HL7IG\_CDS\_KNART\_R1\_D1\_2013JAN



**HL7 Version 3 Standard: Clinical Decision Support Knowledge Artifact Specification, Release 1.2**

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**](http://www.hl7.org/legal/ippolicy.cfm?ref=nav).

**Identifying Information for Specification:**

**Specification Name and Release Number:** HL7 Version 3 Standard: Clinical Decision Support Knowledge Artifact Specification, Release 1.2

**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 2

**Note Regarding Changes since Last Version:**

The following changes are included in this release of the specification:

1. Enumerations throughout the specification were simplified to remove the extension mechanism. This change is backwards compatible, so long as no artifact was using an extended enumeration value.
2. Minor schema definition refactoring was performed to make schema definitions more consistent and reusable. These changes are backwards compatible.
3. UpdateAction\actionSentence and Actor\actor elements were changed to be of type Expression. This change is not backwards compatible. Artifacts that contain instances of these elements will need to be updated.
4. An annotation element was added to the base Expression type to allow expression logic to be decorated with application-specific information such as debug symbols or translation source descriptions. This change is backwards compatible.
5. A new artifact type of ‘Library’ was introduced to allow artifacts to reuse components such as expression logic and action definitions. This change also enables the use of expression logic defined in formats other than HeD Schema.
6. The mechanism for referencing actions was changed to use libraries, rather than the actionGroupReference element. This change is not backwards compatible. Artifacts that contain instances of this element will need to be updated.
7. Reference documentation was reorganized and streamlined to reduce the size of this specification, as well as to better organize the reference material to reflect the order of presentation in the document.
8. HeD Schema Framework was updated to use the latest version of the specification.

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.

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 specification and 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 specification is designed to be flexible and reusable, and to provide a baseline for CDS vendors and CDS Knowledge Artifact implementers.

Table of Contents

[1 Introduction 1](#_Toc386725495)

[1.1 Design Approach and Rationale 1](#_Toc386725496)

[1.1.1 How a CDS Knowledge Artifact Works 3](#_Toc386725497)

[1.1.2 Lifecycle of a CDS knowledge artifact 3](#_Toc386725498)

[1.1.3 Extensibility of the HeD CDS Knowledge Artifact schema 4](#_Toc386725499)

[1.1.4 Representation Rationale 4](#_Toc386725500)

[1.2 Audience 6](#_Toc386725501)

[1.2.1 Requisite Knowledge 6](#_Toc386725502)

[1.3 Scope of the Specification 7](#_Toc386725503)

[1.3.1 Contents of the Proposed Ballot 7](#_Toc386725504)

[1.4 Alignment to HeD Artifact Sharing Use Case 8](#_Toc386725505)

[1.4.1 Use Case Assumptions and Conditions 9](#_Toc386725506)

[1.4.2 Usage Conformance Testing Recommendations 9](#_Toc386725507)

[1.5 Organization of this Specification 9](#_Toc386725508)

[1.5.1 Definitions of Terms 9](#_Toc386725509)

[1.5.2 Definitions of Actors 11](#_Toc386725510)

[1.5.3 Conventions Used 11](#_Toc386725511)

[1.5.4 Schema Roadmap 14](#_Toc386725512)

[2 CDS Knowledge Artifact Types 15](#_Toc386725513)

[2.1 Conformance to the Health eDecisions Knowledge Artifact Specification 15](#_Toc386725514)

[2.2 HeD Knowledge Artifacts 15](#_Toc386725515)

[2.2.1 Metadata Elements 15](#_Toc386725516)

[2.2.2 Def Elements 16](#_Toc386725517)

[2.2.3 Expression Elements 16](#_Toc386725518)

[2.2.4 Trigger Elements 16](#_Toc386725519)

[2.2.5 Condition Elements 16](#_Toc386725520)

[2.2.6 Behavior Elements 16](#_Toc386725521)

[2.2.7 Action Elements 16](#_Toc386725522)

[2.3 Event Condition Action (ECA) Rules 17](#_Toc386725523)

[2.3.1 Conceptual Overview 17](#_Toc386725524)

[2.3.2 ECA Rule Conformance Requirements 18](#_Toc386725525)

[2.4 Order Sets 20](#_Toc386725526)

[2.4.1 Conceptual Overview 20](#_Toc386725527)

[2.4.2 Order Set Conformance Requirements 22](#_Toc386725528)

[2.5 Documentation Templates 23](#_Toc386725529)

[2.5.1 Conceptual Overview 23](#_Toc386725530)

[2.5.2 Documentation Template Conformance Requirements 25](#_Toc386725531)

[2.6 Libraries 27](#_Toc386725532)

[2.6.1 Conceptual Overview 27](#_Toc386725533)

[2.6.2 Library Conformance Requirements 28](#_Toc386725534)

[2.7 Value Sets and Vocabulary Codes 28](#_Toc386725535)

[2.7.1 Value Set Use Case 29](#_Toc386725536)

[2.7.2 Coded Data versus Free Text 29](#_Toc386725537)

[3 CDS Knowledge Artifact Components 30](#_Toc386725538)

[3.1 Metadata 30](#_Toc386725539)

[3.1.1 Clinical Data Models 30](#_Toc386725540)

[3.2 Actions 31](#_Toc386725541)

[3.2.1 Types of Actions 31](#_Toc386725542)

[3.2.2 Creating Action Sentence Expressions 33](#_Toc386725543)

[3.2.3 Organizing Actions 34](#_Toc386725544)

[3.2.4 Conditional Actions 34](#_Toc386725545)

[3.2.5 Behaviors 35](#_Toc386725546)

[3.3 Triggers and Events 40](#_Toc386725547)

[3.3.1 DataEvent 40](#_Toc386725548)

[3.3.2 PeriodicEvent 40](#_Toc386725549)

[3.4 Expressions 40](#_Toc386725550)

[3.4.1 Basic Values (Scalars) 41](#_Toc386725551)

[3.4.2 Comparison Operators 42](#_Toc386725552)

[3.4.3 Logical Operators 43](#_Toc386725553)

[3.4.4 Nullological Operators 44](#_Toc386725554)

[3.4.5 Conditional Operators 44](#_Toc386725555)

[3.4.6 Arithmetic Operators 46](#_Toc386725556)

[3.4.7 String Operators 47](#_Toc386725557)

[3.4.8 Date and Time Operators 48](#_Toc386725558)

[3.4.9 List Values 49](#_Toc386725559)

[3.4.10 Aggregate Operators 56](#_Toc386725560)

[3.4.11 Interval Values 58](#_Toc386725561)

[3.4.12 Structured Values 61](#_Toc386725562)

[3.4.13 Reusing Expressions 63](#_Toc386725563)

[3.4.14 External Data 63](#_Toc386725564)

[3.4.15 Parameters 66](#_Toc386725565)

[3.5 Extending the Schema 67](#_Toc386725566)

[3.5.1 Extending Types 67](#_Toc386725567)

[3.5.2 Other Extensions and Modifications 67](#_Toc386725571)

[4 Language Implementation 69](#_Toc386725572)

[4.1 Clinical Data Retrieval in HeDS Artifacts 69](#_Toc386725573)

[4.1.1 Defining Clinical Data 69](#_Toc386725574)

[4.1.2 Conformance Levels 70](#_Toc386725575)

[4.1.3 Artifact Data Requirements 70](#_Toc386725576)

[4.2 Expression Language Conceptual Model 71](#_Toc386725577)

[4.2.1 Data Model 71](#_Toc386725578)

[4.2.2 Language Elements 73](#_Toc386725579)

[4.2.3 Semantic Validation 73](#_Toc386725580)

[4.2.4 Execution Model 74](#_Toc386725581)

[5 Schema References 77](#_Toc386725582)

[5.1 Datatypes 77](#_Toc386725583)

[5.1.1 AD 77](#_Toc386725584)

[5.1.2 ADXP 78](#_Toc386725585)

[5.1.3 ANY 79](#_Toc386725586)

[5.1.4 BL 80](#_Toc386725587)

[5.1.5 CD 81](#_Toc386725588)

[5.1.6 CO 86](#_Toc386725589)

[5.1.7 CS 88](#_Toc386725590)

[5.1.8 Code 89](#_Toc386725591)

[5.1.9 Decimal 89](#_Toc386725592)

[5.1.10 ED 89](#_Toc386725593)

[5.1.11 EN 95](#_Toc386725594)

[5.1.12 ENXP 96](#_Toc386725595)

[5.1.13 HXIT 97](#_Toc386725596)

[5.1.14 II 98](#_Toc386725597)

[5.1.15 INT 100](#_Toc386725598)

[5.1.16 IVL 101](#_Toc386725599)

[5.1.17 IVL\_CO 102](#_Toc386725600)

[5.1.18 IVL\_INT 103](#_Toc386725601)

[5.1.19 IVL\_PQ 105](#_Toc386725602)

[5.1.20 IVL\_QTY 106](#_Toc386725603)

[5.1.21 IVL\_REAL 107](#_Toc386725604)

[5.1.22 IVL\_TS 109](#_Toc386725605)

[5.1.23 PIVL\_TS 110](#_Toc386725606)

[5.1.24 PQ 112](#_Toc386725607)

[5.1.25 QSET 114](#_Toc386725608)

[5.1.26 QTY 115](#_Toc386725609)

[5.1.27 REAL 116](#_Toc386725610)

[5.1.28 RTO 117](#_Toc386725611)

[5.1.29 ST 118](#_Toc386725612)

[5.1.30 TEL 119](#_Toc386725613)

[5.1.31 TS 121](#_Toc386725614)

[5.1.32 TimeStamp 121](#_Toc386725615)

[5.1.33 Uid 121](#_Toc386725616)

[5.1.34 Uri 122](#_Toc386725617)

[5.1.35 XP 122](#_Toc386725618)

[5.1.36 set\_EntityNamePartQualifier 123](#_Toc386725619)

[5.1.37 set\_EntityNameUse 123](#_Toc386725620)

[5.1.38 set\_PostalAddressUse 124](#_Toc386725621)

[5.1.39 set\_TelecommunicationAddressUse 124](#_Toc386725622)

[5.1.40 set\_TelecommunicationCapability 124](#_Toc386725623)

[5.1.41 AddressPartType 125](#_Toc386725624)

[5.1.42 CalendarCycle 130](#_Toc386725625)

[5.1.43 Compression 133](#_Toc386725626)

[5.1.44 EntityNamePartQualifier 134](#_Toc386725627)

[5.1.45 EntityNamePartType 137](#_Toc386725628)

[5.1.46 EntityNameUse 138](#_Toc386725629)

[5.1.47 IntegrityCheckAlgorithm 142](#_Toc386725630)

[5.1.48 PostalAddressUse 142](#_Toc386725631)

[5.1.49 TelecommunicationAddressUse 146](#_Toc386725632)

[5.1.50 TelecommunicationCapability 148](#_Toc386725633)

[5.2 Base 149](#_Toc386725634)

[5.2.1 Evidence 150](#_Toc386725635)

[5.2.2 InlineResource 151](#_Toc386725636)

[5.2.3 KnowledgeResource 152](#_Toc386725637)

[5.2.4 Organization 155](#_Toc386725638)

[5.2.5 Party 156](#_Toc386725639)

[5.2.6 Person 157](#_Toc386725640)

[5.2.7 ResourceRelationshipReference 158](#_Toc386725641)

[5.2.8 SupportingEvidence 160](#_Toc386725642)

[5.2.9 SupportingResource 160](#_Toc386725643)

[5.2.10 VersionedIdentifier 161](#_Toc386725644)

[5.3 Behavior 162](#_Toc386725645)

[5.3.1 Behavior 162](#_Toc386725646)

[5.3.2 Behaviors 163](#_Toc386725647)

[5.3.3 GroupOrganizationBehavior 163](#_Toc386725648)

[5.3.4 GroupSelectionBehavior 164](#_Toc386725649)

[5.3.5 PrecheckBehavior 164](#_Toc386725650)

[5.3.6 RequiredBehavior 165](#_Toc386725651)

[5.4 Enum 166](#_Toc386725652)

[5.4.1 ArtifactLifeCycleEventType 166](#_Toc386725653)

[5.4.2 ArtifactStatusType 167](#_Toc386725654)

[5.4.3 ArtifactType 168](#_Toc386725655)

[5.4.4 Cardinality 169](#_Toc386725656)

[5.4.5 ConditionRoleType 170](#_Toc386725657)

[5.4.6 ContributorType 171](#_Toc386725658)

[5.4.7 CoverageType 172](#_Toc386725659)

[5.4.8 DataEventType 173](#_Toc386725660)

[5.4.9 EventType 175](#_Toc386725661)

[5.4.10 GroupOrganizationBehaviorType 175](#_Toc386725662)

[5.4.11 GroupSelectionBehaviorType 176](#_Toc386725663)

[5.4.12 PrecheckBehaviorType 178](#_Toc386725664)

[5.4.13 RangeConstraintType 179](#_Toc386725665)

[5.4.14 RequiredBehaviorType 179](#_Toc386725666)

[5.4.15 ResourceRelationshipType 180](#_Toc386725667)

[5.4.16 ValueType 182](#_Toc386725668)

[5.5 Metadata 185](#_Toc386725669)

[5.5.1 ArtifactLifeCycleEvent 185](#_Toc386725670)

[5.5.2 Contribution 186](#_Toc386725671)

[5.5.3 Coverage 187](#_Toc386725672)

[5.5.4 LibraryReference 188](#_Toc386725673)

[5.5.5 Metadata 189](#_Toc386725674)

[5.5.6 ModelReference 199](#_Toc386725675)

[5.5.7 RightsDeclaration 200](#_Toc386725676)

[5.6 Expression 201](#_Toc386725677)

[5.6.1 AggregateOperators 201](#_Toc386725678)

[5.6.2 ArithmeticOperators 208](#_Toc386725679)

[5.6.3 ComparisonOperators 216](#_Toc386725680)

[5.6.4 ConditionalOperators 220](#_Toc386725681)

[5.6.5 CoreElements 223](#_Toc386725682)

[5.6.6 DateTimeOperators 241](#_Toc386725683)

[5.6.7 IntervalOperators 249](#_Toc386725684)

[5.6.8 ListOperators 258](#_Toc386725685)

[5.6.9 LogicalOperators 272](#_Toc386725686)

[5.6.10 MembershipOperators 273](#_Toc386725687)

[5.6.11 NullologicalOperators 278](#_Toc386725688)

[5.6.12 StringOperators 280](#_Toc386725689)

[5.6.13 StructuredValueOperators 285](#_Toc386725690)

[5.6.14 ValueOperators 291](#_Toc386725691)

[5.7 Literalexpression 296](#_Toc386725692)

[5.7.1 AddressLiteral 296](#_Toc386725693)

[5.7.2 BooleanLiteral 296](#_Toc386725694)

[5.7.3 CodeLiteral 297](#_Toc386725695)

[5.7.4 CodedOrdinalLiteral 299](#_Toc386725696)

[5.7.5 EntityNameLiteral 300](#_Toc386725697)

[5.7.6 IdentifierLiteral 301](#_Toc386725698)

[5.7.7 IntegerIntervalLiteral 302](#_Toc386725699)

[5.7.8 IntegerLiteral 303](#_Toc386725700)

[5.7.9 PeriodLiteral 304](#_Toc386725701)

[5.7.10 PhysicalQuantityIntervalLiteral 305](#_Toc386725702)

[5.7.11 PhysicalQuantityLiteral 306](#_Toc386725703)

[5.7.12 QuantityIntervalLiteral 307](#_Toc386725704)

[5.7.13 RatioLiteral 308](#_Toc386725705)

[5.7.14 RealIntervalLiteral 309](#_Toc386725706)

[5.7.15 RealLiteral 310](#_Toc386725707)

[5.7.16 SimpleCodeLiteral 311](#_Toc386725708)

[5.7.17 StringLiteral 311](#_Toc386725709)

[5.7.18 TimestampIntervalLiteral 312](#_Toc386725710)

[5.7.19 TimestampLiteral 313](#_Toc386725711)

[5.7.20 UrlLiteral 314](#_Toc386725712)

[5.8 Clinicalexpression 315](#_Toc386725713)

[5.8.1 ClinicalRequest 316](#_Toc386725714)

[5.8.2 DataRequest 319](#_Toc386725715)

[5.8.3 InValueSet 319](#_Toc386725716)

[5.8.4 RequestBase 320](#_Toc386725717)

[5.8.5 SetSubsumes 322](#_Toc386725718)

[5.8.6 Subsumes 323](#_Toc386725719)

[5.8.7 ValueSet 324](#_Toc386725720)

[5.8.8 RequestCardinality 325](#_Toc386725721)

[5.9 Action 326](#_Toc386725722)

[5.9.1 ActionBase 326](#_Toc386725723)

[5.9.2 ActionGroup 330](#_Toc386725724)

[5.9.3 ActionRef 331](#_Toc386725725)

[5.9.4 Actor 332](#_Toc386725726)

[5.9.5 AtomicAction 333](#_Toc386725727)

[5.9.6 CollectInformationAction 334](#_Toc386725728)

[5.9.7 CreateAction 335](#_Toc386725729)

[5.9.8 DeclareResponseAction 335](#_Toc386725730)

[5.9.9 DocumentationItem 336](#_Toc386725731)

[5.9.10 ExpressionConstraint 339](#_Toc386725732)

[5.9.11 FireEventAction 340](#_Toc386725733)

[5.9.12 ItemDefinition 340](#_Toc386725734)

[5.9.13 ListConstraint 341](#_Toc386725735)

[5.9.14 RangeConstraint 343](#_Toc386725736)

[5.9.15 RemoveAction 344](#_Toc386725737)

[5.9.16 ResponseBinding 345](#_Toc386725738)

[5.9.17 UpdateAction 346](#_Toc386725739)

[5.9.18 ValueSetConstraint 346](#_Toc386725740)

[5.9.19 actionGroup element type 347](#_Toc386725741)

[5.10 Knowledgedocument 348](#_Toc386725742)

[5.10.1 Condition 348](#_Toc386725743)

[5.10.2 Conditions 349](#_Toc386725744)

[5.10.3 Trigger 350](#_Toc386725745)

[5.10.4 Triggers 351](#_Toc386725746)

[5.10.5 knowledgeDocument 351](#_Toc386725747)

[6 Appendix A – Referenced documents 356](#_Toc386725748)

[7 appendix b – acronyms 358](#_Toc386725749)

[8 Appendix C – Definitions 359](#_Toc386725750)

[9 Appendix D – HeD Schema Framework 360](#_Toc386725751)

[9.1 Overview 360](#_Toc386725752)

[9.1.1 Technology 360](#_Toc386725753)

[9.1.2 Solution Structure 360](#_Toc386725754)

[9.1.3 Design Goals 361](#_Toc386725755)

[9.2 Components 362](#_Toc386725756)

[9.2.1 Maps 362](#_Toc386725757)

[9.2.2 Model 363](#_Toc386725758)

[9.2.3 Reading 363](#_Toc386725759)

[9.2.4 Writing 363](#_Toc386725760)

[9.3 Verification 363](#_Toc386725761)

[9.3.1 Type Resolution 364](#_Toc386725762)

[9.3.2 Operator Resolution 364](#_Toc386725763)

[9.3.3 Node Verification 365](#_Toc386725764)

[9.3.4 Symbol Resolution 366](#_Toc386725765)

[9.4 Translation 366](#_Toc386725766)

[9.4.1 Artifact Translation 366](#_Toc386725767)

[9.4.2 Node Translation 366](#_Toc386725768)

[9.4.3 Model Translation 367](#_Toc386725769)

[9.5 CREF Translation 367](#_Toc386725770)

[9.5.1 Metadata 367](#_Toc386725771)

[9.5.2 Syntax 368](#_Toc386725772)

[9.5.3 Model 371](#_Toc386725773)

[9.5.4 Value Sets 374](#_Toc386725774)

[9.5.5 Guidance 374](#_Toc386725775)

[10 Appendix E – Examples 377](#_Toc386725776)

[10.1 FLACC Example 377](#_Toc386725777)

[10.2 RespiratoryOrder Example 384](#_Toc386725778)

[10.3 DopamineComplexIVOrderWithComplexLiteral Example 390](#_Toc386725779)

[10.4 HeartFailureAdmissionToMedSurgOrderSet Example 395](#_Toc386725780)

# 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 specification 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 Specification 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.

## Design Approach and Rationale

The specification defined in this document 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:

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 specification 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 specification 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 specification 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 document.
3. **Modularity**: Those familiar with the CDS Knowledge Artifact Specification 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.
5. **Ease of Use:** The structure of this specification 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 Specification has some similarity to existing approaches that have been used to develop balloted specifications and 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 Specification is an implementation specification structured as a series of layered constraints for each of the components used in the CDS Knowledge Artifact. This specification 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 specification at a high level to support initial piloting and adoption of the CDS Knowledge Artifact.

### 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 specification 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 specification, 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.

### Lifecycle of a CDS knowledge artifact

The CDS Knowledge Artifact is the structured and encoded format that this specification 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**).

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.

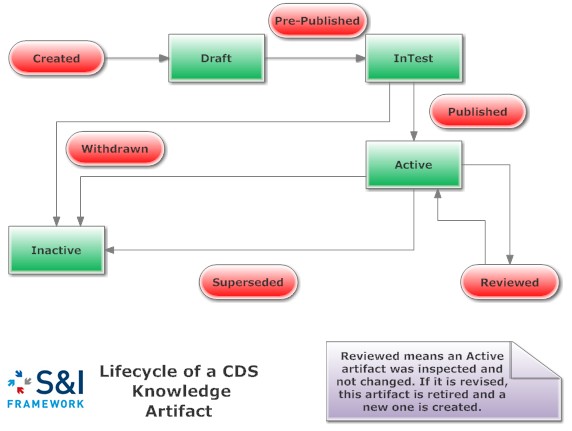


Figure 1 - CDS Knowledge Artifact Lifecycle

### 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.

### 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.

#### 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.

#### 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:



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.

## Audience

The audience of this specification 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 specification also specifically covers consumers and integrators of the CDS Knowledge Artifact.

### Requisite Knowledge

This section includes pre-requisites for implementers and other users of the CDS Knowledge Artifact Specification. Specific prerequisites for CDS implementations using the CDS Knowledge Artifact Specification 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 specification, 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 in this specification 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.

## Scope of the Specification

As defined in HeD Use Case 1, the scope of this specification 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 specification.

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.

### 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 Specification | Specification | 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 |
| 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   1. 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

## Alignment to HeD Artifact Sharing Use Case

The specific requirements implemented within this specification 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 specification, a thorough understanding of the use case is expected for implementation.

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>

### 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.

### 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.

#### System Requirements

This specification 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 specification include the following:

|  |  |
| --- | --- |
| **System Requirement** | **Description** |
| Provides CDS Knowledge Artifact in Structured Format | The specification 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 specification 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 SPECIFICATION

## Organization of this Specification

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

### 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 specification. In addition, the reader should be familiar with the terms defined in Appendix A of the HeD Artifact Sharing Use Case – wherever possible, this specification 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. |
| 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 specification 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 specification 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 specification 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. |
| 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 SPECIFICATION

### Definitions of Actors

This specification 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 Specification 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 specification 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 |

Table 5 – Roles and Actors Supported in this Specification

### Conventions Used

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

#### 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 specification 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

#### Use of Versioning

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

#### 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***](http://wiki.hl7.org/index.php?title=Conformance_Implementation_Manual) for further details

Working code examples are also provided in this specification to assist in understanding the CDS Knowledge Artifact schema. While this specification 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.

#### Use of Conformance Language

Conformance language is defined within this specification 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 specification using **BOLD** **CAPS** to denote the conformance criteria to be applied. The conformance language that is used in this specification is drawn from RFC 2219, and the conformance matrix offered for use in this specification 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 specification, and also lists specific recommendations used in this specification:

| **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 | - | - |
| 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 - Specification- Conformance Verb Matrix

The use of the word “recommendation” is also used throughout this specification, 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. |

Table 8 - Data Element - Optionality Levels

### 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 specification, 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 specification. The location of this repository is:

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

Section 2 CDS Knowledge Artifact Types of this specification covers the different knowledge artifact types supported by the CDS Knowledge Artifact schema. Section 3 of this specification 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.

# CDS Knowledge Artifact Types

This specification 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.

## 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.**

## 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, a documentation template, or a library. 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.

### 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 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.

### 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.

### 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).

### 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.

### 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).

### 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.

### 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 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.

## Event Condition Action (ECA) Rules

### 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:

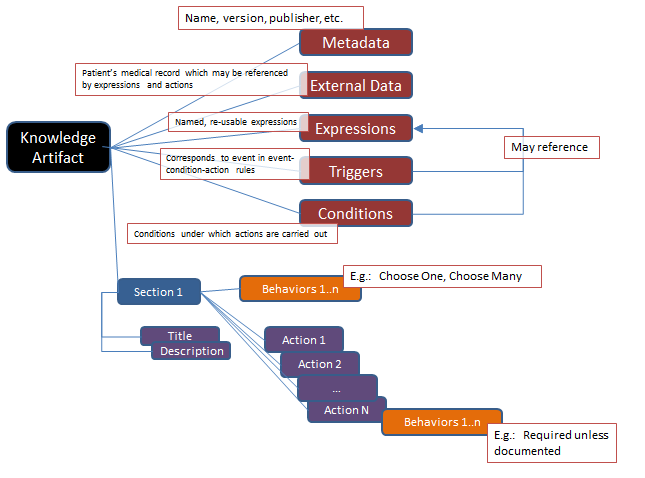


Figure 2 - ECA Rule – Conceptual Overview

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 Actions of the specification).

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

1. Start of encounter in primary care clinic

Conditions

1. Patient is an adult, and
2. Patient has diabetes, and
3. Patient has not had a Hemoglobin A1C test in the last three months, and
4. Patient had an abnormal Hemoglobin A1C test sometime in the nine-month period before the last three months

Actions

1. Notify the provider
2. Order Hemoglobin A1C in the next three days

### 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.

#### 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">

<sch:assert test="./@value='Rule'">

The value attribute of artifact type must be 'Rule'

</sch:assert>

</sch:rule>

</sch:pattern>

#### External Data

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

#### Expressions

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

#### 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.

#### 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>

#### Behaviors

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

#### ActionGroups

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

## Order Sets

### 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.

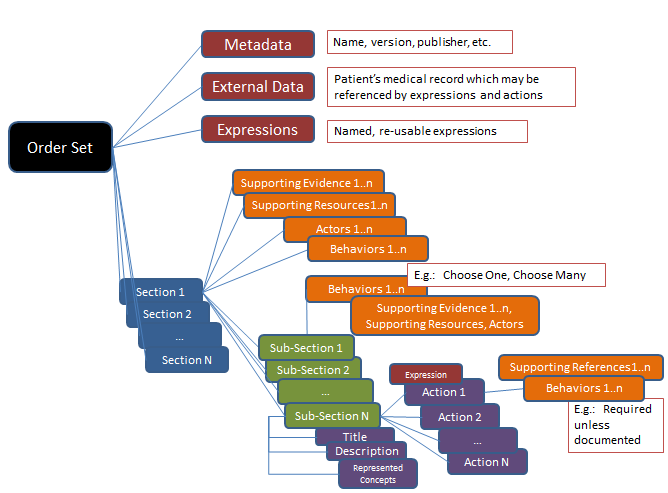


Figure 3 - Order Set – Conceptual Overview

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.

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

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.

### 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.

#### 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>

#### 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.

#### 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.

#### Triggers

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

#### 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.

#### 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.

#### 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.

## Documentation Templates

### 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 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.

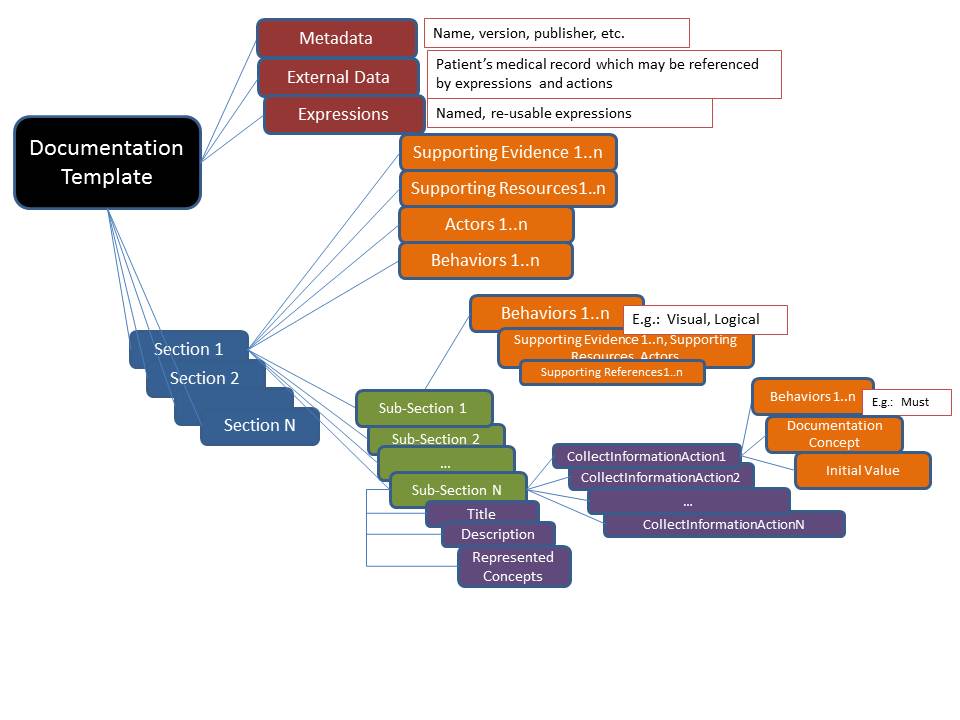


Figure 4 - Documentation Template - Conceptual Overview

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

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)

### 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.

#### 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>

#### 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.

#### 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.

#### Triggers

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

#### 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.

#### 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.

#### 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>

</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>

## Libraries

### Conceptual Overview

Libraries provide a mechanism for artifacts to share common components, as well as the potential to reference components defined in non-HeD formats. For example, an artifact may reference an expression defined in an external library whose format uses Java as the expression language. This specification only defines libraries that use the HeD Schema; non-HeD format libraries would need to be defined in a separate specification.

The Library artifact type uses the basic components of the HeD schema to specify the reusable components of the library. The Metadata component is used to provide library information such as the title, author, and lifecycle information.

The External Data and Expressions components are used to allow for the definition of reusable expression logic.

The Action Groups element is used to allow for the definition of reusable actions within the library.

Artifacts can reference library definitions using the LibraryReference element of the Metadata component, and specific expressions or actions within the library can be referenced by the artifact using the libraryName attribute of the ExpressionRef and ActionRef types.

### Library Conformance Requirements

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

#### Metadata

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

<sch:pattern name="LIB-1: Artifact type is Library">

<sch:rule context="/hed:knowledgeDocument/hed:metadata/hed:artifactType">

<sch:assert test="./@value='Library'">

The value attribute of artifact type must be 'Library'

</sch:assert>

</sch:rule>

</sch:pattern>

#### External Data

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

#### Expressions

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

#### Triggers

Libraries may not contain trigger definitions.

#### Conditions

Libraries may not contain condition definitions.

#### Behaviors

Libraries may not contain behaviors.

#### ActionGroups

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

## 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](https://docs.google.com/a/esacinc.com/document/d/1JdjRxR2rQRioBXXxxiw9jpl6ZP5c2pDEbdW8xq5xjEc/edit).

### 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.

### 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).

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.

# 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.

## 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>

### 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 document uses the vMR exclusively to represent clinical data. This does not imply that the vMR is part of the Knowledge Artifact specification; the specification allows for any data model to be used.

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"

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.

## 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.

### 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

#### 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.

#### 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 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](#InitialValue_FLACC) 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](#ResponseBinding_FLACC) is shown in Appendix E, FLACC Example. An example of a [response container declaration](#DeclareResponseAction_FLACC) 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.

#### 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 .

### 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](#ObjectExpression_RespiratoryOrder) 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](#ComplexLiteral_Dopamine) is shown in Appendix E, DopamineComplexIVOrderWithComplexLiteral.

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.

### 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](#GroupReference_HeartFailure) 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.

### 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](#ApplicableScenario_FLACC) is shown in Appendix E, HeartFailureAdmissionToMedSurgOrderSet.

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>

### 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

@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](#GroupSelectionBehavior_RespiratoryOrder) is shown in Appendix E, RespiratoryOrder. An [example of GroupOrganizationBehavior](#GroupOrganizationBehavior_HeartFailure) 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>

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

</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='GroupOrganizationBehavior' 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='GroupOrganizationBehavior' 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>

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](#RequiredBehavior_HeartFailure) 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](#PrecheckBehavior_HeartFailure) 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>

## 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 specification. See Section 3.5 Extending the Schema on how the schema can be extended.

### 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.

### 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.

## 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.

### 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](#BKM_E73D63CA_6728_401E_8EBA_3C18DCFCD323) | Generic literal for scalar types. |
| [AddressLiteral](#BKM_71DD55BB_306B_406A_875F_C540FCF1150A) | Returns a value of type AD with the given attributes. |
| [BooleanLiteral](#BKM_C46048CE_393F_4323_9E0B_6F31177ED053) | Returns a value of type BL with the given attributes. |
| [CodeLiteral](#BKM_F7C3F80E_823F_45E1_99B3_FEEBD5422FC8) | Returns a value of type CD with the given attributes. |
| [CodedOrdinalLiteral](#BKM_01224A63_D6EE_4D16_83B5_967B29B950B3) | Returns a value of type CO with the given attributes. |
| [SimpleCodeLiteral](#BKM_69A6E2B5_1464_433D_AB57_A83EC6252291) | Returns a value of type CS with the given attributes. |
| [EntityNameLiteral](#BKM_E342EC5C_25C1_4588_830E_9E0CB728A6C1) | Returns a value of type EN with the given attributes. |
| [IdentifierLiteral](#BKM_DA490734_2082_401C_AA24_58C7AA0C34D8) | Returns a value of type II with the given attributes. |
| [IntegerLiteral](#BKM_D6204E22_D98D_4748_B18B_BAEDD59988E1) | Returns a value of type INT with the given attributes. |
| [IntegerIntervalLiteral](#BKM_067C8173_E0EC_4441_BD56_657F70D209C9) | Returns a value of type IVL\_INT with the given attributes. |
| [PeriodLiteral](#BKM_4E54F00F_EEF8_47B9_A8C2_5FD7CACDC2E8) | Returns a value of type PIVL\_TS with the given attributes. |
| [PhysicalQuantityLiteral](#BKM_80791764_71FD_448A_B93E_47A144DF6707) | Returns a value of type PQ with the given attributes. |
| [QuantityIntervalLiteral](#BKM_22C4F21E_B987_4324_94C6_33A4617E2D48) | Returns a value of type IVL\_QTY with the given attributes. |
| [RealIntervalLiteral](#BKM_8E528F72_6136_49E5_AA43_040872E33F9B) | Returns a value of type IVL\_REAL with the given attributes. |
| [RealLiteral](#BKM_EB436F76_7AEF_4FB8_B06F_B38506305D02) | Returns a value of type Real with the given attributes. |
| [StringLiteral](#BKM_5579F811_66A0_4585_8747_300771DAF9A1) | Returns a value of type ST with the given attributes. |
| [TimestampLiteral](#BKM_D1CA7683_B689_4A5F_A940_19D855DBBE2E) | Returns a value of type TS with the given attributes. |
| [TimestampIntervalLiteral](#BKM_8BB77361_810F_4073_8FBA_6F04D420ECA4) | Returns a value of type IVL\_TS with the given attributes. |
| [UrlLiteral](#BKM_722FAE02_1718_42E0_8029_2B5DC3CD218E) | Returns a value of type TEL with the given attributes. |

### 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"/>

</when>

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

|  |  |
| --- | --- |
| Expression | Description |
| [Equal](#BKM_DA5C9DBC_E049_4265_AB72_E53A878C8069) | Returns true if the operands are equal. |
| [NotEqual](#BKM_E46549BB_389B_40DA_8AB1_B091E6453B03) | Returns true if the operands are not equal. |
| [Less](#BKM_F13DFE14_55FB_48EB_9324_0601B5325473) | Returns true if the first operand is less than the second operand. |
| [LessOrEqual](#BKM_960864D6_6152_4511_A37D_02BBDC7B9465) | Returns true if the first operand is less than or equal to the second operand. |
| [Greater](#BKM_5933E598_AEC9_4D1D_9D31_9CD6E312D8B7) | Returns true if the first operand is greater than the second operand. |
| [GreaterOrEqual](#BKM_265D7BF2_82B0_440A_BB93_8E09DE5AFED4) | Returns true if the first operand is greater than or equal to the second operand. |

### 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:

|  |  |
| --- | --- |
| Expression | Description |
| [And](#BKM_FCD74973_96F1_4948_A2C0_6FC4FD9DF374) | Returns the logical conjunction of its operands. |
| [Or](#BKM_2F2A81E5_66D4_4600_9805_AD5ADA02D213) | Returns the logical disjunction of its operands. |
| [Not](#BKM_1E24597A_3DCB_4237_B231_484CC39770F8) | Returns the logical negation of its operand. |

### 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](#BKM_9BEEA5A2_A6B9_4D0E_9428_FA48B493C2C5) | Returns a typed null. |
| [IsNull](#BKM_2CBC360B_A1CF_467C_BBB6_2CF3B4FDFA19) | Returns true if the argument is *null*, false otherwise. |
| [IfNull](#BKM_3BEA3DC3_E10D_4C2C_A733_40C4D3ECDD57) | Returns the first argument if it is not null, otherwise, returns the second argument. |
| [Coalesce](#BKM_C1815CAF_F81A_4D3E_B70A_C1D3DD76BD2C) | Returns the first non-null argument, null if there are no non-null arguments. |

### 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>

<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](#BKM_0A375361_644D_4F35_B3FC_022C2C4E521A) | Allows for conditional evaluation between two expressions. |
| [Case](#BKM_4883005A_DDB7_42F2_9C76_573C5A3807D5) | Allows for multiple conditional expressions, or a comparand with multiple cases. |

### 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](#BKM_234A6077_7830_4D9C_B2A1_9557BF170FBC) | Performs numeric addition of its arguments. |
| [Subtract](#BKM_14C4033C_FAE7_4B78_BA54_2BB1067D3859) | Performs numeric subtraction of its arguments. |
| [Multiply](#BKM_F7A3108A_57F8_4FB7_9748_647EDC97EA10) | Performs numeric multiplication of its arguments. |
| [Divide](#BKM_E7BA236F_5AFB_4946_9DCC_37778C379960) | Performs numeric division of its arguments. |
| [TruncatedDivide](#BKM_3E503B83_9A73_416A_80A2_750EA36A98EF) | Performs integer division of its arguments. |
| [Modulo](#BKM_6C462234_4977_48F9_B1A3_581BBE8E464A) | Computes the remainder of the division of its arguments. |
| [Ceiling](#BKM_02636B8C_DBA5_4EF5_A456_D1092739A349) | Returns the first integer greater than or equal to its argument. |
| [Floor](#BKM_F5EA2E27_2B23_4B8D_A3B0_E377850FC17E) | Returns the first integer less than or equal to its argument. |
| [Truncate](#BKM_4BE3E765_BC81_400A_8D88_FEE72EAB266F) | Returns the integer component of its argument. |
| [Abs](#BKM_02863D5F_D95E_45C6_A06C_E98DD914E531) | Returns the absolute value of its argument. |
| [Negate](#BKM_5C78AD9A_DB9F_4EB1_95B6_86978931DAD2) | Returns the negative value of its argument. |
| [Round](#BKM_187AE3C3_4A29_45AD_8470_ACEE50BC8ABE) | Returns the nearest numeric value to its argument, optionally specified to a number of decimal places for rounding. |
| [Ln](#BKM_20D1EF12_9117_45BF_A94D_50C2211D093A) | Computes the natural logarithm of its argument. |
| [Log](#BKM_086B3A4D_5D11_4D86_B0A9_87207293DDD5) | Computes the logarithm of its first argument, using the second argument as the base. |
| [Power](#BKM_DC627D6B_6D2E_42DE_9BC5_D580941412FB) | Raises the first argument to the power given by the second argument. |
| [Succ](#BKM_37E85599_95E5_4F66_AE2E_186937F02875) | Returns the successor of its argument. |
| [Pred](#BKM_56E4C1B4_452E_4085_8729_C026406231EA) | Returns the predecessor of its argument. |
| [MinValue](#BKM_7568C285_C0E4_4D4D_A9CF_0D881B9C73FA) | Returns the minimum representable value for a type. |
| [MaxValue](#BKM_9DBCC977_6C0D_4B81_A365_E91FAC1E7099) | Returns the maximum representable value for a type. |

### 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](#BKM_CD61AE31_D93F_4E76_869C_3BDFAFA820F1) | Returns the concatenation of its arguments. |
| [Combine](#BKM_7FB5A435_CE7A_40DB_88B8_3AB14BD122E8) | Combines a list of strings, optionally separating them with the given separator. |
| [Split](#BKM_A68DC2B8_6074_426A_B5F9_4FBD57D9B05F) | Splits a string into a list of strings along a given separator. |
| [Length](#BKM_815F85CF_7BE0_44AA_816B_18E0F902A1FB) | Returns the length of its argument. |
| [Upper](#BKM_61DFB85D_474B_48B8_BA9A_786EB7CFA72D) | Returns the upper case representation of its argument. |
| [Lower](#BKM_A861850A_F587_4988_90DD_94B23AF3FE0F) | Returns the lower case representation of its argument. |
| [Indexer](#BKM_49BC2101_9845_4077_B9E9_F1D1C18207BA) | Returns the nth element of its argument. |
| [Pos](#BKM_78E274DD_46E0_4E64_8C9A_209BEB2B0B42) | Returns the starting position of a given pattern within a string. |
| [Substring](#BKM_81B0B1CD_31C1_434E_9EE1_0A7149AA2DE5) | Returns a substring of its argument. |

### 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](#BKM_9B40827C_11F6_4832_B8E1_1A594BF2BCD5) | Adds a given number of periods to its argument. |
| [DateDiff](#BKM_D61648C1_5013_421C_B98F_901E3EDFB8B2) | Computes the number of periods between a starting and ending date. |
| [DatePart](#BKM_ECAE7A28_0BA1_4A4A_8EDB_83941FA09CC7) | Returns a specified component of its argument. |
| [Today](#BKM_5CC5E486_D2DE_4564_94C4_77F6937D1B1D) | Returns the date (with no time component) of the start timestamp associated with the evaluation request. |
| [Now](#BKM_413ECB2D_C9B7_4714_9FB7_2B43917652F6) | Returns the date and time of the start timestamp associated with the evaluation request. |
| [Date](#BKM_33073B0E_B100_4024_B89C_F8736C8DF62F) | Constructs a date from its arguments. |
| [DateOf](#BKM_000312CA_BBB2_4BB9_A238_68882CC5A5FE) | Returns the date (with no time component) of the argument. |
| [TimeOf](#BKM_548136DA_1D0B_491D_830C_5C2D45BB94EF) | Returns the time (with the date set to the minimum representable date) of the argument. |

### 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"/>

</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>

</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"/>

<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">

<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"/>

<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 -->

<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>

</operand>

</expression>

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

|  |  |
| --- | --- |
| Expression | Description |
| [List](#BKM_ED9F5067_CC22_414B_BC08_C8D27EA9A6B7) | Constructs a list from its arguments. |
| [IsEmpty](#BKM_74344F4C_66EA_4DAF_B227_C3AB33DDE4C8) | Returns true if its argument contains no elements. |
| [IsNotEmpty](#BKM_75F458CF_98CE_40D7_B605_3075E4A95EEB) | Returns true if its argument contains any elements. |
| [Equal](#BKM_DA5C9DBC_E049_4265_AB72_E53A878C8069) | Returns true if its arguments have the same number of elements, and for each element considered in order, the elements are equal. |
| [NotEqual](#BKM_E46549BB_389B_40DA_8AB1_B091E6453B03) | Returns true if its arguments are not equal. |
| [Union](#BKM_05887727_1893_437B_85B0_59560BD8CD3F) | Returns a list containing all the elements of its arguments. |
| [Difference](#BKM_026F1EDD_58A8_4B55_8F45_A5EAE16C2E04) | Returns a list containing only the elements in the first list that are not in the second list. |
| [Intersect](#BKM_1F430774_696C_4EA9_860F_2801B3B3CA76) | Returns a list containing only the elements that are in all of its arguments. |
| [Filter](#BKM_4791A5DD_75BE_47F0_9A45_79F41D986BB0) | Returns a list containing only the elements for which the given condition evaluates to true. |
| [IndexOf](#BKM_9266DAA6_3430_4CA6_AB03_BD4398961ED4) | Returns the 1-based index of an element within the list, or 0 if the element is not present. |
| [Indexer](#BKM_49BC2101_9845_4077_B9E9_F1D1C18207BA) | Returns the element at the given 1-based index in the list. |
| [In](#BKM_30460EA4_6E0F_41E1_96C2_98EFE70F88E0) | Returns true if the given element is in a given list. |
| [Contains](#BKM_AB00873E_E971_468C_B7C1_8F9DB1F7CD63) | Returns true if the given list contains a given element. |
| [Includes](#BKM_9A767CBD_BADD_4FB5_AEE4_F2DD2F23B61F) | Returns true if every element in the second list is in the first list. |
| [IncludedIn](#BKM_7593A786_0747_4D42_BF35_3B6BB1DA7315) | Returns true if every element in the first list is in the second list. |
| [ProperIncludes](#BKM_B757B342_9D28_4073_9522_BBBBFF5CF6C2) | Returns true if every element in the second list is in the first list, and the first list is strictly larger than the second. |
| [ProperIncludedIn](#BKM_4A183DAC_C7F9_474C_BD2D_20F7B1E35AF3) | Returns true if the second list contains every element in the first list, and the second list is strictly larger than the first. |
| [Sort](#BKM_553C391E_E461_45A3_86EE_776FB608B6C1) | Returns a list with the same elements, sorted by the given sort criteria. |
| [ForEach](#BKM_D4B77B9C_63F7_4C68_BDDA_20B441DD1AFB) | Returns a list whose elements are determined by evaluated an expression for each element in its argument. |
| [Expand](#BKM_D4ED1F66_9ADA_4808_B510_7C865DDC6515) | Flattens a list of lists into a single list with all the elements from every list in the input. Duplicates are not eliminated by this operation. |
| [Distinct](#BKM_13252CE5_5ED8_4111_9075_F9ED9C2A018B) | Returns a list that contains the unique elements within its argument. |
| [Current](#BKM_218ABED2_95B5_4765_A8E2_0CCADBF98821) | Returns the contents of the current scope. |

### 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:

<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](#BKM_FB1E3C58_87F8_4C77_96C3_DB60C31D34C0) | Returns the number of non-null elements in the source. |
| [Sum](#BKM_D41BFAFA_F529_4420_BBD7_3F1BAED84D24) | Computes the sum of non-null elements in the source. |
| [Min](#BKM_6F0C0C73_7D29_43FD_9AFC_3A01C3A6565D) | Returns the minimum element in the source. |
| [Max](#BKM_A5ABC1E1_7BE0_4E3D_934E_30BFDEF38C3C) | Returns the max element in the source. |
| [Avg](#BKM_0593B2E8_30BB_4B1E_B683_F38C815251FB) | Returns the average of the elements in the source. |
| [Median](#BKM_005DC0D4_45F0_40B6_9D73_0991A9DBECDB) | Returns the median of the elements in the source. |
| [Mode](#BKM_9015D7C1_F2DF_4FB1_9EC8_784E500F4D45) | Returns the mode of the elements in the source. |
| [Variance](#BKM_58591B47_77AA_478C_BFF1_427E9A09E080) | Returns the statistical variance of the elements in the source. |
| [PopulationVariance](#BKM_9404FC95_2F04_43D1_A4ED_363AAF0A0DA6) | Returns the population variance of the elements in the source. |
| [StdDev](#BKM_6D1B7366_3132_4B5A_B7D3_72A68314F222) | Returns the standard deviation of the elements in the source. |
| [PopulationStdDev](#BKM_A5DE0F63_9BE3_402F_9D32_4A1FECBAAB91) | Returns the population standard deviation of the elements in the source. |
| [AllTrue](#BKM_145E6A36_4F8F_4CF3_96A0_5F2F1E460398) | Returns true if all the elements in source are true. |
| [AnyTrue](#BKM_698C8DAB_0BA4_45DD_8E8D_6BD2EC19B442) | Returns true if any element in source is true. |

### 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>

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>

<!-- 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](#BKM_DA5C9DBC_E049_4265_AB72_E53A878C8069) | Returns true if the arguments are the same interval. |
| [NotEqual](#BKM_E46549BB_389B_40DA_8AB1_B091E6453B03) | Returns true if the arguments are not the same interval. |
| [Contains](#BKM_AB00873E_E971_468C_B7C1_8F9DB1F7CD63) | Returns true if the interval contains the given point. |
| [In](#BKM_30460EA4_6E0F_41E1_96C2_98EFE70F88E0) | Returns true if the given point is in the interval. |
| [Includes](#BKM_9A767CBD_BADD_4FB5_AEE4_F2DD2F23B61F) | Returns true if the first interval completely includes the second (i.e. starts on or before and ends on or after). |
| [IncludedIn](#BKM_7593A786_0747_4D42_BF35_3B6BB1DA7315) | Returns true if the first interval is completely included in the second (i.e. starts on or after and ends on or before). |
| [ProperIncludes](#BKM_B757B342_9D28_4073_9522_BBBBFF5CF6C2) | Returns true if the first interval completely includes the second and the first interval is strictly larger (i.e. starts before and ends after). |
| [ProperIncludedIn](#BKM_4A183DAC_C7F9_474C_BD2D_20F7B1E35AF3) | Returns true if the first interval is completely included in the second and the second interval is strictly larger (i.e. starts after and ends before). |
| [Before](#BKM_E2A392E6_254E_4420_BDDF_9473B0B683A9) | Returns true if the first interval ends before the second one starts. |
| [After](#BKM_FAE7134E_432E_429B_8DFC_5135E5A33C10) | Returns true if the first interval starts after the second one ends. |
| [Meets](#BKM_83A78536_F177_49CB_892A_5FBD00131B50) | Returns true if the first interval ends immediately before the second interval starts, or if the first interval starts immediately after the second interval ends. |
| [Overlaps](#BKM_2ECCAF6D_AC09_4306_B706_C946FE488A57) | Returns true if the first interval overlaps the second. |
| [OverlapsBefore](#BKM_A94E0F12_4D61_4B22_8325_708731E68A7F) | Returns true if the first interval starts before and overlaps the second. |
| [OverlapsAfter](#BKM_0DE77813_6050_4927_A54B_C87A2BBA5B36) | Returns true if the first interval ends after and overlaps the second. |
| [Union](#BKM_05887727_1893_437B_85B0_59560BD8CD3F) | Returns the interval that results from combining the arguments. |
| [Intersect](#BKM_1F430774_696C_4EA9_860F_2801B3B3CA76) | Returns the interval that results from the intersection of the arguments. |
| [Difference](#BKM_026F1EDD_58A8_4B55_8F45_A5EAE16C2E04) | Returns the interval that results from subtracting the second interval from the first. |
| [Length](#BKM_815F85CF_7BE0_44AA_816B_18E0F902A1FB) | Returns the length of the interval. |
| [Begin](#BKM_2116E945_B734_41C0_A0F8_8688A2D8D4C1) | Returns the starting point of the interval. |
| [End](#BKM_AC8B9E5C_0FB1_4325_8632_D1798F9CA4BB) | Returns the ending point of the interval. |
| [Begins](#BKM_AFD192CE_BCD7_4FDA_9CAD_7E91920A1D72) | Returns true if the first interval begins the second. |
| [Ends](#BKM_94DBEC89_EAE7_44FC_BA93_A21FFBDD0DDE) | Returns true if the first interval ends the second. |

### 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">

<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](#BKM_A90DB4A3_3FE1_42AE_9A90_DEA9033855CC) | Constructs a new structured value. |
| [ObjectRedefine](#BKM_D9AB6E58_5722_4E04_ADE5_BD576CE71557) | Constructs a new structured value based on the values of an existing structured value, with specific properties given new values in the result. |
| [Property](#BKM_591A024B_213B_47F2_A65C_7D6FB172B2F9) | Returns the value of a specific property of a structured value. |

### 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">

<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](#BKM_232D8D7C_F86E_404C_9C96_0C882B68814F) | Defines a named expression that can be referenced by other expressions. |
| [ExpressionRef](#BKM_0B67F2C5_4343_4292_8FFF_72ABFAEF7E01) | Returns the result of evaluating a named expression. |

### 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 specification, 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 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>

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](#BKM_0E478F6C_1652_4423_9084_23946622640B) | Defines clinical data that will be used within the artifact. |
| [ValueSet](#BKM_D24AEDBD_BB87_44D8_BB2D_716BA21D2835) | Returns the list of codes for a value set. |
| [InValueSet](#BKM_288D6A37_C5AC_489F_A124_F52607112ABE) | Tests a code for membership in a value set. |
| [Subsumes](#BKM_6A4C7C08_5571_4324_812F_F9C1992E36B4) | Tests two codes for subsumption (i.e. whether the concept represented by one code is subsumed by the concept represented by another). |
| [SetSubsumes](#BKM_845EFC79_531B_4DC0_A389_33517EA185C9) | Computes the intersection of two sets of codes using the subsumption relationship. |

### 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>

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

|  |  |
| --- | --- |
| Expression | Description |
| [ParameterDef](#BKM_A32FEEC3_6F03_48D3_AB9D_125D96723982) | Defines a parameter to the artifact. |
| [ParameterRef](#BKM_F115D230_D9F7_4BFA_B065_ECC08634DE1C) | Returns the value of a parameter. |

## 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.

### 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.

### Other Extensions and Modifications

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

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

# 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.

## 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.

### 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.

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.

### 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:

#### 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.

#### 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.

### 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. |

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.

## 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.

### 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 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.

#### 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.

#### 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.

#### 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.

#### 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.

#### 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.

### 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.

### 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:

|  |  |
| --- | --- |
| 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.

### 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.

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 |
| 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.

# Schema References

## Datatypes

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.

**Datatypes** - *(Class diagram)*



Figure: 1

### AD

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AD](#BKM_60A89184_1F5F_456B_A6FE_72E8AED9BA8A) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **part** ADXP  [1..\*] | A sequence of address parts, such as street or post office Box, city, postal code, country, etc. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |
| **use** set\_PostalAddressUse  [1] | 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. | *Default:*    [use = optional ] |

### ADXP

Type: **Class** Derived From: XP

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).

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ADXP](#BKM_E4E09526_0C5B_4875_8B56_8B9F535B3248) | [XP](#BKM_0D52E664_9AB2_4823_8AB2_82F5EE0F956C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **type** AddressPartType  [1] | 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. | *Default:*    [use = required ] |

### ANY

Type: **Class** Derived From: HXIT

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AD](#BKM_60A89184_1F5F_456B_A6FE_72E8AED9BA8A) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) | [HXIT](#BKM_E70E6C97_3BCC_4888_B020_0E68E2D938C5) |  |
| [BL](#BKM_432ADA26_08C5_44C2_B08F_AA97294C4445) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [CD](#BKM_984C8021_B33C_4982_ABE2_61E5194B28D2) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [CS](#BKM_754D9FDE_1FAE_474A_BEE5_087DD56A2073) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [ED](#BKM_60841C9F_8C61_4692_A276_009DF2DAC2DB) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [EN](#BKM_580F950D_FE76_4A20_8F76_C34F9E6D1C2A) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [II](#BKM_51542036_E725_4E17_976D_D340C608641C) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [QSET](#BKM_A14F30A0_57D7_4860_8B12_A999DFF1771B) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [ST](#BKM_0F09BA88_A403_493F_BB5E_57CB519DF2EA) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [TEL](#BKM_D5047ACE_5BF5_434E_984C_FBCF241A8B15) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

### BL

Type: **Class** Derived From: ANY

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

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [BL](#BKM_432ADA26_08C5_44C2_B08F_AA97294C4445) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** boolean  [1] | The value of the BL. | *Default:*    [use = required ] |

### CD

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CD](#BKM_984C8021_B33C_4982_ABE2_61E5194B28D2) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** Code  [1] | 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. | *Default:*    [use = optional ] |
| **codeSystem** Uid  [1] | 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. | *Default:*    [use = optional ] |
| **codeSystemName** string  [1] | 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. | *Default:*    [use = optional ] |
| **codeSystemVersion** string  [1] | If applicable, a version descriptor defined specifically for the given code system. | *Default:*    [use = optional ] |
| **displayName** ST  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **originalText** ST  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **translation** CD  [0..\*] | Translation of the base code / codeSystem to other codeSystems. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |
| **valueSet** Uid  [1] | The value set that applied when this CD was created. | *Default:*    [use = optional ] |
| **valueSetVersion** string  [1] | The version of the valueSet in which the code was found. | *Default:*    [use = optional ] |

### CO

Type: **Class** Derived From: QTY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CO](#BKM_E8E08D30_96FF_48AF_A4BF_4FF15B47D32D) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** CD  [0..1] | A code representing the definition of the ordinal item | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **value** Decimal  [1] | 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. | *Default:*    [use = optional ] |

### CS

Type: **Class** Derived From: ANY

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

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CS](#BKM_754D9FDE_1FAE_474A_BEE5_087DD56A2073) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** string  [1] | 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. | *Default:*    [use = required ] |

### Code

Type: **Class** Derived From: string

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.

### Decimal

Type: **Class** Derived From: double

A number that is not restricted to an integer, and may contain fractional values between two integers.

### ED

Type: **Class** Derived From: ANY

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

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ED](#BKM_60841C9F_8C61_4692_A276_009DF2DAC2DB) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **charset** Code  [1] | An Internet Assigned Numbers Authority (IANA) Charset Registered character set and character encoding for character-based encoding types&#60;b&#62;. &#60;/b&#62;  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 | *Default:*    [use = optional ] |
| **compression** Compression  [1] | 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. | *Default:*    [use = optional ] |
| **data** base64Binary  [0..1] | A simple sequence of byte values that contains the content. (Base64 Encoded String). | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **description** ST  [0..1] | 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 &#34;translation&#34; property.  The intent of this property is to allow compliance with disability requirements such as those expressed in American&#39;s with Disability Act (also known as &#34;Section 508&#34;), 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **integrityCheck** base64Binary  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **integrityCheckAlgorithm** IntegrityCheckAlgorithm  [1] | The algorithm used to compute the integrityCheck value.  If populated, the value of this attribute SHALL be taken from the HL7 IntegrityCheckAlgorithm code system. | *Default:*    [use = optional ] |
| **language** Code  [1] | 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. | *Default:*    [use = optional ] |
| **mediaType** Code  [1] | 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 &#60;i&#62;mediaType&#60;/i&#62; 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. | *Default:*    [default = text/plain ]  [use = optional ] |
| **reference** TEL  [0..1] | 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 | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **value** string  [1] | 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 | *Default:*    [use = optional ] |
| **xml** anyType  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### EN

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [EN](#BKM_580F950D_FE76_4A20_8F76_C34F9E6D1C2A) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **part** ENXP  [1..\*] | A sequence of name parts, such as given name or family name, prefix, suffix, etc. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |
| **use** set\_EntityNameUse  [1] | 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. | *Default:*    [use = optional ] |

### ENXP

Type: **Class** Derived From: XP

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.).

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ENXP](#BKM_85E29798_F71E_4C5E_AF7C_67246C4D8856) | [XP](#BKM_0D52E664_9AB2_4823_8AB2_82F5EE0F956C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **qualifier** set\_EntityNamePartQualifier  [1] | 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. | *Default:*    [use = optional ] |
| **type** EntityNamePartType  [1] | 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. | *Default:*    [use = required ] |

### HXIT

Type: **Class** Derived From:

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) | [HXIT](#BKM_E70E6C97_3BCC_4888_B020_0E68E2D938C5) |  |

### II

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [II](#BKM_51542036_E725_4E17_976D_D340C608641C) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [VersionedIdentifier](#BKM_82E8B711_839C_4D5A_8AA8_C77FE391EA4A) | [II](#BKM_51542036_E725_4E17_976D_D340C608641C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **extension** string  [1] | 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. | *Default:*    [use = optional ] |
| **identifierName** string  [1] | A human readable description for this identifier. | *Default:*    [use = optional ] |
| **root** Uid  [1] | 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. | *Default:*    [use = required ] |

### INT

Type: **Class** Derived From: QTY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [INT](#BKM_0098CDC3_5B53_45F0_A6B7_7D6AF4CD7E10) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** integer  [1] | 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. | *Default:*    [use = required ] |

### IVL

Type: **Class** Derived From: QSET

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) | [QSET](#BKM_A14F30A0_57D7_4860_8B12_A999DFF1771B) |  |
| [IVL\_CO](#BKM_513C2473_CA80_44D1_B854_1F36AC50558E) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |
| [IVL\_INT](#BKM_1B8168D3_86BE_43C2_BFD1_E8EA38654FE7) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |
| [IVL\_PQ](#BKM_B424FD87_017D_4390_932B_7D208AF19CCB) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |
| [IVL\_QTY](#BKM_BD60D7D1_97F8_42F1_88B3_1841B13573C3) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |
| [IVL\_REAL](#BKM_5E240903_D2B5_4186_9487_F63827877B67) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |
| [IVL\_TS](#BKM_EFE5E52B_4934_46B6_953B_17B409666242) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

### IVL\_CO

Type: **Class** Derived From: IVL

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

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL\_CO](#BKM_513C2473_CA80_44D1_B854_1F36AC50558E) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** CO  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |
| **low** CO  [0..1] | This is the low limit. If the low limit is not known, it may be null.  The low limit SHALL NOT be positive infinity. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |

### IVL\_INT

Type: **Class** Derived From: IVL

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL\_INT](#BKM_1B8168D3_86BE_43C2_BFD1_E8EA38654FE7) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** INT  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |
| **low** INT  [0..1] | This is the low limit. If the low limit is not known, it may be null.  The low limit SHALL NOT be positive infinity. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |

### IVL\_PQ

Type: **Class** Derived From: IVL

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL\_PQ](#BKM_B424FD87_017D_4390_932B_7D208AF19CCB) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** PQ  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] | Whether high is included in the IVL (is closed) or excluded from the IVL (is open). | *Default:*    [use = optional ] |
| **low** PQ  [0..1] | This is the low limit. If the low limit is not known, it may be null.  The low limit SHALL NOT be positive infinity. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |

### IVL\_QTY

Type: **Class** Derived From: IVL

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL\_QTY](#BKM_BD60D7D1_97F8_42F1_88B3_1841B13573C3) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** QTY  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |
| **low** QTY  [0..1] | This is the low limit. If the low limit is not known, it may be null.  The low limit SHALL NOT be positive infinity. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |

### IVL\_REAL

Type: **Class** Derived From: IVL

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL\_REAL](#BKM_5E240903_D2B5_4186_9487_F63827877B67) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** REAL  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |
| **low** REAL  [0..1] | This is the low limit. If the low limit is not known, it may be null.  The low limit SHALL NOT be positive infinity. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |

### IVL\_TS

Type: **Class** Derived From: IVL

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL\_TS](#BKM_EFE5E52B_4934_46B6_953B_17B409666242) | [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** TS  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |
| **low** TS  [0..1] | This is the low limit. If the low limit is not known, it may be null.  The low limit SHALL NOT be positive infinity. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] | 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). | *Default:*    [use = optional ] |

### PIVL\_TS

Type: **Class** Derived From: QTY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PIVL\_TS](#BKM_C2ACBC71_E17C_450C_A465_D8B326525BF6) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **alignment** CalendarCycle  [1] |  | *Default:*    [use = optional ] |
| **count** INT  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **frequency** RTO  [0..1] | 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). | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **isFlexible** boolean  [1] | 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". | *Default:*    [use = optional ] |
| **period** PQ  [0..1] | A time duration specified as a reciprocal measure of the frequency at which the PIVL repeats. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **phase** IVL\_TS  [0..1] | 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 | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### PQ

Type: **Class** Derived From: QTY

A dimensioned quantity expressing the result of measuring.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PQ](#BKM_62FB5ACE_0000_49F6_82AB_9242F0043C0C) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **unit** Code  [1] | 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. | *Default:*    [use = required ] |
| **value** Decimal  [1] | The number which is multiplied by the unit to make the PQ. | *Default:*    [use = required ] |

### QSET

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IVL](#BKM_F343EB29_BED5_440A_95DA_DD403281D3CE) | [QSET](#BKM_A14F30A0_57D7_4860_8B12_A999DFF1771B) |  |
| [QSET](#BKM_A14F30A0_57D7_4860_8B12_A999DFF1771B) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

### QTY

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CO](#BKM_E8E08D30_96FF_48AF_A4BF_4FF15B47D32D) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |
| [INT](#BKM_0098CDC3_5B53_45F0_A6B7_7D6AF4CD7E10) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |
| [PIVL\_TS](#BKM_C2ACBC71_E17C_450C_A465_D8B326525BF6) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |
| [PQ](#BKM_62FB5ACE_0000_49F6_82AB_9242F0043C0C) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |
| [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |
| [REAL](#BKM_E2FE3505_047C_4C68_A28C_37E1E2BE5DE3) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |
| [RTO](#BKM_3BFB5359_6F70_4D50_A4B6_6B7F77E67392) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |
| [TS](#BKM_9E5BECCA_8C41_435F_87C2_42FE8E18E04D) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

### REAL

Type: **Class** Derived From: QTY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [REAL](#BKM_E2FE3505_047C_4C68_A28C_37E1E2BE5DE3) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** Decimal  [1] | The value of the REAL. | *Default:*    [use = required ] |

### RTO

Type: **Class** Derived From: QTY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RTO](#BKM_3BFB5359_6F70_4D50_A4B6_6B7F77E67392) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **denominator** QTY  [1] | The quantity that divides the numerator in the ratio.  The denominator SHALL not be zero. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **numerator** QTY  [1] | The quantity that is being divided in the ratio | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### ST

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ST](#BKM_0F09BA88_A403_493F_BB5E_57CB519DF2EA) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** string  [1] | The actual content of the string. | *Default:*    [use = required ] |

### TEL

Type: **Class** Derived From: ANY

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TEL](#BKM_D5047ACE_5BF5_434E_984C_FBCF241A8B15) | [ANY](#BKM_9DA7902A_F59D_4D46_9AC8_81F4C00CBE0C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **capabilities** set\_TelecommunicationCapability  [1] | 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 | *Default:*    [use = optional ] |
| **use** set\_TelecommunicationAddressUse  [1] | 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 | *Default:*    [use = optional ] |
| **value** anyURI  [1] | 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. | *Default:*    [use = required ] |

### TS

Type: **Class** Derived From: QTY

A quantity specifying a point on the axis of natural time. A point in time is most often represented as a calendar expression.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TS](#BKM_9E5BECCA_8C41_435F_87C2_42FE8E18E04D) | [QTY](#BKM_2DA49DC1_A8CD_4EF5_A92B_5A4447CA5E2E) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** TimeStamp  [1] | 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. | *Default:*    [use = required ] |

### TimeStamp

Type: **Class** Derived From: string

Represents a timestamp such as 20101127235417.123+0930

### Uid

Type: **Class** Derived From: string

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.

### Uri

Type: **Class** Derived From: string

Universal Resource Identifier

### XP

Type: **Class** Derived From:

A part of a name or address. Each part is a character string.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ADXP](#BKM_E4E09526_0C5B_4875_8B56_8B9F535B3248) | [XP](#BKM_0D52E664_9AB2_4823_8AB2_82F5EE0F956C) |  |
| [ENXP](#BKM_85E29798_F71E_4C5E_AF7C_67246C4D8856) | [XP](#BKM_0D52E664_9AB2_4823_8AB2_82F5EE0F956C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** string  [1] | The actual string value of the part. | *Default:*    [use = required ] |

### set\_EntityNamePartQualifier

Type: **Class** Derived From: dt:EntityNamePartQualifier

### set\_EntityNameUse

Type: **Class** Derived From: dt:EntityNameUse

### set\_PostalAddressUse

Type: **Class** Derived From: dt:PostalAddressUse

### set\_TelecommunicationAddressUse

Type: **Class** Derived From: dt:TelecommunicationAddressUse

### set\_TelecommunicationCapability

Type: **Class** Derived From: dt:TelecommunicationCapability

### AddressPartType

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **AL** string  [1] | 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. | *Default:* |
| **ADL** string  [1] | 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) | *Default:* |
| **UNID** string  [1] | Unit Identifier : The number or name of a specific unit contained within a building or complex, as assigned by that building or complex | *Default:* |
| **UNIT** string  [1] | Unit Designator: Indicates the type of specific unit contained within a building or complex. E.g. Apartment, Floor | *Default:* |
| **DAL** string  [1] | 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. | *Default:* |
| **DINST** string  [1] | 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. | *Default:* |
| **DINSTA** string  [1] | 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. | *Default:* |
| **DINSTQ** string  [1] | Delivery Installation Qualifier: A number, letter or name identifying a delivery installation. E.g., for Station A, the delivery installation qualifier would be 'A'. | *Default:* |
| **DMOD** string  [1] | Delivery Mode: Indicates the type of service offered, method of delivery. For example: post office box, rural route, general delivery, etc. | *Default:* |
| **DMODID** string  [1] | 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). | *Default:* |
| **SAL** string  [1] | 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. | *Default:* |
| **BNR** string  [1] | 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. | *Default:* |
| **BNN** string  [1] | Building Number Numeric: The numeric portion of a building number | *Default:* |
| **BNS** string  [1] | Building Number Suffix: Any alphabetic character, fraction or other text that may appear after the numeric portion of a building number | *Default:* |
| **STR** string  [1] | Street Name: The name of the street, including the type | *Default:* |
| **STB** string  [1] | Street Name Base: The base name of a roadway or artery recognized by a municipality (excluding street type and direction) | *Default:* |
| **STTYP** string  [1] | Street Type: The designation given to the street. (e.g. Street, Avenue, Crescent, etc.) | *Default:* |
| **DIR** string  [1] | Direction (e.g., N, S, W, E) | *Default:* |
| **INT** string  [1] | Intersection: An intersection denotes that the actual address is located at or close to the intersection of two or more streets | *Default:* |
| **CAR** string  [1] | 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 | *Default:* |
| **CEN** string  [1] | Census Tract: A geographic sub-unit delineated for demographic purposes. | *Default:* |
| **CNT** string  [1] | Country | *Default:* |
| **CPA** string  [1] | 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".) | *Default:* |
| **CTY** string  [1] | Municipality: The name of the city, town, village, or other community or delivery center | *Default:* |
| **DEL** string  [1] | Delimiter: Delimiters are printed without framing white space. If no value component is provided, the delimiter appears as a line break. | *Default:* |
| **POB** string  [1] | Post Box: A numbered box located in a post station. | *Default:* |
| **PRE** string  [1] | Precinct: A subsection of a municipality | *Default:* |
| **STA** string  [1] | State or Province: A sub-unit of a country with limited sovereignty in a federally organized country. | *Default:* |
| **ZIP** string  [1] | Postal Code: A postal code designating a region defined by the postal service. | *Default:* |
| **DPID** string  [1] | Delivery Point Identifier : A value that uniquely identifies the postal address. | *Default:* |

### CalendarCycle

Type: **Enumeration** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **CY** string  [1] | year | *Default:* |
| **MY** string  [1] | month of the year | *Default:* |
| **CM** string  [1] | month (continuous) | *Default:* |
| **CW** string  [1] | week (continuous) | *Default:* |
| **WM** string  [1] | week of the month | *Default:* |
| **WY** string  [1] | week of the year | *Default:* |
| **DM** string  [1] | day of the month | *Default:* |
| **CD** string  [1] | day (continuous) | *Default:* |
| **DY** string  [1] | day of the year | *Default:* |
| **DW** string  [1] | day of the week (begins with monday) | *Default:* |
| **HD** string  [1] | hour of the day | *Default:* |
| **CH** string  [1] | hour (continuous) | *Default:* |
| **NH** string  [1] | minute of the hour | *Default:* |
| **CN** string  [1] | minute (continuous) | *Default:* |
| **SN** string  [1] | second of the minute | *Default:* |
| **CS** string  [1] | second (continuous) | *Default:* |

### Compression

Type: **Enumeration** Derived From:

The compression algorithm, specified in the HL7 CompressionAlgorithm code system.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **DF** string  [1] | Deflate : The deflate compressed data format as specified in IETF RFC 1951. | *Default:* |
| **GZ** string  [1] | GZIP : A compressed data format that is compatible with the widely used GZIP utility as specified in IETF RFC 1952(uses the deflate algorithm). | *Default:* |
| **ZL** string  [1] | ZLIB : A compressed data format that also uses the deflate algorithm. Specified as IETF RFC 1950. | *Default:* |
| **Z** string  [1] | Compress : Original UNIX compress algorithm and file format using the LZC algorithm (a variant of LZW). Patent encumbered and less efficient than deflate. | *Default:* |
| **BZ** string  [1] | BZIP : bzip-2 compression format. See [http://www.bzip.org/] for more information. | *Default:* |
| **Z7** string  [1] | Z7 : 7z compression file format. See [http://www.7-zip.org/7z.html] for more information. | *Default:* |

### EntityNamePartQualifier

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **LS** string  [1] | Legal Status : For organizations a suffix indicating the legal status, e.g., "Inc.", "Co.", "AG", "GmbH", "B.V." "S.A.", "Ltd." Etc. | *Default:* |
| **AC** string  [1] | Academic : Indicates that a prefix like "Dr." or a suffix like "M.D." or "Ph.D." is an academic title | *Default:* |
| **NB** string  [1] | 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 | *Default:* |
| **PR** string  [1] | Professional : Primarily in the British Imperial culture people tend to have an abbreviation of their professional organization as part of their credential suffices | *Default:* |
| **HON** string  [1] | Honorific : A honorific such as "The Right Honourable" or "Weledelgeleerde Heer". | *Default:* |
| **BR** string  [1] | 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". | *Default:* |
| **AD** string  [1] | 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) | *Default:* |
| **SP** string  [1] | 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 | *Default:* |
| **MID** string  [1] | 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 | *Default:* |
| **CL** string  [1] | Callme : Callme is used to indicate which of the various name parts is used when interacting with the person | *Default:* |
| **IN** string  [1] | 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" | *Default:* |
| **PFX** string  [1] | 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). | *Default:* |
| **SFX** string  [1] | 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). | *Default:* |

### EntityNamePartType

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **FAM** string  [1] | 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 | *Default:* |
| **GIV** string  [1] | Given: Given name.  Note: don't call it "first name" since this given names do not always come first | *Default:* |
| **TITLE** string  [1] | 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 | *Default:* |
| **DEL** string  [1] | 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 | *Default:* |

### EntityNameUse

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **ABC** string  [1] | Alphabetic: Alphabetic transcription of name (Japanese: romaji) | *Default:* |
| **IDE** string  [1] | Ideographic : Ideographic representation of name (e.g., Japanese kanji, Chinese characters) | *Default:* |
| **SYL** string  [1] | Syllabic: Syllabic transcription of name (e.g., Japanese kana, Korean hangul) | *Default:* |
| **C** string  [1] | Customary : Known as/conventional/the one you normally use | *Default:* |
| **OR** string  [1] | 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 | *Default:* |
| **T** string  [1] | 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. | *Default:* |
| **I** string  [1] | Indigenous/Tribal: e.g. Chief Red Cloud | *Default:* |
| **P** string  [1] | 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) | *Default:* |
| **ANON** string  [1] | Anonymous : Anonymous assigned name (used to protect a person's identity for privacy reasons) | *Default:* |
| **A** string  [1] | 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) | *Default:* |
| **R** string  [1] | Religious : A name assumed as part of a religious vocation. e.g. Sister Mary Francis, Brother John | *Default:* |
| **OLD** string  [1] | 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) | *Default:* |
| **DN** string  [1] | 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". | *Default:* |
| **M** string  [1] | 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 | *Default:* |
| **PHON** string  [1] | 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. | *Default:* |
| **SRCH** string  [1] | Search Type Uses: A name intended for use in searching or matching | *Default:* |

### IntegrityCheckAlgorithm

Type: **Enumeration** Derived From:

The algorithm used to compute the integrityCheck value.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **SHA1** string  [1] | Secure Hash Algorithm - 1 : This algorithm is defined in FIPS PUB 180-1: Secure Hash Standard. As of April 17, 1995 | *Default:* |
| **SHA256** string  [1] | Secure Hash Algorithm - 256 : This algorithm is defined in FIPS PUB 180-2: Secure Hash Standard | *Default:* |

### PostalAddressUse

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **H** string  [1] | 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. | *Default:* |
| **HP** string  [1] | Primary Home: The primary home, to reach a person after business hours. | *Default:* |
| **HV** string  [1] | Vacation Home: A vacation home, to reach a person while on vacation. | *Default:* |
| **WP** string  [1] | Work Place: An office address. First choice for business related contacts during business hours. | *Default:* |
| **DIR** string  [1] | 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'. | *Default:* |
| **PUB** string  [1] | 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. | *Default:* |
| **BAD** string  [1] | Bad Address: A flag indicating that the address is bad, in fact, useless. | *Default:* |
| **PHYS** string  [1] | Physical Visit Address: Used primarily to visit an address. | *Default:* |
| **PST** string  [1] | Postal Address: Used to send mail. | *Default:* |
| **TMP** string  [1] | Temporary Address: A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information. | *Default:* |
| **ABC** string  [1] | Alphabetic: Alphabetic transcription of name (Japanese: romaji) | *Default:* |
| **IDE** string  [1] | Ideographic: Ideographic representation of name (e.g., Japanese kanji, Chinese characters) | *Default:* |
| **SYL** string  [1] | Syllabic: Syllabic transcription of name (e.g., Japanese kana, Korean hangul) | *Default:* |
| **SRCH** string  [1] | Search Type Uses: A name intended for use in searching or matching. | *Default:* |
| **SNDX** string  [1] | Soundex: An address spelled according to the SoundEx algorithm. | *Default:* |
| **PHON** string  [1] | 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. | *Default:* |

### TelecommunicationAddressUse

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **H** string  [1] | 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 | *Default:* |
| **HP** string  [1] | Primary Home: The primary home, to reach a person after business hours. | *Default:* |
| **HV** string  [1] | Vacation Home: vacation home, to reach a person while on vacation. | *Default:* |
| **WP** string  [1] | Work Place: An office address. First choice for business related contacts during business hours. | *Default:* |
| **DIR** string  [1] | 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'. | *Default:* |
| **PUB** string  [1] | 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. | *Default:* |
| **BAD** string  [1] | Bad Address: A flag indicating that the address is bad, in fact, useless. | *Default:* |
| **TMP** string  [1] | Temporary Address: A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information. | *Default:* |
| **AS** string  [1] | 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. | *Default:* |
| **EC** string  [1] | Emergency Contact: A contact specifically designated to be used for emergencies. This is the first choice in emergencies, independent of any other use codes. | *Default:* |
| **MC** string  [1] | 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. | *Default:* |
| **PG** string  [1] | Pager: A paging device suitable to solicit a callback or to leave a very short message. | *Default:* |

### TelecommunicationCapability

Type: **Enumeration** Derived From:

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

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **voice** string  [1] | Voice : This device can receive voice calls (i.e. talking to another person, or a recording device, or a voice activated computer) | *Default:* |
| **fax** string  [1] | Fax : This device can receive faxes. | *Default:* |
| **data** string  [1] | Data : This device can receive data calls (i.e. modem) | *Default:* |
| **tty** string  [1] | Text : This device is a text telephone. | *Default:* |
| **sms** string  [1] | SMS : This device can receive SMS messages | *Default:* |

## Base

This file defines the base types used in defining knowledge artifacts.

**Base** - *(Class diagram)*



Figure: 2

### Evidence

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **qualityOfEvidence** CD  [0..1] | 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). | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **resources** SupportingResource  [0..1] | The set of resource references associated with  the evidence. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **strengthOfRecommendation** CD  [0..1] | 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). | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### InlineResource

Type: **Class** Derived From: KnowledgeResource

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [InlineResource](#BKM_EB1567C8_B381_4E94_8546_03B95450A025) | [KnowledgeResource](#BKM_00DBA520_44BD_4124_B1CB_AD682ED8A232) |  |

#### content

Type: **Class** Derived From:

The document content in xhtml format.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **ext\_ref\_1** div  [1] |  | *Default:*    [anonymousRole = true ]  [anonymousType = false ]  [maxOccurs = 1 ]  [minOccurs = 1 ] |

### KnowledgeResource

Type: **Class** Derived From:

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [InlineResource](#BKM_EB1567C8_B381_4E94_8546_03B95450A025) | [KnowledgeResource](#BKM_00DBA520_44BD_4124_B1CB_AD682ED8A232) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **citation** ST  [0..1] | The resource citation. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **description** ST  [0..1] | A short textual description of the resource. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **location** TEL  [0..1] | The URL of the given resource. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **title** ST  [0..1] | The title of the document | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### identifiers

Type: **Class** Derived From:

The set of unique identifiers for this resource.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **identifier** VersionedIdentifier  [1] | A unique resource identifier. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### templateIds

Type: **Class** Derived From:

The set of unique identifiers for the templates

associated with this resource.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **templateId** II  [1] | A unique identifier for the template  associated with this resource. Templates are used to constrain  the resource model. For more information on this process,  please refer to the implementation guide. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### Organization

Type: **Class** Derived From: Party

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Organization](#BKM_7B37919B_0424_4E2A_BA39_5D22F1D305F8) | [Party](#BKM_058851E7_DAD5_4A92_87C2_FE63048EEC9F) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **name** ST  [1] | The name of the organization. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### Party

Type: **Class** Derived From:

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).

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Organization](#BKM_7B37919B_0424_4E2A_BA39_5D22F1D305F8) | [Party](#BKM_058851E7_DAD5_4A92_87C2_FE63048EEC9F) |  |
| [Person](#BKM_BB84EAF0_E4D2_48E6_8B4D_0D83C313B812) | [Party](#BKM_058851E7_DAD5_4A92_87C2_FE63048EEC9F) |  |

#### addresses

Type: **Class** Derived From:

The set of addresses associated with this entity.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **address** AD  [1..\*] | An individual address of type AD associated with this entity. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### contacts

Type: **Class** Derived From:

The set of contact information associated with this entity.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **contact** TEL  [1..\*] | An individual contact item of type TEL associated with this entity. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### Person

Type: **Class** Derived From: Party

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

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Person](#BKM_BB84EAF0_E4D2_48E6_8B4D_0D83C313B812) | [Party](#BKM_058851E7_DAD5_4A92_87C2_FE63048EEC9F) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **affiliation** Organization  [0..1] | The organizational affiliation for this person. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **name** EN  [1] | The name of the person. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### ResourceRelationshipReference

Type: **Class** Derived From:

ResourceRelationshipReference defines the association between a resource and a resource set. If one considers 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.

#### relationship

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ResourceRelationshipType  [1] |  | *Default:*    [use = required ] |

#### resources

Type: **Class** Derived From:

The set of resources associated with the subject of this relationship.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **resource** KnowledgeResource  [1..\*] | An associated resource instance. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### SupportingEvidence

Type: **Class** Derived From:

The evidence grade and the sources of evidence associated with this artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **evidence** Evidence  [1..\*] | A single evidence reference. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### SupportingResource

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **resource** KnowledgeResource  [1..\*] | An individual supporting resource. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### VersionedIdentifier

Type: **Class** Derived From: II

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [VersionedIdentifier](#BKM_82E8B711_839C_4D5A_8AA8_C77FE391EA4A) | [II](#BKM_51542036_E725_4E17_976D_D340C608641C) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **version** string  [1] |  | *Default:* |

## Behavior

This file defines the behaviors used within knowledge documents.

**Behavior** - *(Class diagram)*



Figure: 3

### Behavior

Type: **Class** Derived From:

A behavior may be specified for a specific action or a group of actions. This is the base type for all Behaviors.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [GroupSelectionBehavior](#BKM_523D78EC_F434_428A_86E1_BCC6E24489E1) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |
| [GroupOrganizationBehavior](#BKM_0FCD7276_9EED_45F3_8E0C_148FB7FDFDE6) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |
| [RequiredBehavior](#BKM_DC6C3FC6_EF8C_4F22_B0CC_8A0CB05E3A49) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |
| [PrecheckBehavior](#BKM_E3B1C0A1_8B54_45B1_A5F1_3F0CF5CC798A) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |

### Behaviors

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **behavior** Behavior  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### GroupOrganizationBehavior

Type: **Class** Derived From: Behavior

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [GroupOrganizationBehavior](#BKM_0FCD7276_9EED_45F3_8E0C_148FB7FDFDE6) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** GroupOrganizationBehaviorType  [1] |  | *Default:* |

### GroupSelectionBehavior

Type: **Class** Derived From: Behavior

For a group of actions, specifies the number of actions that may be chosen by an end user.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [GroupSelectionBehavior](#BKM_523D78EC_F434_428A_86E1_BCC6E24489E1) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** GroupSelectionBehaviorType  [1] |  | *Default:* |

### PrecheckBehavior

Type: **Class** Derived From: Behavior

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PrecheckBehavior](#BKM_E3B1C0A1_8B54_45B1_A5F1_3F0CF5CC798A) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** PrecheckBehaviorType  [1] |  | *Default:* |

### RequiredBehavior

Type: **Class** Derived From: Behavior

For a single action, specifies what level of requiredness is associated with the action.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RequiredBehavior](#BKM_DC6C3FC6_EF8C_4F22_B0CC_8A0CB05E3A49) | [Behavior](#BKM_FF02AD14_8602_4198_BBB1_FD18FC189B8A) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** RequiredBehaviorType  [1] |  | *Default:* |

## Enum

### ArtifactLifeCycleEventType

Type: **Enumeration** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Created** string  [1] | The date/time when a particular version of an artifact is created may be recorded in the history of the artifact-version. | *Default:* |
| **Pre-published** string  [1] | 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. | *Default:* |
| **Published** string  [1] |  | *Default:* |
| **Reviewed** string  [1] | 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. | *Default:* |
| **Withdrawn** string  [1] | A specific version of an artifact may be withdrawn by a publisher for various reasons. | *Default:* |
| **Superseded** string  [1] | A specific version of an artifact may be superseded by another version of that same artifact, or by another artifact altogether. | *Default:* |

### ArtifactStatusType

Type: **Enumeration** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Draft** string  [1] |  | *Default:* |
| **InTest** string  [1] |  | *Default:* |
| **Active** string  [1] |  | *Default:* |
| **Inactive** string  [1] |  | *Default:* |

### ArtifactType

Type: **Enumeration** Derived From:

Three types of artifacts are in scope for Health eDecisions Use Case #1: Event-condition-action rules, order sets and documentation templates.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Rule** string  [1] | 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. | *Default:* |
| **Order Set** string  [1] | 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. | *Default:* |
| **Documentation Template** string  [1] | 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. | *Default:* |
| **Library** string  [1] | Denotes a Library of artifact components that can be reused by reference in other artifacts. | *Default:* |

### Cardinality

Type: **Enumeration** Derived From:

Cardinality defines the expected cardinality of an element, single

or multiple.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Single** string  [1] |  | *Default:* |
| **Multiple** string  [1] |  | *Default:* |

### ConditionRoleType

Type: **Enumeration** Derived From:

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)

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **ApplicableScenario** string  [1] | 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. | *Default:* |

### ContributorType

Type: **Enumeration** Derived From:

Enumeration of roles that contribute to the development and maintenance of a knowledge artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Author** string  [1] |  | *Default:* |
| **Editor** string  [1] |  | *Default:* |
| **Endorser** string  [1] |  | *Default:* |
| **Reviewer** string  [1] |  | *Default:* |

### CoverageType

Type: **Enumeration** Derived From:

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).

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **PatientGender** string  [1] | The gender of the patient. For this item type, use HL7 administrative gender codes (OID: 2.16.840.1.113883.1.11.1) | *Default:* |
| **PatientAgeGroup** string  [1] | 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] | *Default:* |
| **ClinicalFocus** string  [1] | 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. | *Default:* |
| **TargetUser** string  [1] | 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) | *Default:* |
| **WorkflowSetting** string  [1] | The settings in which the artifact is intended for use. For example, admission, pre-op, etc. | *Default:* |
| **WorkflowTask** string  [1] | 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 | *Default:* |
| **ClinicalVenue** string  [1] | 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 | *Default:* |

### DataEventType

Type: **Enumeration** Derived From:

Enumeration of types of events related to access, creation, removal, or update of data.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **DataElementAdded** string  [1] |  | *Default:* |
| **DataElementModified** string  [1] |  | *Default:* |
| **DataElementRemoved** string  [1] |  | *Default:* |
| **DataElementAccessed** string  [1] |  | *Default:* |
| **DataElementAccessEnded** string  [1] |  | *Default:* |

### EventType

Type: **Enumeration** Derived From:

An enumeration of event types. Events occur external to the artifact that can be used as a trigger to the artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **DataEvent** string  [1] | 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. | *Default:* |
| **PeriodicEvent** string  [1] | 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 | *Default:* |

### GroupOrganizationBehaviorType

Type: **Enumeration** Derived From:

Defines organization behavior of a group: gives the reason why the items are grouped together.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **VisualGroup** string  [1] | Any group marked with this behavior should be displayed as a visual group to the end user. | *Default:* |
| **LogicalGroup** string  [1] | 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. | *Default:* |
| **SentenceGroup** string  [1] | 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". | *Default:* |

### GroupSelectionBehaviorType

Type: **Enumeration** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Any** string  [1] | Any number of the items in the group may be chosen, from zero to all. | *Default:* |
| **All** string  [1] | All the items in the group must be selected as a single unit. | *Default:* |
| **AllOrNone** string  [1] | 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. | *Default:* |
| **ExactlyOne** string  [1] | 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. | *Default:* |
| **AtMostOne** string  [1] | The end user may choose zero or at most one of the items in the group. | *Default:* |
| **OneOrMore** string  [1] | The end user must choose a minimum of one, and as many additional as desired. | *Default:* |

### PrecheckBehaviorType

Type: **Enumeration** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Yes** string  [1] | 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. | *Default:* |
| **No** string  [1] | 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. | *Default:* |

### RangeConstraintType

Type: **Enumeration** Derived From:

The enumeration of different types of range constraints on values.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Minimum** string  [1] | The constraint specifies the lower bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only. | *Default:* |
| **Maximum** string  [1] | The constraint specifies the upper bound or minimum value of a documentation item value. This constraint type applies to quantitative value types only. | *Default:* |
| **List** string  [1] | The constraint restricts the value to items from a list. | *Default:* |
| **Component** string  [1] |  | *Default:* |

### RequiredBehaviorType

Type: **Enumeration** Derived From:

Defines requiredness behavior for selecting an action or an action group; i.e., whether the action or action group is required or optional.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Must** string  [1] | 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. | *Default:* |
| **Could** string  [1] | An action with this behavior may be included in the set of actions processed by the end user. | *Default:* |
| **MustUnlessDocumented** string  [1] | 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. | *Default:* |

### ResourceRelationshipType

Type: **Enumeration** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **AdaptedFrom** string  [1] |  | *Default:* |
| **AssociatedResource** string  [1] |  | *Default:* |
| **DependsOn** string  [1] |  | *Default:* |
| **DerivedFrom** string  [1] |  | *Default:* |
| **SimilarTo** string  [1] |  | *Default:* |
| **VersionOf** string  [1] |  | *Default:* |

### ValueType

Type: **Enumeration** Derived From:

A specification of a constraint on the range of values for an item.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Boolean** string  [1] |  | *Default:* |
| **Code** string  [1] |  | *Default:* |
| **CodedOrdinal** string  [1] |  | *Default:* |
| **EntityName** string  [1] |  | *Default:* |
| **Identifier** string  [1] |  | *Default:* |
| **Integer** string  [1] |  | *Default:* |
| **IntegerInterval** string  [1] |  | *Default:* |
| **Period** string  [1] |  | *Default:* |
| **PhysicalQuantity** string  [1] |  | *Default:* |
| **PhysicalQuantityInterval** string  [1] |  | *Default:* |
| **QuantityInterval** string  [1] |  | *Default:* |
| **Ratio** string  [1] |  | *Default:* |
| **Real** string  [1] |  | *Default:* |
| **RealInterval** string  [1] |  | *Default:* |
| **SimpleCode** string  [1] |  | *Default:* |
| **String** string  [1] |  | *Default:* |
| **Timestamp** string  [1] |  | *Default:* |
| **TimestampInterval** string  [1] |  | *Default:* |
| **URL** string  [1] |  | *Default:* |

## Metadata

This file defines the meta data components used within a knowledge artifact.

### ArtifactLifeCycleEvent

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **eventDateTime** TS  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### eventType

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ArtifactLifeCycleEventType  [1] |  | *Default:*    [use = required ] |

### Contribution

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **contributor** Party  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### role

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ContributorType  [1] |  | *Default:* |

### Coverage

Type: **Class** Derived From:

Specifies various attributes of the patient population for whom and/or environment of care in which the CDS artifact is applicable.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **description** ST  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **value** CD  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### focus

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** CoverageType  [1] |  | *Default:*    [use = required ] |

### LibraryReference

Type: **Class** Derived From:

A reference to a library.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **mediaType** anyURI  [1] | Defines the type of the library. If this attribute is omitted, the library is assumed to be an HeD library artifact. | *Default:*    [default = application/hed+xml ]  [use = optional ] |
| **name** string  [1] | A unique name within this artifact for the library reference. This name is used within this artifact to reference components of this library. | *Default:*    [use = required ] |
| **path** anyURI  [1] | Defines the path to the library. | *Default:*    [use = required ] |

### Metadata

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **description** ST  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **documentation** InlineResource  [0..1] | Documentation for this knowledge reference may  consist of a reference to an external resource; the documentation  may also be included in-line if desired. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **language** CD  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **schemaIdentifier** VersionedIdentifier  [1] | This is the identifier of the XML schema (and its version) which governs the structure of this CDS Knowledge Artifact. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **supportingEvidence** SupportingEvidence  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **title** ST  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### applicability

Type: **Class** Derived From:

Specifies the conditions under which this

artifact is applicable.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **coverage** Coverage  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### artifactType

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ArtifactType  [1] |  | *Default:*    [use = required ] |

#### categories

Type: **Class** Derived From:

Provides a list of coded categories to which this

artifact belongs.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **category** CD  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### contributions

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **contribution** Contribution  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### dataModels

Type: **Class** Derived From:

Set of data models referenced in the Expression

objects in this knowledge artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **modelReference** ModelReference  [1..\*] | A reference for the data models that are  used in the artifact, e..g., the Virtual Medical Record. In  this case, the name could be "vmr" and the value is  the namespace universal resource identifier of the HL7 VMR schema | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### eventHistory

Type: **Class** Derived From:

This is the history of events which have occurred

for this particular version of the artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **artifactLifeCycleEvent** ArtifactLifeCycleEvent  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### identifiers

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **identifier** VersionedIdentifier  [1..\*] | The identifier of a CDS knowledge artifact consists of two parts: (1) a base identifier which uniquely identifies the CDS Knowledge Artifact from all other CDS Knowledge Artifacts, regardless of version. (2) a version identifier which uniquely identifies the CDS Knowledge Artifact from all other CDS Knowledge Artifact versions.  Essentially, the full identifier is for a particular "artifact version". | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### keyTerms

Type: **Class** Derived From:

Provides a list of coded key terms that pertain to this artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **term** CD  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### libraries

Type: **Class** Derived From:

Set of libraries referenced by this artifact. Components of referenced libraries may be referenced by this artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **libraryReference** LibraryReference  [1..\*] | A reference to a library whose component can be referenced  within the artifact. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### publishers

Type: **Class** Derived From:

The set of people and/or organizations who

publish the artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **publisher** Party  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### relatedResources

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **relatedResource** ResourceRelationshipReference  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### status

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ArtifactStatusType  [1] |  | *Default:*    [use = required ] |

#### templateIds

Type: **Class** Derived From:

These are the identifiers of templates which

further constrain the structure of this knowledge artifact.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **templateId** VersionedIdentifier  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### usageTerms

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **rightsDeclaration** RightsDeclaration  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### ModelReference

Type: **Class** Derived From:

A reference to some model by its Universal Resource Identifier.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **description** ST  [1] | A description of the model. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### referencedModel

Type: **Class** Derived From:

The identifier of the object that is being

referenced.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** anyURI  [1] | The URI of the object that is being referenced. This URL must also be defined as a namespace in the root element of the document | *Default:* |

### RightsDeclaration

Type: **Class** Derived From:

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 permissions are specified as a free-form text string.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **assertedRights** ST  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **rightsHolder** Party  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### permissions

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **permissions** ST  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

## Expression

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.

### AggregateOperators

#### AllTrue

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AllTrue](#BKM_145E6A36_4F8F_4CF3_96A0_5F2F1E460398) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### AnyTrue

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AnyTrue](#BKM_698C8DAB_0BA4_45DD_8E8D_6BD2EC19B442) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Avg

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Avg](#BKM_0593B2E8_30BB_4B1E_B683_F38C815251FB) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Count

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Count](#BKM_FB1E3C58_87F8_4C77_96C3_DB60C31D34C0) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Max

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Max](#BKM_A5ABC1E1_7BE0_4E3D_934E_30BFDEF38C3C) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Median

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Median](#BKM_005DC0D4_45F0_40B6_9D73_0991A9DBECDB) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Min

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Min](#BKM_6F0C0C73_7D29_43FD_9AFC_3A01C3A6565D) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Mode

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Mode](#BKM_9015D7C1_F2DF_4FB1_9EC8_784E500F4D45) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### PopulationStdDev

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PopulationStdDev](#BKM_A5DE0F63_9BE3_402F_9D32_4A1FECBAAB91) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### PopulationVariance

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PopulationVariance](#BKM_9404FC95_2F04_43D1_A4ED_363AAF0A0DA6) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### StdDev

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [StdDev](#BKM_6D1B7366_3132_4B5A_B7D3_72A68314F222) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Sum

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Sum](#BKM_D41BFAFA_F529_4420_BBD7_3F1BAED84D24) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

#### Variance

Type: **Class** Derived From: AggregateExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Variance](#BKM_58591B47_77AA_478C_BFF1_427E9A09E080) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

### ArithmeticOperators

#### Abs

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Abs](#BKM_02863D5F_D95E_45C6_A06C_E98DD914E531) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Add

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Add](#BKM_234A6077_7830_4D9C_B2A1_9557BF170FBC) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Ceiling

Type: **Class** Derived From: UnaryExpression

The Ceiling operator returns the first integer greater than or equal to the argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Ceiling](#BKM_02636B8C_DBA5_4EF5_A456_D1092739A349) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Divide

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Divide](#BKM_E7BA236F_5AFB_4946_9DCC_37778C379960) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Floor

Type: **Class** Derived From: UnaryExpression

The Floor operator returns the first integer less than or equal to the argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Floor](#BKM_F5EA2E27_2B23_4B8D_A3B0_E377850FC17E) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Ln

Type: **Class** Derived From: UnaryExpression

The Ln operator computes the natural logarithm of its argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Ln](#BKM_20D1EF12_9117_45BF_A94D_50C2211D093A) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Log

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Log](#BKM_086B3A4D_5D11_4D86_B0A9_87207293DDD5) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Modulo

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Modulo](#BKM_6C462234_4977_48F9_B1A3_581BBE8E464A) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Multiply

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Multiply](#BKM_F7A3108A_57F8_4FB7_9748_647EDC97EA10) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Negate

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Negate](#BKM_5C78AD9A_DB9F_4EB1_95B6_86978931DAD2) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Power

Type: **Class** Derived From: BinaryExpression

The Power operator raises the first argument to the power given by the second argument.

If either argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Power](#BKM_DC627D6B_6D2E_42DE_9BC5_D580941412FB) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Round

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Round](#BKM_187AE3C3_4A29_45AD_8470_ACEE50BC8ABE) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **operand** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **precision** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### Subtract

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Subtract](#BKM_14C4033C_FAE7_4B78_BA54_2BB1067D3859) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Truncate

Type: **Class** Derived From: UnaryExpression

The Truncate operator returns the integer component of its argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Truncate](#BKM_4BE3E765_BC81_400A_8D88_FEE72EAB266F) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### TruncatedDivide

Type: **Class** Derived From: BinaryExpression

The TruncatedDivide operator performs integer division of its arguments.

If either argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TruncatedDivide](#BKM_3E503B83_9A73_416A_80A2_750EA36A98EF) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

### ComparisonOperators

#### Equal

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Equal](#BKM_DA5C9DBC_E049_4265_AB72_E53A878C8069) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Greater

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Greater](#BKM_5933E598_AEC9_4D1D_9D31_9CD6E312D8B7) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### GreaterOrEqual

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [GreaterOrEqual](#BKM_265D7BF2_82B0_440A_BB93_8E09DE5AFED4) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Less

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Less](#BKM_F13DFE14_55FB_48EB_9324_0601B5325473) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### LessOrEqual

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [LessOrEqual](#BKM_960864D6_6152_4511_A37D_02BBDC7B9465) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### NotEqual

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [NotEqual](#BKM_E46549BB_389B_40DA_8AB1_B091E6453B03) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

### ConditionalOperators

#### Case

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Case](#BKM_4883005A_DDB7_42F2_9C76_573C5A3807D5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **caseItem** CaseItem  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |
| **comparand** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **else** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### CaseItem

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **then** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **when** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Coalesce

Type: **Class** Derived From: NaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Coalesce](#BKM_C1815CAF_F81A_4D3E_B70A_C1D3DD76BD2C) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

#### Conditional

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Conditional](#BKM_0A375361_644D_4F35_B3FC_022C2C4E521A) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **condition** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **else** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **then** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### CoreElements

**CoreElements** - *(Class diagram)*



Figure: 4

#### AggregateExpression

Type: **Class** Derived From: Expression

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).

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Count](#BKM_FB1E3C58_87F8_4C77_96C3_DB60C31D34C0) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Sum](#BKM_D41BFAFA_F529_4420_BBD7_3F1BAED84D24) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Min](#BKM_6F0C0C73_7D29_43FD_9AFC_3A01C3A6565D) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Max](#BKM_A5ABC1E1_7BE0_4E3D_934E_30BFDEF38C3C) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Avg](#BKM_0593B2E8_30BB_4B1E_B683_F38C815251FB) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Median](#BKM_005DC0D4_45F0_40B6_9D73_0991A9DBECDB) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Mode](#BKM_9015D7C1_F2DF_4FB1_9EC8_784E500F4D45) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [Variance](#BKM_58591B47_77AA_478C_BFF1_427E9A09E080) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [PopulationVariance](#BKM_9404FC95_2F04_43D1_A4ED_363AAF0A0DA6) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [StdDev](#BKM_6D1B7366_3132_4B5A_B7D3_72A68314F222) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [PopulationStdDev](#BKM_A5DE0F63_9BE3_402F_9D32_4A1FECBAAB91) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [AllTrue](#BKM_145E6A36_4F8F_4CF3_96A0_5F2F1E460398) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |
| [AnyTrue](#BKM_698C8DAB_0BA4_45DD_8E8D_6BD2EC19B442) | [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **path** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### BinaryExpression

Type: **Class** Derived From: Expression

The BinaryExpression type defines the abstract base type for all expressions that take two arguments.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [IfNull](#BKM_3BEA3DC3_E10D_4C2C_A733_40C4D3ECDD57) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Equal](#BKM_DA5C9DBC_E049_4265_AB72_E53A878C8069) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [NotEqual](#BKM_E46549BB_389B_40DA_8AB1_B091E6453B03) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Less](#BKM_F13DFE14_55FB_48EB_9324_0601B5325473) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Greater](#BKM_5933E598_AEC9_4D1D_9D31_9CD6E312D8B7) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [LessOrEqual](#BKM_960864D6_6152_4511_A37D_02BBDC7B9465) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [GreaterOrEqual](#BKM_265D7BF2_82B0_440A_BB93_8E09DE5AFED4) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Add](#BKM_234A6077_7830_4D9C_B2A1_9557BF170FBC) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Subtract](#BKM_14C4033C_FAE7_4B78_BA54_2BB1067D3859) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Multiply](#BKM_F7A3108A_57F8_4FB7_9748_647EDC97EA10) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Divide](#BKM_E7BA236F_5AFB_4946_9DCC_37778C379960) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [TruncatedDivide](#BKM_3E503B83_9A73_416A_80A2_750EA36A98EF) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Modulo](#BKM_6C462234_4977_48F9_B1A3_581BBE8E464A) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Log](#BKM_086B3A4D_5D11_4D86_B0A9_87207293DDD5) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Power](#BKM_DC627D6B_6D2E_42DE_9BC5_D580941412FB) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Contains](#BKM_AB00873E_E971_468C_B7C1_8F9DB1F7CD63) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [In](#BKM_30460EA4_6E0F_41E1_96C2_98EFE70F88E0) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Includes](#BKM_9A767CBD_BADD_4FB5_AEE4_F2DD2F23B61F) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [IncludedIn](#BKM_7593A786_0747_4D42_BF35_3B6BB1DA7315) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [ProperIncludes](#BKM_B757B342_9D28_4073_9522_BBBBFF5CF6C2) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [ProperIncludedIn](#BKM_4A183DAC_C7F9_474C_BD2D_20F7B1E35AF3) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Before](#BKM_E2A392E6_254E_4420_BDDF_9473B0B683A9) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [After](#BKM_FAE7134E_432E_429B_8DFC_5135E5A33C10) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Meets](#BKM_83A78536_F177_49CB_892A_5FBD00131B50) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Overlaps](#BKM_2ECCAF6D_AC09_4306_B706_C946FE488A57) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [OverlapsBefore](#BKM_A94E0F12_4D61_4B22_8325_708731E68A7F) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [OverlapsAfter](#BKM_0DE77813_6050_4927_A54B_C87A2BBA5B36) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Begins](#BKM_AFD192CE_BCD7_4FDA_9CAD_7E91920A1D72) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Ends](#BKM_94DBEC89_EAE7_44FC_BA93_A21FFBDD0DDE) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |
| [Difference](#BKM_026F1EDD_58A8_4B55_8F45_A5EAE16C2E04) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **operand** Expression  [2] |  | *Default:*    [maxOccurs = 2 ]  [minOccurs = 2 ] |

#### Expression

Type: **Class** Derived From:

The Expression type defines the abstract base type for all expressions used in the HeDS expression language.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [TernaryExpression](#BKM_F26554E3_E366_48AC_986C_615FB7566BFF) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ExpressionRef](#BKM_0B67F2C5_4343_4292_8FFF_72ABFAEF7E01) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ParameterRef](#BKM_F115D230_D9F7_4BFA_B065_ECC08634DE1C) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Literal](#BKM_E73D63CA_6728_401E_8EBA_3C18DCFCD323) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ComplexLiteral](#BKM_46CE0091_0888_49DE_9548_33732797D964) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ObjectExpression](#BKM_A90DB4A3_3FE1_42AE_9A90_DEA9033855CC) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ObjectDescriptor](#BKM_FDE45307_B792_4A58_B9B9_4C2BAC297A46) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ObjectRedefine](#BKM_D9AB6E58_5722_4E04_ADE5_BD576CE71557) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Interval](#BKM_5654A0CB_6D5F_4D57_8FED_55A137F2E699) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [List](#BKM_ED9F5067_CC22_414B_BC08_C8D27EA9A6B7) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Conditional](#BKM_0A375361_644D_4F35_B3FC_022C2C4E521A) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Case](#BKM_4883005A_DDB7_42F2_9C76_573C5A3807D5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Null](#BKM_9BEEA5A2_A6B9_4D0E_9428_FA48B493C2C5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Round](#BKM_187AE3C3_4A29_45AD_8470_ACEE50BC8ABE) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [MinValue](#BKM_7568C285_C0E4_4D4D_A9CF_0D881B9C73FA) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [MaxValue](#BKM_9DBCC977_6C0D_4B81_A365_E91FAC1E7099) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Combine](#BKM_7FB5A435_CE7A_40DB_88B8_3AB14BD122E8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Split](#BKM_A68DC2B8_6074_426A_B5F9_4FBD57D9B05F) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Indexer](#BKM_49BC2101_9845_4077_B9E9_F1D1C18207BA) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Pos](#BKM_78E274DD_46E0_4E64_8C9A_209BEB2B0B42) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Substring](#BKM_81B0B1CD_31C1_434E_9EE1_0A7149AA2DE5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [DateAdd](#BKM_9B40827C_11F6_4832_B8E1_1A594BF2BCD5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [DateDiff](#BKM_D61648C1_5013_421C_B98F_901E3EDFB8B2) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [DatePart](#BKM_ECAE7A28_0BA1_4A4A_8EDB_83941FA09CC7) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Today](#BKM_5CC5E486_D2DE_4564_94C4_77F6937D1B1D) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Now](#BKM_413ECB2D_C9B7_4714_9FB7_2B43917652F6) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Date](#BKM_33073B0E_B100_4024_B89C_F8736C8DF62F) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Filter](#BKM_4791A5DD_75BE_47F0_9A45_79F41D986BB0) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [First](#BKM_C2C1BB80_CD40_4A51_9A56_E92986733A75) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Last](#BKM_06656B15_FE1D_438B_BF5C_FD4C693CB5F2) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [IndexOf](#BKM_9266DAA6_3430_4CA6_AB03_BD4398961ED4) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Sort](#BKM_553C391E_E461_45A3_86EE_776FB608B6C1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ForEach](#BKM_D4B77B9C_63F7_4C68_BDDA_20B441DD1AFB) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Distinct](#BKM_13252CE5_5ED8_4111_9075_F9ED9C2A018B) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Current](#BKM_218ABED2_95B5_4765_A8E2_0CCADBF98821) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [AggregateExpression](#BKM_6E494192_6952_425F_8A90_F71EDF012D8D) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Property](#BKM_591A024B_213B_47F2_A65C_7D6FB172B2F9) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [AddressLiteral](#BKM_71DD55BB_306B_406A_875F_C540FCF1150A) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [BooleanLiteral](#BKM_C46048CE_393F_4323_9E0B_6F31177ED053) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [CodeLiteral](#BKM_F7C3F80E_823F_45E1_99B3_FEEBD5422FC8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [CodedOrdinalLiteral](#BKM_01224A63_D6EE_4D16_83B5_967B29B950B3) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [SimpleCodeLiteral](#BKM_69A6E2B5_1464_433D_AB57_A83EC6252291) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [EntityNameLiteral](#BKM_E342EC5C_25C1_4588_830E_9E0CB728A6C1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [IdentifierLiteral](#BKM_DA490734_2082_401C_AA24_58C7AA0C34D8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [IntegerLiteral](#BKM_D6204E22_D98D_4748_B18B_BAEDD59988E1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [IntegerIntervalLiteral](#BKM_067C8173_E0EC_4441_BD56_657F70D209C9) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [PhysicalQuantityIntervalLiteral](#BKM_013DA214_86A0_42A2_967A_F232F5D31AEF) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [QuantityIntervalLiteral](#BKM_22C4F21E_B987_4324_94C6_33A4617E2D48) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [RealIntervalLiteral](#BKM_8E528F72_6136_49E5_AA43_040872E33F9B) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [TimestampIntervalLiteral](#BKM_8BB77361_810F_4073_8FBA_6F04D420ECA4) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [PhysicalQuantityLiteral](#BKM_80791764_71FD_448A_B93E_47A144DF6707) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [RealLiteral](#BKM_EB436F76_7AEF_4FB8_B06F_B38506305D02) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [RatioLiteral](#BKM_EDBCF546_5000_467F_B996_847E6518F8A3) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [StringLiteral](#BKM_5579F811_66A0_4585_8747_300771DAF9A1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [UrlLiteral](#BKM_722FAE02_1718_42E0_8029_2B5DC3CD218E) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [TimestampLiteral](#BKM_D1CA7683_B689_4A5F_A940_19D855DBBE2E) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [PeriodLiteral](#BKM_4E54F00F_EEF8_47B9_A8C2_5FD7CACDC2E8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [RequestBase](#BKM_57ECEBB2_9624_411C_AD92_BEA0903D35D1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [ValueSet](#BKM_D24AEDBD_BB87_44D8_BB2D_716BA21D2835) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Subsumes](#BKM_6A4C7C08_5571_4324_812F_F9C1992E36B4) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [SetSubsumes](#BKM_845EFC79_531B_4DC0_A389_33517EA185C9) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **annotation** anyType  [0..\*] | The annotation element provides a mechanism for decorating expressions with application-specific information such as translation hints, visual designer information, or debug symbols. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |
| **description** string  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### ExpressionDef

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **expression** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **name** string  [1] |  | *Default:* |

#### ExpressionRef

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ExpressionRef](#BKM_0B67F2C5_4343_4292_8FFF_72ABFAEF7E01) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **name** string  [1] |  | *Default:* |

#### NaryExpression

Type: **Class** Derived From: Expression

The NaryExpression type defines an abstract base class for an expression that takes any number of arguments, including zero.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [And](#BKM_FCD74973_96F1_4948_A2C0_6FC4FD9DF374) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |
| [Or](#BKM_2F2A81E5_66D4_4600_9805_AD5ADA02D213) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |
| [Coalesce](#BKM_C1815CAF_F81A_4D3E_B70A_C1D3DD76BD2C) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |
| [Concat](#BKM_CD61AE31_D93F_4E76_869C_3BDFAFA820F1) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |
| [Union](#BKM_05887727_1893_437B_85B0_59560BD8CD3F) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |
| [Intersect](#BKM_1F430774_696C_4EA9_860F_2801B3B3CA76) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **operand** Expression  [0..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

#### ParameterDef

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **default** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **name** string  [1] |  | *Default:* |
| **parameterType** QName  [1] |  | *Default:* |

#### ParameterRef

Type: **Class** Derived From: Expression

The ParameterRef expression allows the value of a parameter to be referenced as part of an expression.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ParameterRef](#BKM_F115D230_D9F7_4BFA_B065_ECC08634DE1C) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **name** string  [1] |  | *Default:* |

#### TernaryExpression

Type: **Class** Derived From: Expression

The TernaryExpression type defines the abstract base type for all expressions that take three arguments.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TernaryExpression](#BKM_F26554E3_E366_48AC_986C_615FB7566BFF) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **operand** Expression  [3] |  | *Default:*    [maxOccurs = 3 ]  [minOccurs = 3 ] |

#### UnaryExpression

Type: **Class** Derived From: Expression

The UnaryExpression type defines the abstract base type for all expressions that take a single argument.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [Not](#BKM_1E24597A_3DCB_4237_B231_484CC39770F8) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [IsNull](#BKM_2CBC360B_A1CF_467C_BBB6_2CF3B4FDFA19) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Is](#BKM_86684B19_BFD8_4848_904D_56062BCEE223) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [As](#BKM_C8D1672D_4E9C_4157_9BEB_A5358A20F080) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Convert](#BKM_A47E66B2_D0D8_43FF_81AC_09737EDB9571) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Ceiling](#BKM_02636B8C_DBA5_4EF5_A456_D1092739A349) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Floor](#BKM_F5EA2E27_2B23_4B8D_A3B0_E377850FC17E) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Truncate](#BKM_4BE3E765_BC81_400A_8D88_FEE72EAB266F) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Abs](#BKM_02863D5F_D95E_45C6_A06C_E98DD914E531) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Negate](#BKM_5C78AD9A_DB9F_4EB1_95B6_86978931DAD2) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Ln](#BKM_20D1EF12_9117_45BF_A94D_50C2211D093A) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Succ](#BKM_37E85599_95E5_4F66_AE2E_186937F02875) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Pred](#BKM_56E4C1B4_452E_4085_8729_C026406231EA) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Length](#BKM_815F85CF_7BE0_44AA_816B_18E0F902A1FB) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Upper](#BKM_61DFB85D_474B_48B8_BA9A_786EB7CFA72D) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Lower](#BKM_A861850A_F587_4988_90DD_94B23AF3FE0F) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [DateOf](#BKM_000312CA_BBB2_4BB9_A238_68882CC5A5FE) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [TimeOf](#BKM_548136DA_1D0B_491D_830C_5C2D45BB94EF) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Begin](#BKM_2116E945_B734_41C0_A0F8_8688A2D8D4C1) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [End](#BKM_AC8B9E5C_0FB1_4325_8632_D1798F9CA4BB) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Collapse](#BKM_AE54F731_42D3_4D5E_989B_957CA404DFF8) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [IsEmpty](#BKM_74344F4C_66EA_4DAF_B227_C3AB33DDE4C8) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [IsNotEmpty](#BKM_75F458CF_98CE_40D7_B605_3075E4A95EEB) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [Expand](#BKM_D4ED1F66_9ADA_4808_B510_7C865DDC6515) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |
| [InValueSet](#BKM_288D6A37_C5AC_489F_A124_F52607112ABE) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **operand** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### DateTimeOperators

#### Date

Type: **Class** Derived From: Expression

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 millisceond are all null, they are assumed to be 0. However, if hour is null, minute, second, and millisecond must all be null as well.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Date](#BKM_33073B0E_B100_4024_B89C_F8736C8DF62F) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **day** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **hour** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **millisecond** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **minute** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **month** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **second** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **year** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### DateAdd

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DateAdd](#BKM_9B40827C_11F6_4832_B8E1_1A594BF2BCD5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **date** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **granularity** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **numberOfPeriods** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### DateDiff

Type: **Class** Derived From: Expression

The DateDiff operator returns the number of granularity boundaries occurring between startDate and endDate.

If either argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DateDiff](#BKM_D61648C1_5013_421C_B98F_901E3EDFB8B2) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **endDate** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **granularity** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **startDate** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### DateOf

Type: **Class** Derived From: UnaryExpression

The DateOf operator returns the date (with no time component) of the argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DateOf](#BKM_000312CA_BBB2_4BB9_A238_68882CC5A5FE) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### DatePart

Type: **Class** Derived From: Expression

The DatePart operator returns the granularity component of the given date.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DatePart](#BKM_ECAE7A28_0BA1_4A4A_8EDB_83941FA09CC7) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **date** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **granularity** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Now

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Now](#BKM_413ECB2D_C9B7_4714_9FB7_2B43917652F6) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

#### TimeOf

Type: **Class** Derived From: UnaryExpression

The TimeOf operator returns the time (with no date component) of the argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TimeOf](#BKM_548136DA_1D0B_491D_830C_5C2D45BB94EF) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Today

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Today](#BKM_5CC5E486_D2DE_4564_94C4_77F6937D1B1D) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

#### DateGranularity

Type: **Enumeration** Derived From:

The DateGranularity type specifies the granularities available for temporal operations such as DateAdd, DateDiff, and DatePart.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Year** string  [1] |  | *Default:* |
| **Month** string  [1] |  | *Default:* |
| **Week** string  [1] |  | *Default:* |
| **Day** string  [1] |  | *Default:* |
| **Hour** string  [1] |  | *Default:* |
| **Minute** string  [1] |  | *Default:* |
| **Second** string  [1] |  | *Default:* |
| **Millisecond** string  [1] |  | *Default:* |

### IntervalOperators

#### After

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [After](#BKM_FAE7134E_432E_429B_8DFC_5135E5A33C10) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Before

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Before](#BKM_E2A392E6_254E_4420_BDDF_9473B0B683A9) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Begin

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Begin](#BKM_2116E945_B734_41C0_A0F8_8688A2D8D4C1) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Begins

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Begins](#BKM_AFD192CE_BCD7_4FDA_9CAD_7E91920A1D72) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### End

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [End](#BKM_AC8B9E5C_0FB1_4325_8632_D1798F9CA4BB) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Ends

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Ends](#BKM_94DBEC89_EAE7_44FC_BA93_A21FFBDD0DDE) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Interval

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Interval](#BKM_5654A0CB_6D5F_4D57_8FED_55A137F2E699) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **begin** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **beginOpen** boolean  [1] |  | *Default:*    [default = false ]  [use = optional ] |
| **end** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **endOpen** boolean  [1] |  | *Default:*    [default = false ]  [use = optional ] |

#### Meets

Type: **Class** Derived From: BinaryExpression

The Meets operator returns true if the first interval ends immediately before the second interval starts, or if the first interval starts immediately after the second interval ends. In other words, if the ending point of the first interval is equal to the predecessor of the starting point of the second, or if the starting point of the first interval is equal to the successor of 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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Meets](#BKM_83A78536_F177_49CB_892A_5FBD00131B50) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Overlaps

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Overlaps](#BKM_2ECCAF6D_AC09_4306_B706_C946FE488A57) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### OverlapsAfter

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [OverlapsAfter](#BKM_0DE77813_6050_4927_A54B_C87A2BBA5B36) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### OverlapsBefore

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [OverlapsBefore](#BKM_A94E0F12_4D61_4B22_8325_708731E68A7F) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

### ListOperators

#### Collapse

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Collapse](#BKM_AE54F731_42D3_4D5E_989B_957CA404DFF8) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Current

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Current](#BKM_218ABED2_95B5_4765_A8E2_0CCADBF98821) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **scope** string  [1] |  | *Default:*    [use = optional ] |

#### Difference

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Difference](#BKM_026F1EDD_58A8_4B55_8F45_A5EAE16C2E04) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Distinct

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Distinct](#BKM_13252CE5_5ED8_4111_9075_F9ED9C2A018B) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Expand

Type: **Class** Derived From: UnaryExpression

The Expand operator flattens a list of lists into a single list.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Expand](#BKM_D4ED1F66_9ADA_4808_B510_7C865DDC6515) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Filter

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Filter](#BKM_4791A5DD_75BE_47F0_9A45_79F41D986BB0) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **condition** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **scope** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### First

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [First](#BKM_C2C1BB80_CD40_4A51_9A56_E92986733A75) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **orderBy** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### ForEach

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ForEach](#BKM_D4B77B9C_63F7_4C68_BDDA_20B441DD1AFB) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **element** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **scope** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### IndexOf

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IndexOf](#BKM_9266DAA6_3430_4CA6_AB03_BD4398961ED4) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **element** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Indexer

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Indexer](#BKM_49BC2101_9845_4077_B9E9_F1D1C18207BA) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **index** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **operand** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Intersect

Type: **Class** Derived From: NaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Intersect](#BKM_1F430774_696C_4EA9_860F_2801B3B3CA76) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

#### IsEmpty

Type: **Class** Derived From: UnaryExpression

The IsEmpty operator returns true if the list contains no elements.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IsEmpty](#BKM_74344F4C_66EA_4DAF_B227_C3AB33DDE4C8) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### IsNotEmpty

Type: **Class** Derived From: UnaryExpression

The IsNotEmpty operator returns true if the list contains any elements.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IsNotEmpty](#BKM_75F458CF_98CE_40D7_B605_3075E4A95EEB) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Last

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Last](#BKM_06656B15_FE1D_438B_BF5C_FD4C693CB5F2) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **orderBy** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Length

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Length](#BKM_815F85CF_7BE0_44AA_816B_18E0F902A1FB) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### List

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [List](#BKM_ED9F5067_CC22_414B_BC08_C8D27EA9A6B7) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **element** Expression  [0..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |
| **key** string  [1] |  | *Default:*    [use = optional ] |

#### Sort

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Sort](#BKM_553C391E_E461_45A3_86EE_776FB608B6C1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **orderBy** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Union

Type: **Class** Derived From: NaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Union](#BKM_05887727_1893_437B_85B0_59560BD8CD3F) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

### LogicalOperators

#### And

Type: **Class** Derived From: NaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [And](#BKM_FCD74973_96F1_4948_A2C0_6FC4FD9DF374) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

#### Not

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Not](#BKM_1E24597A_3DCB_4237_B231_484CC39770F8) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Or

Type: **Class** Derived From: NaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Or](#BKM_2F2A81E5_66D4_4600_9805_AD5ADA02D213) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

### MembershipOperators

#### Contains

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Contains](#BKM_AB00873E_E971_468C_B7C1_8F9DB1F7CD63) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### In

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [In](#BKM_30460EA4_6E0F_41E1_96C2_98EFE70F88E0) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### IncludedIn

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IncludedIn](#BKM_7593A786_0747_4D42_BF35_3B6BB1DA7315) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### Includes

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Includes](#BKM_9A767CBD_BADD_4FB5_AEE4_F2DD2F23B61F) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### ProperIncludedIn

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ProperIncludedIn](#BKM_4A183DAC_C7F9_474C_BD2D_20F7B1E35AF3) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### ProperIncludes

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ProperIncludes](#BKM_B757B342_9D28_4073_9522_BBBBFF5CF6C2) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

### NullologicalOperators

#### IfNull

Type: **Class** Derived From: BinaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IfNull](#BKM_3BEA3DC3_E10D_4C2C_A733_40C4D3ECDD57) | [BinaryExpression](#BKM_355F8BE3_D94E_40B1_8B28_A7308F55CB19) |  |

#### IsNull

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IsNull](#BKM_2CBC360B_A1CF_467C_BBB6_2CF3B4FDFA19) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Null

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Null](#BKM_9BEEA5A2_A6B9_4D0E_9428_FA48B493C2C5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **valueType** QName  [1] |  | *Default:*    [use = required ] |

### StringOperators

#### Combine

Type: **Class** Derived From: Expression

The Combine operator combines a list of strings, optionally separating each string with the given separator.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Combine](#BKM_7FB5A435_CE7A_40DB_88B8_3AB14BD122E8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **separator** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **source** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Concat

Type: **Class** Derived From: NaryExpression

The Concat operator performs string concatenation of its arguments.

If any argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Concat](#BKM_CD61AE31_D93F_4E76_869C_3BDFAFA820F1) | [NaryExpression](#BKM_FA60EA5D_66B0_4D00_92AA_DB370A125C7D) |  |

#### Lower

Type: **Class** Derived From: UnaryExpression

The Lower operator returns the lower case of its argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Lower](#BKM_A861850A_F587_4988_90DD_94B23AF3FE0F) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Pos

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Pos](#BKM_78E274DD_46E0_4E64_8C9A_209BEB2B0B42) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **pattern** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **string** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Split

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Split](#BKM_A68DC2B8_6074_426A_B5F9_4FBD57D9B05F) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **separator** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **stringToSplit** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Substring

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Substring](#BKM_81B0B1CD_31C1_434E_9EE1_0A7149AA2DE5) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

#### Upper

Type: **Class** Derived From: UnaryExpression

The Upper operator returns the upper case of its argument.

If the argument is null, the result is null.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Upper](#BKM_61DFB85D_474B_48B8_BA9A_786EB7CFA72D) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

### StructuredValueOperators

#### As

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [As](#BKM_C8D1672D_4E9C_4157_9BEB_A5358A20F080) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **asType** QName  [1] |  | *Default:*    [use = required ] |
| **strict** boolean  [1] |  | *Default:*    [default = false ]  [use = optional ] |

#### Is

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Is](#BKM_86684B19_BFD8_4848_904D_56062BCEE223) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **isType** QName  [1] |  | *Default:*    [use = required ] |

#### ObjectDescriptor

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ObjectDescriptor](#BKM_FDE45307_B792_4A58_B9B9_4C2BAC297A46) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **objectType** QName  [1] |  | *Default:* |
| **property** PropertyExpression  [0..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

#### ObjectExpression

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ObjectExpression](#BKM_A90DB4A3_3FE1_42AE_9A90_DEA9033855CC) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **objectType** QName  [1] |  | *Default:* |
| **property** PropertyExpression  [0..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

#### ObjectRedefine

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ObjectRedefine](#BKM_D9AB6E58_5722_4E04_ADE5_BD576CE71557) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **property** PropertyExpression  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |
| **scope** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### Property

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Property](#BKM_591A024B_213B_47F2_A65C_7D6FB172B2F9) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **path** string  [1] |  | *Default:*    [use = required ] |
| **scope** string  [1] |  | *Default:*    [use = optional ] |
| **source** Expression  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### PropertyExpression

Type: **Class** Derived From:

The PropertyExpression type is used within the ObjectLiteral type to provide the value of a specific property within an object literal expression.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **name** string  [1] |  | *Default:*    [use = required ] |
| **value** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### ValueOperators

#### ComplexLiteral

Type: **Class** Derived From: Expression

The ComplexLiteral expression allows an xml literal of any type to be included in an expression.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ComplexLiteral](#BKM_46CE0091_0888_49DE_9548_33732797D964) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** anyType  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### Convert

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Convert](#BKM_A47E66B2_D0D8_43FF_81AC_09737EDB9571) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **toType** QName  [1] |  | *Default:*    [use = required ] |

#### Literal

Type: **Class** Derived From: Expression

The Literal type defines a single scalar value. For example, the literal 5, the boolean value true or the string "antithrombotic".

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Literal](#BKM_E73D63CA_6728_401E_8EBA_3C18DCFCD323) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** anySimpleType  [1] |  | *Default:*    [use = optional ] |
| **valueType** QName  [1] |  | *Default:*    [use = required ] |

#### MaxValue

Type: **Class** Derived From: Expression

The MaxValue operator returns the maximum representable value for the given type.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [MaxValue](#BKM_9DBCC977_6C0D_4B81_A365_E91FAC1E7099) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **valueType** QName  [1] |  | *Default:*    [use = required ] |

#### MinValue

Type: **Class** Derived From: Expression

The MinValue operator returns the minimum representable value for the given type.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [MinValue](#BKM_7568C285_C0E4_4D4D_A9CF_0D881B9C73FA) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **valueType** QName  [1] |  | *Default:*    [use = required ] |

#### Pred

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Pred](#BKM_56E4C1B4_452E_4085_8729_C026406231EA) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

#### Succ

Type: **Class** Derived From: UnaryExpression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Succ](#BKM_37E85599_95E5_4F66_AE2E_186937F02875) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

## Literalexpression

This file defines additional expressions that provide syntactic short-hands for literals for each of the base data types.

### AddressLiteral

Type: **Class** Derived From: Expression

The AddressLiteral expression returns a value of type AD with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AddressLiteral](#BKM_71DD55BB_306B_406A_875F_C540FCF1150A) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **part** ADXP  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |
| **use** set\_PostalAddressUse  [1] |  | *Default:*    [use = optional ] |

### BooleanLiteral

Type: **Class** Derived From: Expression

The BooleanLiteral expression returns a value of type BL with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [BooleanLiteral](#BKM_C46048CE_393F_4323_9E0B_6F31177ED053) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** boolean  [1] |  | *Default:*    [use = required ] |

### CodeLiteral

Type: **Class** Derived From: Expression

The CodeLiteral expression returns a value of type CD with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CodeLiteral](#BKM_F7C3F80E_823F_45E1_99B3_FEEBD5422FC8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** Code  [1] |  | *Default:*    [use = required ] |
| **codeSystem** Uid  [1] |  | *Default:*    [use = required ] |
| **codeSystemName** string  [1] |  | *Default:*    [use = optional ] |
| **codeSystemVersion** string  [1] |  | *Default:*    [use = optional ] |
| **displayName** string  [1] |  | *Default:*    [use = optional ] |
| **valueSet** Uid  [1] |  | *Default:*    [use = optional ] |
| **valueSetVersion** string  [1] |  | *Default:*    [use = optional ] |

### CodedOrdinalLiteral

Type: **Class** Derived From: Expression

The CodedOrdinalLiteral expression returns a value of type CO with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CodedOrdinalLiteral](#BKM_01224A63_D6EE_4D16_83B5_967B29B950B3) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** Code  [1] |  | *Default:*    [use = required ] |
| **codeSystem** Uid  [1] |  | *Default:*    [use = required ] |
| **codeSystemName** string  [1] |  | *Default:*    [use = optional ] |
| **displayName** string  [1] |  | *Default:*    [use = optional ] |
| **value** Decimal  [1] |  | *Default:*    [use = required ] |

### EntityNameLiteral

Type: **Class** Derived From: Expression

The EntityNameLiteral expression returns a value of type EN with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [EntityNameLiteral](#BKM_E342EC5C_25C1_4588_830E_9E0CB728A6C1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **part** ENXP  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |
| **use** set\_EntityNameUse  [1] |  | *Default:*    [use = optional ] |

### IdentifierLiteral

Type: **Class** Derived From: Expression

The IdentifierLiteral expression returns a value of type II with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IdentifierLiteral](#BKM_DA490734_2082_401C_AA24_58C7AA0C34D8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **extension** string  [1] |  | *Default:*    [use = optional ] |
| **root** Uid  [1] |  | *Default:*    [use = required ] |

### IntegerIntervalLiteral

Type: **Class** Derived From: Expression

The IntegerIntervalLiteral expression returns a value of type IVL\_INT with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IntegerIntervalLiteral](#BKM_067C8173_E0EC_4441_BD56_657F70D209C9) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** int  [1] |  | *Default:* |
| **highClosed** boolean  [1] |  | *Default:*    [use = optional ] |
| **low** int  [1] |  | *Default:* |
| **lowClosed** boolean  [1] |  | *Default:*    [use = optional ] |

### IntegerLiteral

Type: **Class** Derived From: Expression

The IntegerLiteral expression returns a value of type INT with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [IntegerLiteral](#BKM_D6204E22_D98D_4748_B18B_BAEDD59988E1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** int  [1] |  | *Default:*    [use = required ] |

### PeriodLiteral

Type: **Class** Derived From: Expression

The PeriodLiteral expression returns a value of type PIVL\_TS with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PeriodLiteral](#BKM_4E54F00F_EEF8_47B9_A8C2_5FD7CACDC2E8) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **alignment** CalendarCycle  [1] |  | *Default:*    [use = optional ] |
| **count** INT  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **frequency** RTO  [0..1] | 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). | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **isFlexible** boolean  [1] | 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". | *Default:*    [use = optional ] |
| **period** PQ  [0..1] | A time duration specified as a reciprocal measure of the frequency at which the PIVL repeats. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **phase** IVL\_TS  [0..1] | 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 | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### PhysicalQuantityIntervalLiteral

Type: **Class** Derived From: Expression

The PhysicalQuantityIntervalLiteral expression returns a value of type IVL\_PQ with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PhysicalQuantityIntervalLiteral](#BKM_013DA214_86A0_42A2_967A_F232F5D31AEF) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** PQ  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] |  | *Default:*    [use = optional ] |
| **low** PQ  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] |  | *Default:*    [use = optional ] |

### PhysicalQuantityLiteral

Type: **Class** Derived From: Expression

The PhysicalQuantityLiteral expression returns a value of type PQ with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [PhysicalQuantityLiteral](#BKM_80791764_71FD_448A_B93E_47A144DF6707) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **unit** Code  [1] |  | *Default:*    [use = required ] |
| **value** Decimal  [1] |  | *Default:*    [use = required ] |

### QuantityIntervalLiteral

Type: **Class** Derived From: Expression

The QuantityIntervalLiteral expression returns a value of type IVL\_QTY with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [QuantityIntervalLiteral](#BKM_22C4F21E_B987_4324_94C6_33A4617E2D48) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** QTY  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] |  | *Default:*    [use = optional ] |
| **low** QTY  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] |  | *Default:*    [use = optional ] |

### RatioLiteral

Type: **Class** Derived From: Expression

The RaioLiteral expression returns a value of type RTO with the given numerator and denominator.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RatioLiteral](#BKM_EDBCF546_5000_467F_B996_847E6518F8A3) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **denominator** QTY  [1] | The quantity that divides the numerator in the ratio.  The denominator SHALL not be zero. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **numerator** QTY  [1] | The quantity that is being divided in the ratio | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### RealIntervalLiteral

Type: **Class** Derived From: Expression

The RealIntervalLiteral expression returns a value of type IVL\_REAL with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RealIntervalLiteral](#BKM_8E528F72_6136_49E5_AA43_040872E33F9B) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** Decimal  [1] |  | *Default:* |
| **highClosed** boolean  [1] |  | *Default:*    [use = optional ] |
| **low** Decimal  [1] |  | *Default:* |
| **lowClosed** boolean  [1] |  | *Default:*    [use = optional ] |

### RealLiteral

Type: **Class** Derived From: Expression

The RealLiteral expression returns a value of type Real with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RealLiteral](#BKM_EB436F76_7AEF_4FB8_B06F_B38506305D02) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** Decimal  [1] |  | *Default:*    [use = required ] |

### SimpleCodeLiteral

Type: **Class** Derived From: Expression

The SimpleCodeLiteral expression returns a value of type CS with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [SimpleCodeLiteral](#BKM_69A6E2B5_1464_433D_AB57_A83EC6252291) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** string  [1] |  | *Default:*    [use = required ] |

### StringLiteral

Type: **Class** Derived From: Expression

The StringLiteral expression returns a value of type ST with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [StringLiteral](#BKM_5579F811_66A0_4585_8747_300771DAF9A1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** string  [1] |  | *Default:*    [use = required ] |

### TimestampIntervalLiteral

Type: **Class** Derived From: Expression

The TimestampIntervalLiteral expression returns a value of type IVL\_TS with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TimestampIntervalLiteral](#BKM_8BB77361_810F_4073_8FBA_6F04D420ECA4) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **high** TS  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **highClosed** boolean  [1] |  | *Default:*    [use = optional ] |
| **low** TS  [0..1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **lowClosed** boolean  [1] |  | *Default:*    [use = optional ] |

### TimestampLiteral

Type: **Class** Derived From: Expression

The TimestampLiteral expression returns a value of type TS with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [TimestampLiteral](#BKM_D1CA7683_B689_4A5F_A940_19D855DBBE2E) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** TimeStamp  [1] |  | *Default:*    [use = required ] |

### UrlLiteral

Type: **Class** Derived From: Expression

The UrlLiteral expression returns a value of type TEL with the given attributes.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [UrlLiteral](#BKM_722FAE02_1718_42E0_8029_2B5DC3CD218E) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **capabilities** set\_TelecommunicationCapability  [1] |  | *Default:*    [use = optional ] |
| **use** set\_TelecommunicationAddressUse  [1] |  | *Default:*    [use = optional ] |
| **value** anyURI  [1] |  | *Default:*    [use = required ] |

## Clinicalexpression

This file defines the RequestBase extensions that introduce clinically relevant dependencies such as terminology, relevant time, and value set considerations.

**Clinicalexpression** - *(Class diagram)*



Figure: 5

### ClinicalRequest

Type: **Class** Derived From: RequestBase

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ClinicalRequest](#BKM_0E478F6C_1652_4423_9084_23946622640B) | [RequestBase](#BKM_57ECEBB2_9624_411C_AD92_BEA0903D35D1) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **codeProperty** string  [1] | 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. | *Default:*    [use = optional ] |
| **codes** Expression  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **dateProperty** string  [1] | 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. | *Default:*    [use = optional ] |
| **dateRange** Expression  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **subject** Expression  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **subjectProperty** string  [1] | The subjectProperty optionally specifies which property of the model contains the related value for the subject of the clinical request. | *Default:*    [use = optional ] |
| **useSubsumption** boolean  [1] | The useSubsumption attribute determines whether subsumption should be used to determine whether or not a given data item should be included in the result. | *Default:*    [default = false ]  [use = optional ] |
| **useValueSets** boolean  [1] | 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. | *Default:*    [default = false ]  [use = optional ] |

### DataRequest

Type: **Class** Derived From: RequestBase

The DataRequest expression provides basic data access.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DataRequest](#BKM_DF251DDE_8381_4036_8598_9AFF473838A5) | [RequestBase](#BKM_57ECEBB2_9624_411C_AD92_BEA0903D35D1) |  |

### InValueSet

Type: **Class** Derived From: UnaryExpression

The InValueSet operator returns true if the given code, or list of codes, is in the given value set.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [InValueSet](#BKM_288D6A37_C5AC_489F_A124_F52607112ABE) | [UnaryExpression](#BKM_79267805_023C_4836_AC40_6CB5EA171F80) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **authority** string  [1] | 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. | *Default:*    [use = optional ] |
| **id** string  [1] | The unique identifier of the value set to be retrieved. This identifier is evaluated within the context of the authority. | *Default:*    [use = required ] |
| **version** string  [1] | The version of the value set to be retrieved. If an asterisk (\*) is used, the current version of the value set is returned. | *Default:*    [use = optional ] |

### RequestBase

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RequestBase](#BKM_57ECEBB2_9624_411C_AD92_BEA0903D35D1) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |
| [DataRequest](#BKM_DF251DDE_8381_4036_8598_9AFF473838A5) | [RequestBase](#BKM_57ECEBB2_9624_411C_AD92_BEA0903D35D1) |  |
| [ClinicalRequest](#BKM_0E478F6C_1652_4423_9084_23946622640B) | [RequestBase](#BKM_57ECEBB2_9624_411C_AD92_BEA0903D35D1) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **cardinality** RequestCardinality  [1] | 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. | *Default:*    [use = required ] |
| **dataType** QName  [1] | The dataType attribute specifies the type of clinical data being requested. | *Default:*    [use = required ] |
| **idProperty** string  [1] | The idProperty attribute specifies which property of the model contains the Id for the clinical statement. | *Default:*    [use = optional ] |
| **isInitial** boolean  [1] | The isInitial attribute determines whether the request is part of the initial data requirements for the artifact. | *Default:*    [default = true ]  [use = optional ] |
| **scope** string  [1] | The scope attribute optionally specifies a name for the scope for this operator that can be used within expressions to reference the current element. | *Default:*    [use = optional ] |
| **templateId** string  [1] | 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. | *Default:*    [use = optional ] |
| **timeOffset** Expression  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **triggerType** DataEventType  [1] | 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. | *Default:*    [use = optional ] |

### SetSubsumes

Type: **Class** Derived From: Expression

The SetSubsumes operator returns the list of descendants that were subsumed by some code in the list of ancestors.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [SetSubsumes](#BKM_845EFC79_531B_4DC0_A389_33517EA185C9) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **ancestors** Expression  [0..1] | The codes that will be tested for the ancestor relationship. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **descendents** Expression  [0..1] | The codes that will be tested for the descendent relationship. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### Subsumes

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [Subsumes](#BKM_6A4C7C08_5571_4324_812F_F9C1992E36B4) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **ancestor** Expression  [0..1] | The code that will be tested for the ancestor relationship. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **descendent** Expression  [0..1] | The code that will be tested for the descendent relationship. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### ValueSet

Type: **Class** Derived From: Expression

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ValueSet](#BKM_D24AEDBD_BB87_44D8_BB2D_716BA21D2835) | [Expression](#BKM_5A5467F7_3AD8_4801_AB1A_A31B743A72EC) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **authority** string  [1] | 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. | *Default:*    [use = optional ] |
| **id** string  [1] | The unique identifier of the value set to be retrieved. This identifier is evaluated within the context of the authority. | *Default:*    [use = required ] |
| **version** string  [1] | The version of the value set to be retrieved. If an asterisk (\*) is used, the current version of the value set is returned. | *Default:*    [use = optional ] |

### RequestCardinality

Type: **Enumeration** Derived From:

RequestCardinality defines the expected cardinality of the request, single or multiple.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **Single** string  [1] |  | *Default:* |
| **Multiple** string  [1] |  | *Default:* |

## Action

This file defines the Action types used to describe guidance within a knowledge artifact.

**Action** - *(Class diagram)*



Figure: 6

### ActionBase

Type: **Class** Derived From:

Actions are the output of the CDS system and represent the tasks that must be carried out by a human or a computer system.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ActionGroup](#BKM_AF985B2F_7F3D_45D9_A6BB_759052C6B000) | [ActionBase](#BKM_4D47E0B4_3E0C_4C72_AC72_10C1275A8D25) |  |
| [ActionRef](#BKM_FB7343C0_B318_446E_800B_E58B239F8C82) | [ActionBase](#BKM_4D47E0B4_3E0C_4C72_AC72_10C1275A8D25) |  |
| [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) | [ActionBase](#BKM_4D47E0B4_3E0C_4C72_AC72_10C1275A8D25) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionId** II  [0..1] | An identifier for the action. The identifier must  be unique within the scope of the artifact. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **behaviors** Behaviors  [0..1] | The behaviors associated with how the action is  presented and executed. The semantics and the validity of  behaviors for actions are described elsewhere. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **conditions** Conditions  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **supportingEvidence** SupportingEvidence  [0..1] | The evidence grade and the sources of evidence  associated with this artifact. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **supportingResources** SupportingResource  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### actors

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actor** Actor  [1..\*] | The actor that is responsible for executing  the action.  This element is used when an artifact can have  multiple actors  responsible for the execution of various CDS  actions,  forexample, an interdisciplinary plan of care.  It is  important to note the distinction between a CDS action and  an  ensuing clinical action. A CDS action might be to order  patient  ambulation, the actor for which is a physician  responsible for  writing the order. A nurse might be responsible  for ensuring  that the patient ambulates. In this case, the  artifact will  specify the physician as the actor. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### ActionGroup

Type: **Class** Derived From: ActionBase

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ActionGroup](#BKM_AF985B2F_7F3D_45D9_A6BB_759052C6B000) | [ActionBase](#BKM_4D47E0B4_3E0C_4C72_AC72_10C1275A8D25) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **description** ST  [0..1] | A lengthier description of the action group  that can  be displayed to the user or the recipient of the CDS | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **title** ST  [0..1] | A brief title that is shown to the user of the  artifact, i.e., the recipient of the CDS | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### representedConcepts

Type: **Class** Derived From:

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).

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **concept** CD  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

#### subElements

Type: **Class** Derived From:

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [subElements](#BKM_12B6F619_E743_4E90_9DD6_873A6CD55C96) | [actionGroup element type](#BKM_83390F75_B0F8_4541_9321_8E00CD5B37FD) |  |

### ActionRef

Type: **Class** Derived From: ActionBase

A reference to an action defined in a library.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ActionRef](#BKM_FB7343C0_B318_446E_800B_E58B239F8C82) | [ActionBase](#BKM_4D47E0B4_3E0C_4C72_AC72_10C1275A8D25) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **referencedActionId** string  [1] | The Id of the action being referenced. | *Default:* |

### Actor

Type: **Class** Derived From:

An actor is an entity responsible for the execution of an action.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actor** Expression  [1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### AtomicAction

Type: **Class** Derived From: ActionBase

An action that is not further broken down into constituent actions.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) | [ActionBase](#BKM_4D47E0B4_3E0C_4C72_AC72_10C1275A8D25) |  |
| [CreateAction](#BKM_AFCD768D_52EE_4C83_B05C_E812C8D93630) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |
| [UpdateAction](#BKM_0112EA27_5249_42C6_BFB7_C1C31F09DEB7) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |
| [RemoveAction](#BKM_90B2868E_D64F_48CD_AB7F_61A1B29AD7FC) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |
| [FireEventAction](#BKM_88E281A1_EF2D_4756_A819_B77CB0F7F144) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |
| [DeclareResponseAction](#BKM_53EF0A63_9327_46E8_B582_76130585FED4) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |
| [CollectInformationAction](#BKM_A3EA4250_FEB2_418F_90A6_1B8475101C25) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **textEquivalent** ST  [0..1] | A brief textual description of the action that  summarizes the action | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### CollectInformationAction

Type: **Class** Derived From: AtomicAction

This action requests information from the actor. The information request is specified as a DocumentationItem.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CollectInformationAction](#BKM_A3EA4250_FEB2_418F_90A6_1B8475101C25) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **documentationConcept** DocumentationItem  [1] | This provides a specification of the  information to be collected from the user. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **initialValue** Expression  [0..1] | 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 | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **responseBinding** ResponseBinding  [0..1] | Defines the response binding for the documentation item. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### CreateAction

Type: **Class** Derived From: AtomicAction

A new action to be executed by a user or a computer system. The sentence provides the details of the action to be executed.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [CreateAction](#BKM_AFCD768D_52EE_4C83_B05C_E812C8D93630) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionSentence** Expression  [0..1] | The parameters of the action that is to be  executed. For example, an action may be to order a medication. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### DeclareResponseAction

Type: **Class** Derived From: AtomicAction

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DeclareResponseAction](#BKM_53EF0A63_9327_46E8_B582_76130585FED4) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **name** string  [1] |  | *Default:*    [default = Responses ]  [use = optional ] |

### DocumentationItem

Type: **Class** Derived From: ItemDefinition

An item type representing the definition of an individual item to be recorded in a structured clinical document.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DocumentationItem](#BKM_660E5285_7E6A_4E64_AB6E_778A5E418555) | [ItemDefinition](#BKM_2409E2D2_F927_44B5_B584_C80D8F04065A) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **description** ST  [0..1] | A lengthier description of this documentation  item that is displayed to the user. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **displayText** ST  [1] | Brief text or title for this documentation item  that is the caption displayed to the user performing the  documentation. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **responseRange** RangeConstraint  [0..\*] | The constraints (within the responseDataType)  on the values that may be entered by the user | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

#### itemCodes

Type: **Class** Derived From:

A collection of codes for concepts that are the

equivalents of this documentation item.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **itemCode** CD  [0..\*] | A code for a concept that is the equivalent  of this documentation item. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

#### responseCardinality

Type: **Class** Derived From:

The number of allowed responses, single or

multiple.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** Cardinality  [1] |  | *Default:* |

#### responseDataType

Type: **Class** Derived From:

The data type of the value or the response that

is entered by the user.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ValueType  [1] |  | *Default:* |

### ExpressionConstraint

Type: **Class** Derived From: RangeConstraint

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ExpressionConstraint](#BKM_2661E98C_3A55_448D_A872_98A031B65C4D) | [RangeConstraint](#BKM_AD959182_C870_4940_AD49_ADF33FF46505) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **constraint** Expression  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### FireEventAction

Type: **Class** Derived From: AtomicAction

This action fires an event. The event can serve as a trigger to another artifact.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [FireEventAction](#BKM_88E281A1_EF2D_4756_A819_B77CB0F7F144) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionSentence** Expression  [0..1] | The action sentence is the payload of the  event. Another artifact receives this payload as an input. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **eventType** EventType  [1] | The type of the event that is fired. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### ItemDefinition

Type: **Class** Derived From:

An item definition is the equivalent of an item in a data dictionary or a catalog in an electronic health record system.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [DocumentationItem](#BKM_660E5285_7E6A_4E64_AB6E_778A5E418555) | [ItemDefinition](#BKM_2409E2D2_F927_44B5_B584_C80D8F04065A) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **identifier** VersionedIdentifier  [0..1] | The identifier for the item. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### ListConstraint

Type: **Class** Derived From: RangeConstraint

A constraint specifying that the value is from a list included here.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ListConstraint](#BKM_036B2ED0_0815_4D36_B1AF_F259391007FA) | [RangeConstraint](#BKM_AD959182_C870_4940_AD49_ADF33FF46505) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **strictSelection** boolean  [1] | 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. | *Default:* |

#### item

Type: **Class** Derived From:

An item in the list of possible values

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **displayText** ST  [0..1] | Short text label for the list item. When  displayText is present, this is displayed to the user  performing the documentation. In such a case, the value is  displayed optionally | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **value** Expression  [1] | The value of the item is specified as an  expression. This is the value that could be displayed to the  user and which forms the recorded value of the documentation  item. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

##### codes

Type: **Class** Derived From:

A collection of codes that are the

equivalent of this value.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **code** CD  [0..\*] | An individual code that is the equivalent  of this value. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

### RangeConstraint

Type: **Class** Derived From:

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ExpressionConstraint](#BKM_2661E98C_3A55_448D_A872_98A031B65C4D) | [RangeConstraint](#BKM_AD959182_C870_4940_AD49_ADF33FF46505) |  |
| [ListConstraint](#BKM_036B2ED0_0815_4D36_B1AF_F259391007FA) | [RangeConstraint](#BKM_AD959182_C870_4940_AD49_ADF33FF46505) |  |
| [ValueSetConstraint](#BKM_7B2BAD01_4C40_492C_A36D_78C5FD2402E0) | [RangeConstraint](#BKM_AD959182_C870_4940_AD49_ADF33FF46505) |  |

#### constraintType

Type: **Class** Derived From:

The constraint type defines how the value range

is being constrained. For example, the constraint type may

indicate the lower bound of the range.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** RangeConstraintType  [1] |  | *Default:* |

### RemoveAction

Type: **Class** Derived From: AtomicAction

This action removes another proposed action or an ongoing action.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [RemoveAction](#BKM_90B2868E_D64F_48CD_AB7F_61A1B29AD7FC) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionSentence** Expression  [1] | The expression must resolve to the action that  is being removed. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### ResponseBinding

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **container** string  [1] |  | *Default:*    [default = Responses ]  [use = optional ] |
| **property** string  [1] |  | *Default:*    [use = required ] |

### UpdateAction

Type: **Class** Derived From: AtomicAction

This action changes the value of another existing action. The action being modified may be a proposed action (e.g., a 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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [UpdateAction](#BKM_0112EA27_5249_42C6_BFB7_C1C31F09DEB7) | [AtomicAction](#BKM_E22D33EB_95CC_46C5_BA9B_3A4543216739) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionSentence** Expression  [1] | The modification to the action. This is  specified by modifying the properties of an existing action using  an ObjectRedefine expression. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### ValueSetConstraint

Type: **Class** Derived From: RangeConstraint

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.

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [ValueSetConstraint](#BKM_7B2BAD01_4C40_492C_A36D_78C5FD2402E0) | [RangeConstraint](#BKM_AD959182_C870_4940_AD49_ADF33FF46505) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **valueSet** ValueSet  [1] |  | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

### actionGroup element type

Type: **Class** Derived From:

*See Also*

| **Source** | **Target** | **Notes** |
| --- | --- | --- |
| [subElements](#BKM_12B6F619_E743_4E90_9DD6_873A6CD55C96) | [actionGroup element type](#BKM_83390F75_B0F8_4541_9321_8E00CD5B37FD) |  |

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionGroup** ActionGroup  [1] | This subelement is a group of actions that  are specified in line. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **actionRef** ActionRef  [1] | This subelement specifies an action to be included by reference  from a referenced library as defined in the libraries section of the metadata. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **simpleAction** AtomicAction  [1] | The subelement is an atomic or single  action. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |

## Knowledgedocument

This file defines the root knowledge document type and element.

### Condition

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **logic** Expression  [1] | 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. | *Default:*    [anonymousRole = false ]  [default = ]  [fixed = ]  [form = ]  [maxOccurs = 1 ]  [minOccurs = 1 ] |

#### conditionRole

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **value** ConditionRoleType  [1] |  | *Default:* |

### Conditions

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **condition** Condition  [1..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### Trigger

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **eventType** EventType  [1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **expression** Expression  [0..1] | The event expression that must be met for the trigger to activate the record. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

### Triggers

Type: **Class** Derived From:

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **trigger** Trigger  [1..\*] | 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. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 1 ] |

### knowledgeDocument

Type: **Class** Derived From:

A knowledge document is an instance of a CDS knowledge artifact such as a rule, an order set, or a documentation template.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **actionGroup** ActionGroup  [1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **behaviors** Behaviors  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **conditions** Conditions  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |
| **metadata** Metadata  [1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 1 ] |
| **triggers** Triggers  [0..1] | 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. | *Default:*    [maxOccurs = 1 ]  [minOccurs = 0 ] |

#### expressions

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **def** ExpressionDef  [0..\*] |  | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

#### externalData

Type: **Class** Derived From:

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.

Elements/Attributes

| **Element/Attribute** | **Notes** | **Constraints and tags** |
| --- | --- | --- |
| **def** ExpressionDef  [0..\*] | The named expression used to retrieve external data. For instance, an expression to retrieve patient demographic data or a set of SNOMED-CT codes subsumed by another SNOMED-CT code from a terminology server. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |
| **parameter** ParameterDef  [0..\*] | The parameter element define a parameters for the knowledge document. Parameters are expected to be provided by the caller when an evaluation is performed. Parameters can be referenced within any expression using a ParameterRef expression.  For instance, one may define a MonthThreshold parameter, and use this parameter to determine whether the span of time that has elapsed since the last A1C has been performed warrants the elicitation of a reminder. | *Default:*    [maxOccurs = unbounded ]  [minOccurs = 0 ] |

# Appendix A – Referenced documents

The reference documents specific to the CDS Knowledge Artifact are presented in the table below. This specification 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 specification is developed and balloted.

| **Reference Name** | **Location** | **How reference was used in development of CDS Knowledge Artifact Specification** |
| --- | --- | --- |
| 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 specification |
| 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 specification as one of the schemas that has been harmonized within the new CDS Knowledge Artifact schema. |
| 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

# appendix b – acronyms

The following acronyms are referenced in this specification:

| **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 specification

# 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

# Appendix D – HeD Schema Framework

## 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 specification, and as such assumes a level of technical familiarity with software development concepts in general.

### 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.

### 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. |

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.

### 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 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.

## Components

The following diagram depicts the major functional components provided by the HeD Engine assembly:



### 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. |
| 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.

### 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.

### 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.

### 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.

## 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 |
| 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.

### 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. |

### 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.

### 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.

### 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.

## 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.

### 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.

### 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"/>

### 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.

## 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.

### 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.

### 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.

#### 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 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.

#### 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.

#### 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.

#### 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.

#### 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 |
| 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 |

#### 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 |

#### 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] |

#### 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] |
| Power | BinaryOperator[opPower] |

#### 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>) |

#### 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 |

#### 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) |
| Today | Today |
| Now | Not Supported |
| Date | TODO |

### 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">

<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>

#### 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>

#### 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" />

<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.

#### 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.

#### 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.

### 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.

### 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. |

#### 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.

#### 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">

<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>

#### 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..

# ****Appendix E – Examples****

## 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

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">

<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>

<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" />

<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" />

<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" />

<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>

## 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"

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>

</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>

<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">

<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>

<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>

</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>

## 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

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" />

<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

</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="???"

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>

</vmr:clinicalStatement>

</vmr:relatedClinicalStatement>

<!-- 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>

<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>

## HeartFailureAdmissionToMedSurgOrderSet Example

<?xml version="1.0" encoding="UTF-8"?>

<?schematron-schema href="../main/schematron/knowledgeartifact.sch"?>

<?schematron-schema href="../main/schematron/ordersets.sch"?>

<!-- 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

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>

<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"

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. -->

<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>

</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>

<!-- 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>

<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" />

<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:CommunicationProposal">

<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

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

references the same orderable. -->

<behaviors>

<behavior xsi:type="GroupOrganizationBehavior" value="SentenceGroup" />

</behaviors>

<representedConcepts>

<concept codeSystem="2.16.840.1.113883.6.88" code="1998"

codeSystemName="RxNorm">

<dt:displayName value="captopril" />

</concept>

</representedConcepts>

<subElements>

<simpleAction xsi:type="CreateAction">

<textEquivalent

value="captopril 6.25 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="6.25" unit="mg" />

<high value="6.25" 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>

<!-- 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. -->

<!-- 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">

<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" />

<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"

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">

<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

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">

<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="??">

<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"

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="??">

<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)

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>

</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" />

</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"

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"

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>