName="ORAL" />
    </property>
    <!-- vMR does not have a way to say "2 times per day". Instead,
    a dosing period must be specified. Since 2 times per day would be every 12
    hours for a fixed dosing period, we set 12 hours as the dosing period. The
    value of dosingPeriodIntervalIsImportant is used to determine whether the
    interval itself is important or not. If not, then a dosing period of "every
    12 hours" is interpreted as "2 times per day". -->
    <!-- Since dosing frequency is typically a coded value in EMR's,
    perhaps we should consider adding a coded frequency field to vMR? Otherwise,
    this may be very difficult for current EMR's to process. -->
    <property name="frequency">
    <value xsi:type="ComplexLiteral">
    <value xsi:type="vmr:Schedule">
        <vmr:cycle>
            <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
                <vmr:repeatCode code="??">
```


© 2014 Health Level Seven International. All rights reserved

```
                                        <dt:displayName value="2 times per day" />
                                    </vmr:repeatCode>
                                </vmr:cycleTiming>
                            </vmr:cycle>
                        </value>
                        </value>
                        </property>
                                                                                                    </element>
                                                                                                </value>
                                                                                            </property>
                                                                                        </actionSentence>
                                                                                    </simpleAction>
                                                                                    <actionGroup>  <!-- Note that this sentence
group has the same orderable item in both cases
                                                                                                    - "metoprolol succinate SR 25
mg 24 hr tab", but at two different dose levels.
                                                                                                    The representedConcept for the
group captures this common orderable, and
                                                                                                    each CreateAction in the sub-
elements creates each of the two dosing levels. -->
                                                                                        <behaviors>
                                                                                                    <behavior
xsi:type="GroupOrganizationBehavior" value="SentenceGroup" />
                                                                                        </behaviors>
                                                                                        <representedConcepts>
                                                                                                    <concept
codeSystem="2.16.840.1.113883.6.88" code="TBD - now missing from RxNorm"
                        codeSystemName="RxNorm">
                                                                                                                <dt:displayName
                                                                                                                            value="24 HR
Metoprolol Succinate 25 MG Extended Release Tablet" />
                                                                                                    </concept>
                                                                                        </representedConcepts>
                                                                                        <subElements>
                                                                                                    <simpleAction
xsi:type="CreateAction">
                                                                                                                <textEquivalent
                        value="metoprolol succinate SR 25 mg 24 hr tab 0.5 tablet orally once a day" />
                                                                                                                <actionSentence
xsi:type="ObjectExpression"
                        objectType="vmr:SubstanceAdministrationProposal">
                                                                                                                            <property
name="substance.substanceCode">
                                                                                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"
                        code="TBD - now missing from RxNorm" codeSystemName="RxNorm"
```


© 2014 Health Level Seven International. All rights reserved

```
displayName="24 HR Metoprolol Succinate 25 MG Extended Release Tablet" />
                                                                                    </property>
                                                                                    <property
name="dose">
                                                                                        <value
xsi:type="List">
        <element xsi:type="ObjectExpression" objectType="vmr:Dose">

        <property name="doseQuantity">
                <!-- units should be coded value, but ISO datatypes do not
                        support a coded value for the units of measure. We will work with the vMR
                        group to address this. In the meantime, we are using coded values for all
                        UOM from UCUM
(https://phinvads.cdc.gov/vads/ViewCodeSystem.action?id=2.16.840.1.113883.6.8)
                        until this issue is addressed ... -->
                <value xsi:type="PhysicalQuantityIntervalLiteral"
                        lowClosed="true" highClosed="true">
                        <low value="0.5" unit="tbl" />
                        <high value="0.5" unit="tbl" />
                </value>
        </property>
        <property name="deliveryRoute">
                <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"
                        code="001" codeSystemName="FDA Routes of Administration"
                        displayName="ORAL" />
        </property>
        <property name="frequency">
                <value xsi:type="ComplexLiteral">
                        <value xsi:type="vmr:Schedule">
                                <vmr:cycle>
                                        <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
                                                <vmr:repeatCode code="??">
```


© 2014 Health Level Seven International. All rights reserved

```
                                                <dt:displayName value="1 times per day" />
                                            </vmr:repeatCode>
                                        </vmr:cycleTiming>
                                    </vmr:cycle>
                                </value>
                            </value>
                        </property>
                        </element>
                        </value>
                                                                                    </property>
                                                                                </actionSentence>
                                                                            </simpleAction>
                                                                            <simpleAction
xsi:type="CreateAction">
                                                                                <textEquivalent
                        value="metoprolol succinate SR 25 mg 24 hr tab 1 tablet orally once a day" />
                                                                                <actionSentence
xsi:type="ObjectExpression"
                        objectType="vmr:SubstanceAdministrationProposal">
                                                                                    <property
name="substance.substanceCode">
                                                                                        <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"
                        code="TBD - now missing from RxNorm" codeSystemName="RxNorm"
                        displayName="24 HR Metoprolol Succinate 25 MG Extended Release Tablet" />
                                                                                    </property>
                                                                                    <property
name="dose">
                                                                                        <value
xsi:type="List">
                        <element xsi:type="ObjectExpression" objectType="vmr:Dose">

                        <property name="doseQuantity">
                            <!-- units should be coded value, but ISO datatypes do not
                                support a coded value for the units of measure. We will work with the vMR
                                group to address this. In the meantime, we are using coded values for all
                                UOM from UCUM
(https://phinvads.cdc.gov/vads/ViewCodeSystem.action?id=2.16.840.1.113883.6.8)
```


© 2014 Health Level Seven International. All rights reserved

```
                    until this issue is addressed ... -->
            <value xsi:type="PhysicalQuantityIntervalLiteral"
                    lowClosed="true" highClosed="true">
                    <low value="1" unit="tbl" />
                    <high value="1" unit="tbl" />
            </value>
    </property>
    <property name="deliveryRoute">
            <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"
                    code="001" codeSystemName="FDA Routes of Administration"
                    displayName="ORAL" />
    </property>
    <property name="frequency">
            <value xsi:type="ComplexLiteral">
                    <value xsi:type="vmr:Schedule">
                            <vmr:cycle>
                                    <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">
                                            <vmr:repeatCode code="??">
                                                    <dt:displayName value="1 times per day" />
                                            </vmr:repeatCode>
                                    </vmr:cycleTiming>
                            </vmr:cycle>
                    </value>
            </value>
    </property>
    </element>
    </value>
                                                                    </property>
                                                            </actionSentence>
                                                    </simpleAction>
```


© 2014 Health Level Seven International. All rights reserved

```
                                                    </subElements>
                                                </actionGroup>
                                                <simpleAction xsi:type="CreateAction">
                                                    <textEquivalent
                                                        value="metoprolol succinate SR
50 mg 24 hr tab 1 tablet orally once a day" />
                                                    <actionSentence
xsi:type="ObjectExpression"

        objectType="vmr:SubstanceAdministrationProposal">
                                                        <property
name="substance.substanceCode">
                                                            <value
xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.6.88"
                                                                code="TBD -
now missing from RxNorm" codeSystemName="RxNorm"

        displayName="24 HR Metoprolol Succinate 50 MG Extended Release Tablet" />
                                                        </property>
                                                        <property name="dose">
                                                            <value xsi:type="List">
                                                                <element
xsi:type="ObjectExpression" objectType="vmr:Dose">


        <property name="doseQuantity">

        <!-- units should be coded value, but ISO datatypes do not support

        a coded value for the units of measure. We will work with the vMR group to

        address this. In the meantime, we are using coded values for all UOM from

        UCUM (https://phinvads.cdc.gov/vads/ViewCodeSystem.action?id=2.16.840.1.113883.6.8)

        until this issue is addressed ... -->

        <value xsi:type="PhysicalQuantityIntervalLiteral"

        lowClosed="true" highClosed="true">

        <low value="1" unit="tbl" />

        <high value="1" unit="tbl" />

        </value>

        </property>

        <property name="deliveryRoute">

        <value xsi:type="CodeLiteral" codeSystem="2.16.840.1.113883.3.26.1.1.1"

        code="001" codeSystemName="FDA Routes of Administration"

        displayName="ORAL" />
```


© 2014 Health Level Seven International. All rights reserved

```
        </property>

        <property name="frequency">

        <value xsi:type="ComplexLiteral">

        <value xsi:type="vmr:Schedule">

                <vmr:cycle>

                        <vmr:cycleTiming xsi:type="vmr:CodedRecurringEvent">

                                <vmr:repeatCode code="??">

                                        <dt:displayName value="1 times per day" />

                                </vmr:repeatCode>

                        </vmr:cycleTiming>

                </vmr:cycle>

        </value>

        </value>

        </property>
                                                                                                        </element>
                                                                                                </value>
                                                                                        </property>
                                                                                </actionSentence>
                                                                        </simpleAction>
                                                                </subElements>
                                                        </actionGroup>
                                                </subElements>
                                        </actionGroup>
                                        <actionGroup>
                                                <!-- Laboratory -->
                                                <title value="Laboratory" />
                                                <representedConcepts>
                                                        <!-- Coded this as a laboratory tests group -->
                                                        <concept code="15220000"
codeSystem="2.16.840.1.113883.6.96"
                                                                codeSystemName="SNOMED-CT">
                                                                <dt:displayName value="Laboratory Test (procedure)" />
                                                        </concept>
                                                </representedConcepts>
                                                <subElements>
                                                        <simpleAction xsi:type="CreateAction">
                                                                <textEquivalent value="B-type natriuretc peptide (BNP)"
/>
                                                                <actionSentence xsi:type="ObjectExpression"
                                                                        objectType="vmr:ProcedureProposal">
                                                                        <property name="procedureCode">
                                                                                <value xsi:type="CodeLiteral"
codeSystem="2.16.840.1.113883.6.1"
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                                    code="42637-9"
codeSystemName="LOINC"
                                                                                    displayName="Natriuretic
peptide B [Mass/volume] in Blood" />
                                                                        </property>
                                                                    </actionSentence>
                                                                </simpleAction>
                                                                <simpleAction xsi:type="CreateAction">
                                                                    <textEquivalent
                                                                        value="Complete blood cell count with
automated white blood cell differential" />
                                                                    <actionSentence xsi:type="ObjectExpression"
                                                                        objectType="vmr:ProcedureProposal">
                                                                        <property name="procedureCode">
                                                                            <value xsi:type="CodeLiteral"
codeSystem="2.16.840.1.113883.6.1"
                                                                                    code="57021-8"
codeSystemName="LOINC"
                                                                                    displayName="CBC W Auto
Differential panel in Blood" />
                                                                        </property>
                                                                    </actionSentence>
                                                                </simpleAction>
                                                            </subElements>
                                                        </actionGroup>
                                                        <actionGroup>
                                                            <!-- Diagnostic Tests -->
                                                            <title value="Diagnostic Tests" />
                                                            <representedConcepts>
                                                                <!-- Coded this as a diagnostic tests group -->
                                                                <concept code="103693007"
codeSystem="2.16.840.1.113883.6.96"
                                                                    codeSystemName="SNOMED-CT">
                                                                    <dt:displayName value="Diagnostic procedure
(procedure)" />
                                                                </concept>
                                                            </representedConcepts>
                                                            <subElements>
                                                                <simpleAction xsi:type="CreateAction">
                                                                    <textEquivalent value="Radiograph, chest, 2 views" />
                                                                    <actionSentence xsi:type="ObjectExpression"
                                                                        objectType="vmr:ProcedureProposal">
                                                                        <property name="procedureCode">
                                                                            <value xsi:type="CodeLiteral"
codeSystem="2.16.840.1.113883.6.96"
                                                                                    code="399208008"
codeSystemName="SNOMED-CT" displayName="Plain chest X-ray (procedure)" />
                                                                        </property>
                                                                    </actionSentence>
                                                                </simpleAction>
                                                                <simpleAction xsi:type="CreateAction">
                                                                    <textEquivalent value="12-lead ECG " />
                                                                    <actionSentence xsi:type="ObjectExpression"
                                                                        objectType="vmr:ProcedureProposal">
                                                                        <property name="procedureCode">
                                                                            <value xsi:type="CodeLiteral"
codeSystem="2.16.840.1.113883.6.96"
```


© 2014 Health Level Seven International. All rights reserved

```
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                code="268400002"
codeSystemName="SNOMED-CT" displayName="12 lead ECG (procedure)" />
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
</property>
</actionSentence>
</simpleAction>
</subElements>
</actionGroup>
<actionGroup>
<!-- Additional Orders -->
<title value="Additional Orders" />
<subElements>
<!-- There are currently no elements or attibutes defined to help display
these references to other order sets within this parent order set. More input
is needed from the community to determine whether such are needed. Such attributes
should only be included here if their values are specific to this order set;
otherwise, the referenced order sets should be retrieved for any specific
values needed to display these references in the target system. In addition,
note that these references should be resolved at the time that this artifact
is imported into the target system, with the referenced knowledge artifact
being included "in-line" with the rest of this parent order set, or represented
with an internal system link, depending on the capabilities of the target
system. -->
<actionGroupReference
root="www.zynx.com/cds/orderset/RespiratoryProtocol" extension="1234"
version="2.7" />
<actionGroupReference
root="www.zynx.com/cds/orderset/RegularInsulinSlidingScale"
extension="43064" version="1.5" />
</subElements>
</actionGroup>
</subElements>
</actionGroup>
</knowledgeDocument>
```


© 2014 Health Level Seven International. All rights reserved