Date: July 2011

OBJECT MANAGEMENT GROUP

Clinical Decision Support Service (CDSS)

Version 1.0

OMG Document Number:	formal/2011-07-01
Standard Document URL:	http://www.omg.org/spec/CDSS/1.0/
Associated File(s)*:	http://www.omg.org/spec/CDSS/20101201
Normative:	http://www.omg.org/spec/CDSS/20101201/dss.wsdl
	http://www.omg.org/spec/CDSS/20101201/dssBaseComponents.wsdl
	http://www.omg.org/spec/CDSS/20101201/dssEvaluate.wsdl
	http://www.omg.org/spec/CDSS/20101201/dss_xmi.xml
	http://www.omg.org/spec/CDSS/20101201/datatypes.xsd
	http://www.omg.org/spec/CDSS/20101201/datatypes-base.xsd
	http://www.omg.org/spec/CDSS/20101201/INFRAS~1.XSD
	http://www.omg.org/spec/CDSS/20101201/NarrativeBlock.xsd
	http://www.omg.org/spec/CDSS/20101201/OmgDssSchema.xsd
	http://www.omg.org/spec/CDSS/20101201/OmgDssTraitSchema.xsd
	http://www.omg.org/spec/CDSS/20101201/POCD_MT000040.xsd
	http://www.omg.org/spec/CDSS/20101201/voc.xsd
Informative:	http://www.omg.org/spec/CDSS/20101201/DSS.EAP
	http://www.omg.org/spec/CDSS/20101201/DSS_XML_PSM.EAP
	http://www.omg.org/spec/CDSS/20101201/DSS_XML_PSM_XMI.xml

---

* original file(s): dtc/2010-12-08.zip (machine-consumable files)

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.
IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , IMM™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG Systems Modeling Language (OMG SysML)™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG’s Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (*http://www.omg.org/technology/agreement.htm*).

Table of Contents

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications.

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM).

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications.

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

Note – Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

1 Scope

This specification specifies a platform-independent model (PIM) and a platform-specific model (PSM) for XML Web services that define the capabilities and interfaces of a DSS. These models fulfill the requirements specified in the normative sections of the HL7 DSS SFM while improving the simplicity and functional completeness of the service interface.

2 Conformance

Conformance to this specification happens by way of conformance to profiles, which are detailed later in this specification ("see 6.11, Profiles and Semantic Requirements Specified as a Part of this Specification"). Conformance Profiles consist of at least one Functional Profile and at least one Semantic Profile.

Functional Profiles are named subsets of the overall set of operations defined within the specification that provide a cohesive set of functionality that makes sense from the service client's perspective. A reflection style interface is also made available by DSS instances that will identify which profiles they support at any given point in time.

Similarly, Semantic Profiles define the semantics that are used by a DSS. This allows the same behavioral interface to be used with different content models.

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, ***http://www.w3.org/TR/wsdl***

Simple Object Access Protocol (SOAP) Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation 27 April 2007, ***http://www.w3.org/TR/soap12-part1/***

XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001, ***http://www.w3.org/TR/xmlschema-2***

XML Path Language (XPath) 1.0, W3C Recommendation 16 November 1999, ***http://www.w3.org/TR/xpath***

ISO/IEC Standard 19757-3:2006, Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron, Edition 1, 1 June 2006, ***http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip***

Uniform Resource Locators (URL): A Syntax for the Expression of Access Information of Objects on the Network, 21 March 1994, ***http://www.w3.org/Addressing/URL/url-spec.txt***

HL7 Version 3 Normative Standard, 2008 Edition, ***http://www.hl7.org/memonly/downloads/v3edition.cfm#V32008***

HL7 Decision Support Service, Release 1, Service Functional Model Specification, ***http://www.hl7.org/v3ballot2009sep/html/infrastructure/dss/dss.htm***

ISO Standard 3166-1, Country Codes, ***http://www.iso.org/iso/english_country_names_and_code_elements***

ISO Standard 639-1, Codes for the Representation of Names of Languages,
http://www.loc.gov/standards/iso639-2/php/code_list.php

Service Specifications Framework (SSF), Healthcare Services Specification Project (HSSP), ***http://hssp.wikispaces.com***

4 Acronyms

There are a number of acronyms used in this document, and in standards or other documents related to this specification. The following is a brief list of what the most common acronyms stand for.

Acronym	**Full Name**
ANSI	American National Standards Institute (U.S.A.)
DRG	Data requirement group
DRI	Data requirement item
DSS	Decision Support Service
HITSP	Health Information Technology Standards Panel (U.S.A.) of ANSI
HL7	Health Level 7
HSSP	Healthcare Services Specification Project
IHE	Integrating the Healthcare Enterprise
ISO	International Organization for Standardization
KM	Knowledge module
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
RIM	Reference Information Model defined by HL7
RFP	Request for Proposal
RM-ODP	Reference Model of Open Distributed Processing defined by ISO
SDO	Standards Development Organization
SFM	Service Functional Model
UML	Unified Modeling Language

5 Acknowledgements

The following companies submitted and/or supported this specification.

Submitters

- 88Solutions
- Religent, Inc. (healthcare business now spun off into Clinica, Inc.)
- Software Partners, LLC

Supporters

- dbMotion
- Healthcare Services Specification Project (HSSP)
- InferMed

6 DSS Platform Independent Model

6.1 General

The Platform Independent Model (PIM) for Decision Support Service (DSS) represents a platform-independent definition of the DSS interfaces.

The PIM is defined in the accompanying normative UML model, represented as an XMI file. The source Enterprise Architect .EAP model is also provided on a non-normative, reference basis. Elements of this model are presented in this clause to clarify and provide guidance on this model.

Note that the models provided below are extracts from the accompanying normative UML model.

6.2 Foundational Model Elements

This sub clause defines foundational model elements used by various operations in the DSS.

6.2.1 Described Data Object

The Described Data Object (DescribedDO) is an abstract class that is accompanied by a String description and name. This class is defined in the common package. Note that for the diagram below, as well as for all other model fragment diagrams that follow, the “class” referenced in the top left corner of the diagram (e.g., “class common” in the diagram below) actually refers to the package in which the class resides (e.g., the “common” package for DescribedDO).

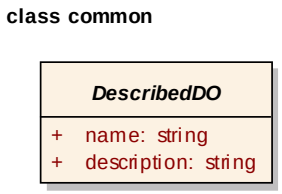


Figure 6.1- Model for Described Data Object

6.2.2 Scoping Entity

A Scoping Entity (ScopingEntity) is a class that extends the Described Data Object and represents an entity that scopes business objects within a DSS. This class is defined in the metadata.scopingentity package.

The Scoping Entity is identified by a String “id.” The intent of this id is to allow scoping entities to be identified uniquely, so that business objects can be identified in a globally unique manner as long as business object identifiers are unique within a scoping entity.

The “id” must start with lowercase English representations of one of the top-level Internet domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166-1 (see “Normative References”). Subsequently, the “id” must start by defining the domain name that is associated with the scoping entity (e.g., “com.clinica,” “com.dbmotion,” “edu.duke,” “org.hl7”). Subsequent identification within the domain

associated with the scoping entity, if any, may be specified as is appropriate for the internal naming conventions by the scoping entity. Also, Scoping Entities may have a hierarchical structure described by the existence of parent and children Scoping Entities.

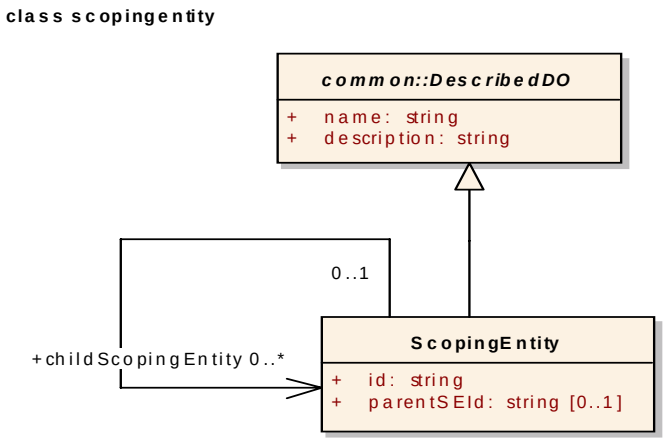


Figure 6.2 - Model for Scoping Entity

6.2.3 Entity Identifier and Interaction Identifier

The Entity Identifier (EntityIdentifier) is used to identify business objects within a DSS. The Entity Identifier consists of the "id" of its Scoping Entity, a String "businessId," and a String "version." The Entity Identifier (the combination of scopingEntityId + businessId + version) must be globally unique. The only restriction on the version relates to the versioning of Knowledge Modules, which is discussed later in 6.4.1.2, Knowledge Module Version. This class is defined in the common package.

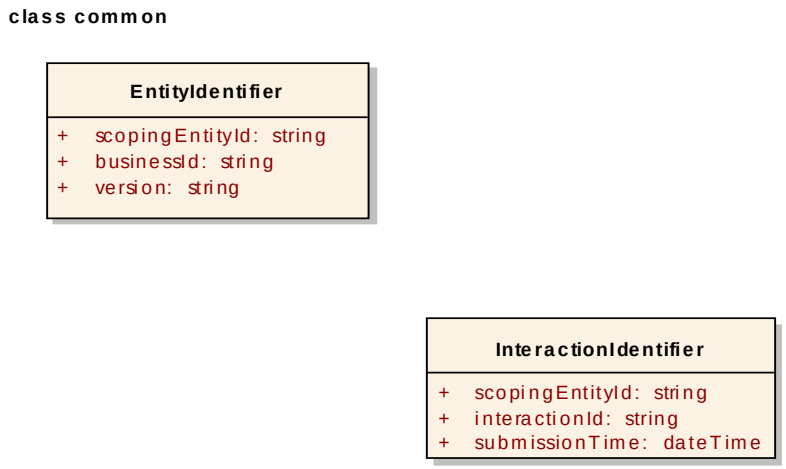


Figure 6.3 - Model for Entity Identifier and Interaction Identifier

6.2.4 Item Identifier

The Item Identifier (ItemIdentifier) is used to identify individual items that constitute subunits of business objects within a DSS. The Item Identifier consists of the Entity Identifier of its containing entity, as well as a String "itemId." The "itemId" must be unique within the scope of the containing entity, and the complete ItemIdentifier (i.e., combination of containingEntityId + itemId) must be globally unique. This class is defined in the common package.

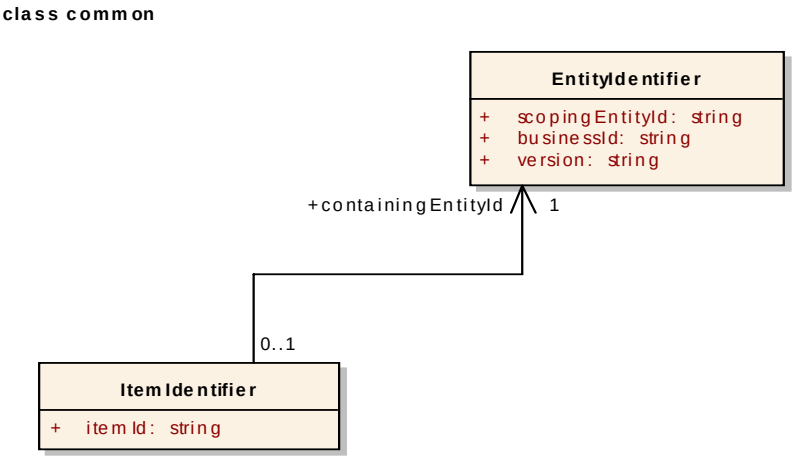


Figure 6.4 - Model for Item Identifier

6.2.5 Scoped Data Object

The Scoped Data Object (ScopedDO) is an extension of the Described Data Object that represents the business object scoped by a scoping entity. This class includes a description, name, and an Entity Identifier. This class is defined in the common package.

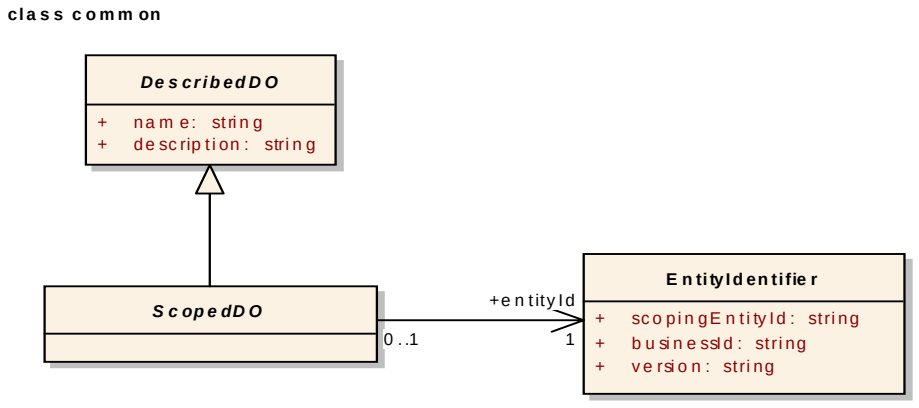


Figure 6.5 - Model for Scoped Data Object

6.3 Metadata Model Elements

This sub clause describes the model elements related to service metadata.

6.3.1 Service Profile

The Service Profile (ServiceProfile) is an abstract class that represents a service profile. This class is defined in the metadata.profile package. Note that many of the descriptions and explanations that follow have been adapted from Chapter 6 of the HL7 DSS SFM.

6.3.1.1 Overview of Service Profiles

This specification is designed as a generic service framework that can be adapted in various ways to meet clients' clinical decision support needs. While this flexibility is desirable, too much flexibility could make it more difficult to implement a DSS and/or to achieve plug-and-play interoperability among multiple DSSs. The specification of profiles allows the service to be constrained to the degree required for implementation and interoperability.

Of note, it is envisaged that many profiles will be defined after the adoption of this specification. Some of these profiles may be specified as formal, balloted profiles defined by standards development organizations such as HL7 and OMG, while other profiles may be specified as informal profiles defined by individual vendors, institutions, geographic regions, and other domains.

6.3.1.2 Profile Types

Table 6.1 summarizes the types of profiles that may be specified.

Table 6.1 - Types of profiles that may be specified for a DSS

Profile Type	Description
Functional profile	Specifies the list of supported service operations.
Semantic profile	Specifies that all knowledge modules hosted by the service fulfill a specified set of semantic requirements (described in 6.4.2, Semantic Requirement).
Conformance profile	Specifies a list of one or more supported functional profiles and one or more supported semantic profiles.

6.3.1.3 Service Profile Model

The Service Profile's class model is shown below. In addition, a model that groups profiles by type, used in the metadata discovery interface, is provided below for reference.

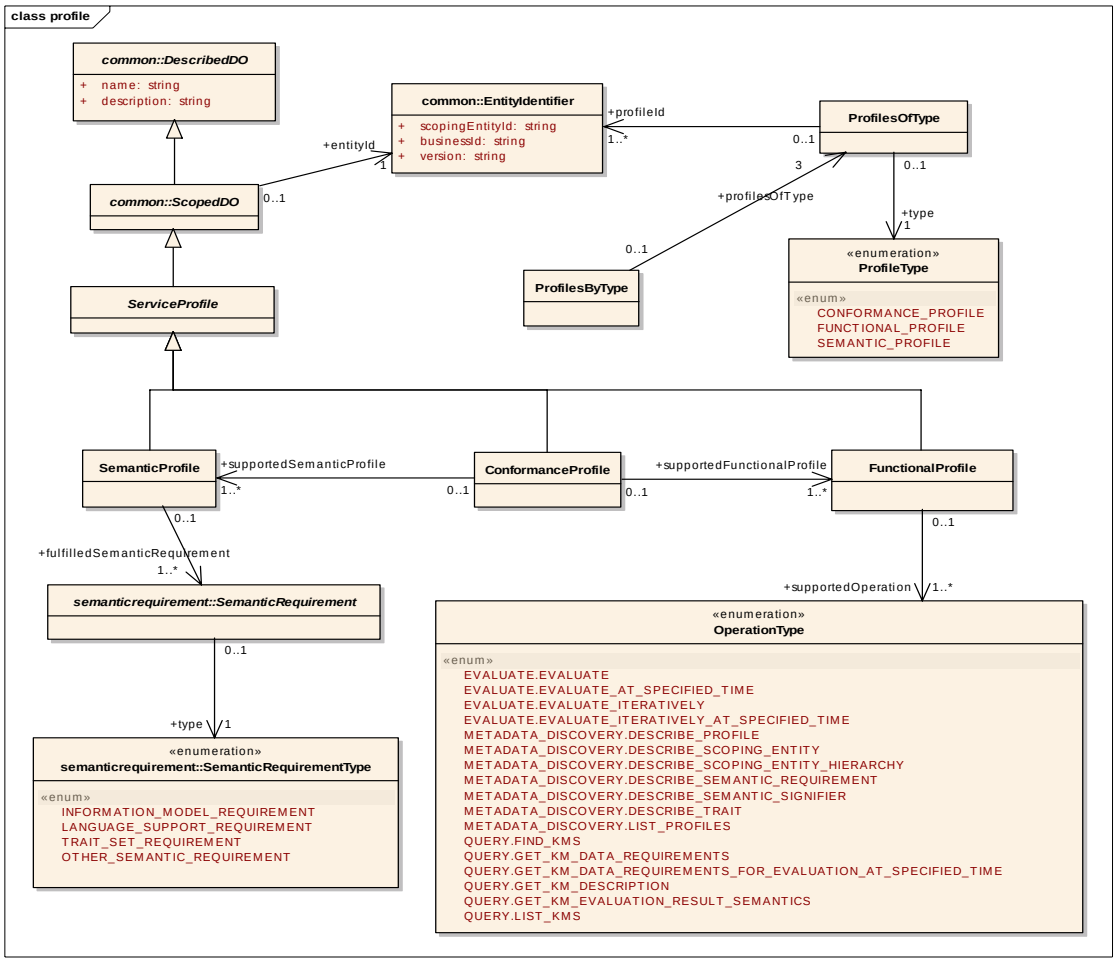


Figure 6.6 - Model for Service Profile

6.3.2 Semantic Signifier and Related Classes

A Semantic Signifier (SemanticSignifier) is a class that represents an information model. This class is defined in the metadata.semanticsignifier package. Note that many of the descriptions and explanations that follow have been adapted from 2.3.3 of the HL7 DSS SFM.

6.3.2.1 Overview of Semantic Signifiers

The HSSP defines semantic signifiers as identifiers of information constructs that specify the structure and meaning of data. Semantic signifiers may identify standardized information constructs from HL7 (e.g., an HL7 version 3 Refined Message Information Model [RMIM]), standardized information constructs from a standards development organization other than HL7 (e.g., a DICOM image format), or non-standard local information constructs (e.g., Health System A's laboratory data exchange format).

6.3.2.2 Use of Semantic Signifier within DSS

In this specification, semantic signifiers are used to specify the semantics:

1. by which data should be provided to the DSS for evaluating patients using a knowledge module.
2. by which the query conditions for knowledge module data requirements are expressed by the DSS.
3. by which patient evaluation results will be returned by the DSS.
4. related to the traits and trait search criteria of DSS knowledge modules; and
5. by which warnings are provided related to knowledge module evaluations.

6.3.2.3 Semantic Signifier Model

The Semantic Signifier Data Object model is shown below. As noted, a semantic signifier is a Scoped Entity with a computable information model definition (e.g., an XML Schema Definition [XSD]) and zero or more computable integrity ruleset definitions (e.g., Schematrons) and an optional narrative model restriction guide. Currently, these definitions are made accessible via URLs. For the XML Web Service PSM defined in this specification, the XSDComputableDefinition defined below shall be used, consisting of a Uniform Resource Locator (URL) to a single XSD, URLs to zero or more Schematrons, an optional URL to a narrative model restriction guide, and a specification of the global element that serves as the root element of the information model. Note that an XSD used in this context must have the root element defined as a global element so that it can be directly used for automated instance validation. See 3, Normative References for normative references to the XSD, Schematron, and URL standards.

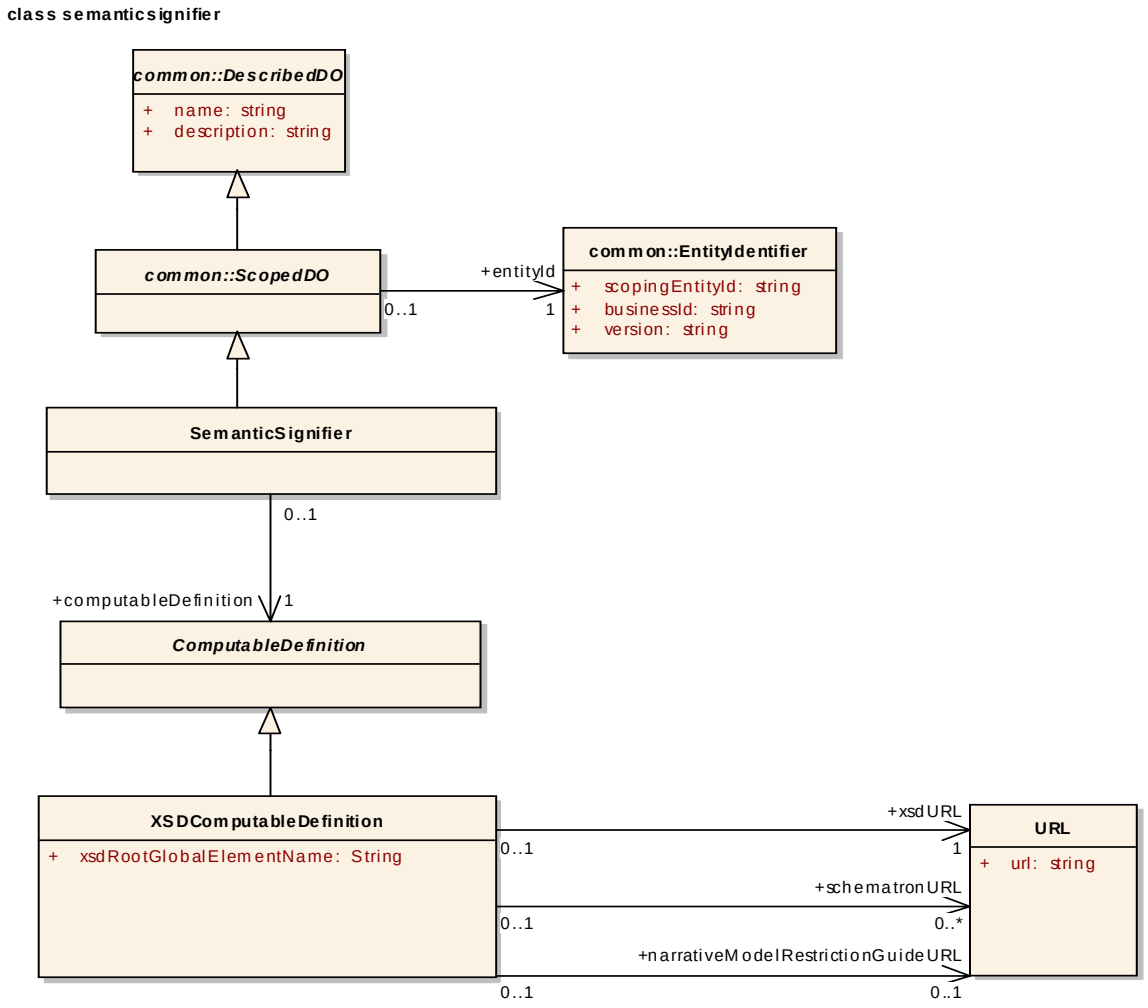


Figure 6.7 - Model for Semantic Signifier

6.3.2.4 Convention for Referring to HL7 Version 3 Semantic Signifiers

For referring to HL7 version 3 semantic signifiers, the use of the following conventions are recommended:

- For schemas that reside in the processable\coreschemas section of the HL7 version 3 standard (2008 version 3 normative standard referenced here may be downloaded from http://www.hl7.org/documentcenter/private/standards/v3/edition2008/Edition2008_StPub.zip), use the scoping entity identifier of org.hl7.v3.coreschemas.
- For schemas that reside in the processable\multicacheschemas section of the HL7 version 3 standard, use the scoping entity identifier of org.hl7.v3.multicacheschemas.
- For schemas that reside in the infrastructure section of the HL7 version 3 standard, use org.hl7.v3 as the scoping entity identifier, followed by the package name within this infrastructure section (e.g., org.hl7.v3.cda, org.hl7.v3.cda.coreschemas).

- For the business identifier, use the identifier assigned by HL7 version 3 and reflected in the name of the schema (e.g., COCT_HD010000UV01, FICR_IN310201UV02), followed by a period and then by the root element or complex type to be used from the schema. For example, if the complex type of interest with the schema COCT_HD010000UV01 is named COCT_HD010000UV01.Encounter, then make the business identifier COCT_HD010000UV01.COCT_HD010000UV01.Encounter.
- For the version, use "1.0" since HL7 version 3 schemas are assigned different identifiers when modified (e.g., MODELX, MODELXUV, MODELXUV01, etc.).
- In the case of XML schemas, almost all HL7 version 3 schemas currently define complex types, but do not define elements that can be used for the purposes of automated instance validation. Therefore, it is recommended that DSS providers wishing to use HL7 version 3 XML schemas provide clients with a separate XML schema that defines its focal, root element using the name and semantics of the HL7 complex type. For example, org.hl7.v3.multicacheschemas' COCT_HD010000UV01 schema defines a complex type named COCT_HD010000UV01.Encounter. To use this concept, it is recommended that a DSS provider do the following:
    - Create or otherwise obtain an XML schema in which an element named COCT_HD010000UV01.Encounter is defined, whose type is the COCT_HD010000UV01.Encounter complex type defined in the HL7 version 3 schema.
    - Identify this schema as org.hl7.v3.multicacheschemas' COCT_HD010000UV01 schema, with a focal element of COCT_HD010000UV01.Encounter. As described above, the scoping entity identifier would be org.hl7.v3.multicacheschemas, and the business identifier would be COCT_HD010000UV01.COCT_HD010000UV01.Encounter.
    - An example using this convention follows:

Schema	COCT_HD010000UV01
+ Complex Type	COCT_HD010000UV01.Encounter
= Business Identifier	COCT_HD010000UV01.COCT_HD010000UV01.Encounter

6.4 Knowledge Module Model Elements

This section describes the model elements related to knowledge modules (KMs).

6.4.1 Knowledge Module Description

The Knowledge Module Description (KMDescription) and Extended Knowledge Module Description (ExtendedKMDescription) provide core meta-data regarding a DSS knowledge module. The class is modeled in the query.km package. The model is provided below, and relevant aspects of this model are described thereafter.

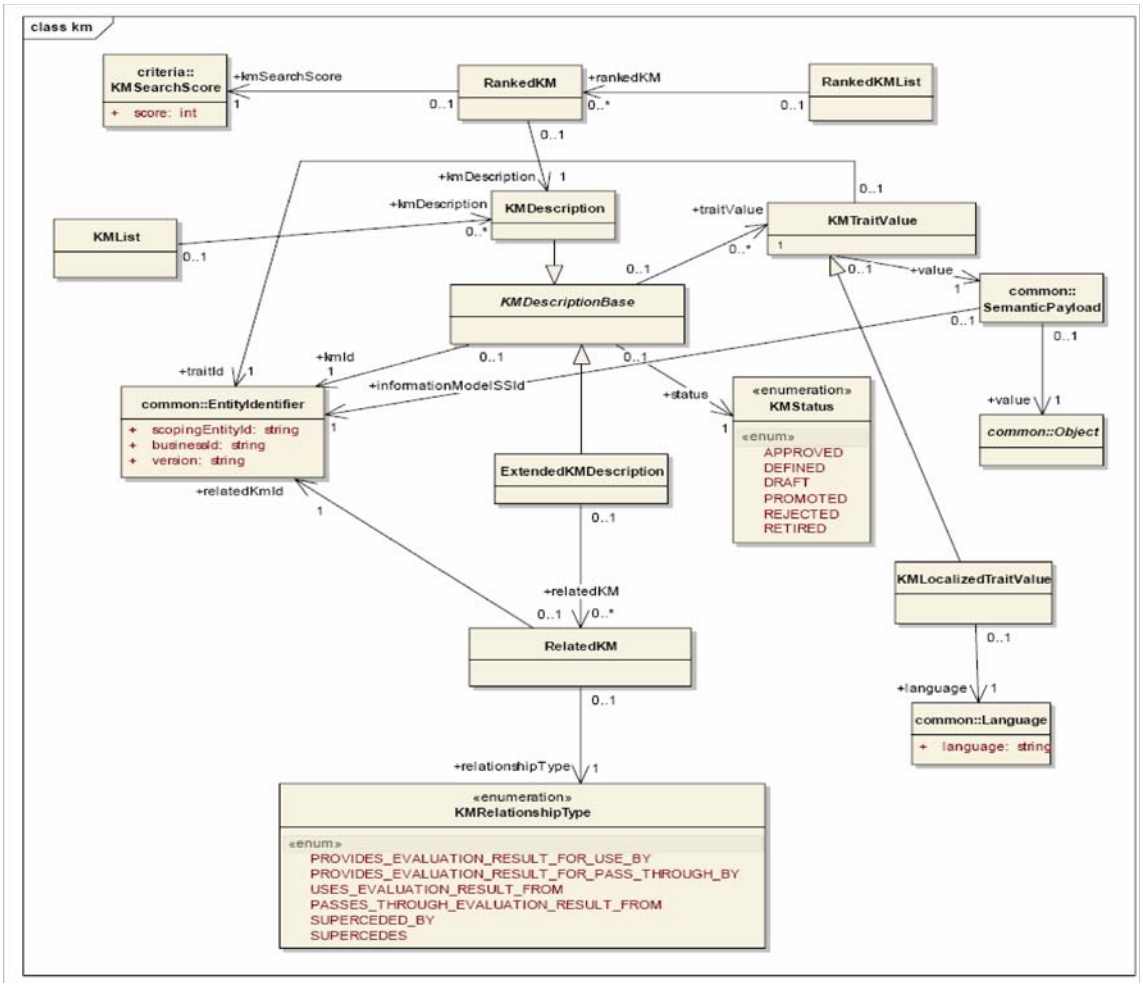


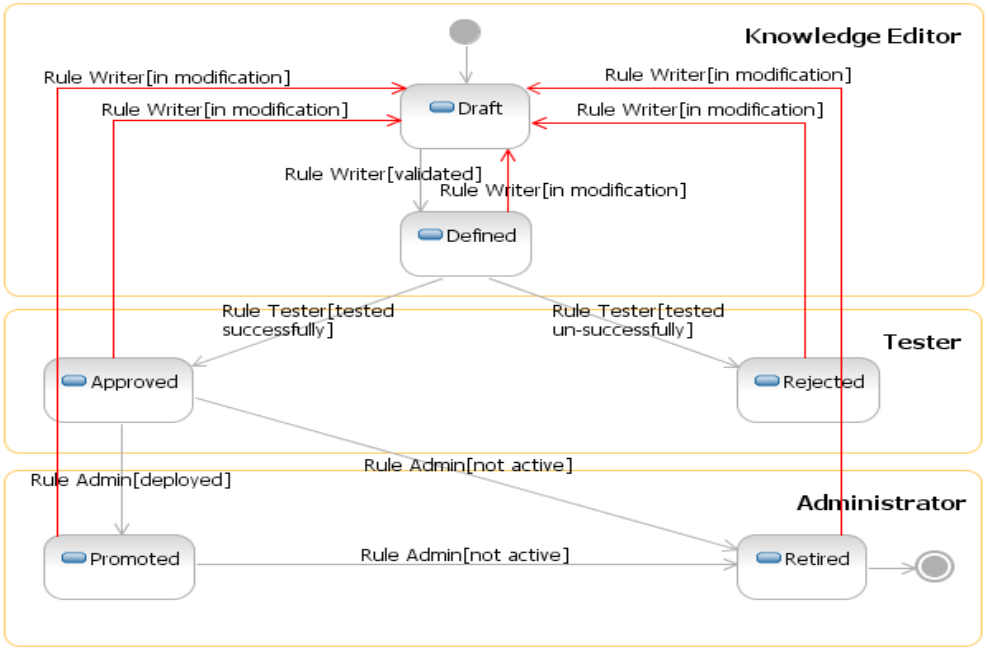
Figure 6.8 - Model for Knowledge Module Description

6.4.1.1 Knowledge Module Status

The allowed values and definitions of a KM status are as follows:

- DRAFT - the KM has been created and can be modified.
- DEFINED - the KM has been defined and is currently in unit test.
- REJECTED - the KM has been tested un-successfully.
- APPROVED - the KM has been tested successfully and can be deployed.
- PROMOTED - the KM has been deployed on a production platform.
- RETIRED - the KM was deployed or approved on a production platform but is no longer active.

The accompanying lifecycle diagram is as follows. Note that the term “rule” in the diagram should be considered to be equivalent to the concept of “KM” in the rest of this specification.



Furthermore, the following are accompanying guidelines for the management of KM status. These guidelines are to be considered normative.

- When a KM is created its status is "DRAFT." As long as the KM is in this status, every change made does not affect the KM version.
- Every time the KM status changes to "DRAFT," a new KM version is created, i.e., the lifecycle is restarted.
- Once a KM is "PROMOTED" the user cannot update it. He needs to create a new KM version and restart the life cycle up to "APPROVED."
- A "RETIRED" KM should no longer be used (e.g., due to changes in underlying clinical guidelines or the availability of an improved version of the KM). A "RETIRED" KM may still be usable, but continued support is not guaranteed. The expectation is that a "RETIRED" KM will be replaced by an improved "PROMOTED" KM. However, this is not guaranteed, as in the case when a KM must be quickly retired due to the emergence of new evidence that a drug that was previously recommended should no longer be prescribed due to serious side effects, but a replacement KM cannot be developed, tested, and promoted before the original KM is retired.
- Whether a given DSS client is allowed to search for and/or use KMs of a given status is outside of the scope of this specification and is up to the DSS provider. For example, a DSS provider may make "DRAFT" KMs available only to internal developers, or a DSS provider may allow clients to continue using "RETIRED" KMs as long as the clients are aware that continued support for such KMs cannot be guaranteed indefinitely. Similarly, KMs with a status other than "PROMOTED" may be searchable by internal developers using the DSS interface but not searchable by a typical client.
- Similar to the above consideration, within KMs of the same status, it is outside of the scope of this specification which KMs are visible to and usable by a DSS's clients. For example, a DSS provider may decide to segment KMs into sets of knowledge that require different levels of licenses to access.

6.4.1.2 Knowledge Module Version

A component of the Entity Identifier for a knowledge module is the version. The following are defined as normative requirements for the version:

- As noted above, a new KM version is created each time that a new KM is created or its status is changed to "DRAFT."
- The KM version shall take the following form: [Major Version Number].[Minor Version Number].[Revision Number] (e.g., 1.0.0).
- Conceptually, the different components of the KM version can be understood as follows:
    - Major Version Number - reflects major changes in the KM's run-time interface and/or the underlying clinical logic. Starts with 1, and increments up by 1.
    - Minor Version Number - reflects minor changes in the KM's run-time interface and/or the underlying clinical logic. Starts with 0 within a major version, and increments up by 1.
    - Revision Number - reflects revisions that do not make any significant changes to the KM's run-time interface or the underlying clinical logic. Starts with 0 within the combination of a major version and minor version, and increments up by 1.
    - Which version part to change is up to the discretion of the DSS provider under the conceptual framework above. However, if a KM change involves one of the aspects specified in Table 6.2, then the version change must adhere to the versioning approach specified in the table.

Table 6.2 - Possible version number changes given changes to a KM

	Change May Be Reflected In Version Part*:				
KM Change	**Maj**	**Min**	**Rev**	**None**	**Comments**
Modifications to name or description of a Described Object			X	X	Version change not required, but may wish to update revision number to support version history.
Traits					
trait value changed	X	X	X	X	Up to DSS provider discretion. Expected to usually result in no change in version or change in revision number.
Data Requirements					
Data Requirement Group					
Add	X				
Delete	X	X			Providing no longer required data should not cause run-time interaction to fail.
Data Requirement Item					
Add	X				
Delete	X	X			Providing no longer required data should not cause run-time interaction to fail.
Alternative Information Model					

Table 6.2 - Possible version number changes given changes to a KM

Add	X	X			
Delete	X				Data requirement item is still required, but consumer may no longer be able to provide the required data.
Update					
information model (SS*)	X				
query model (SS*)	X				
Query	X	X			Includes changes in the use of Consumer Provided Query Parameters
Decision Logic					
Change	X	X	X		Up to DSS provider discretion
Evaluation Results					
Add	X	X			Up to DSS provider discretion
Delete	X				
update information model (SS*)	X				
Consumer Provided Query Parameters					
Add	X				
Delete	X	X			Up to DSS provider discretion
update type	X				
Fulfilled Semantic Requirements					
Add	X	X			Up to DSS provider discretion
Delete	X	X			Up to DSS provider discretion

*Maj = Major Version Number; Min = Minor Version Number; Rev = Revision Number; None = No change in version. SS = semantic signifier.

The following are valid ways of specifying the version number of a knowledge module to use in the Evaluation operations (see 6.10, Evaluation Interface):

- The specific version number (e.g., 2.1.0) - This will result in the evaluation of version 2.1.0.
- The specific major and minor version, with * as the revision number (e.g., 2.1.*) - This will result in the evaluation of the highest revision with the specified major and minor version number (e.g., 2.1.0, 2.1.1, or 2.1.2, depending on the latest available revision).
- The specific major version, with * as the minor version and * as the revision number (e.g., 2.*.*) - This will result in the evaluation of the highest minor version, and the highest revision within that minor version (e.g., if the highest minor version within major version 2 is 3, and the highest revision within version 2.3 is revision 1, then 2.3.1 would be used).

6.4.1.3 Knowledge Module Relationship

A KM relationship may be of the following types:

- USES_EVALUATION_RESULT_FROM - The current KM uses one or more of the evaluation results from the related KM as an evaluation input.
- PROVIDES_EVALUATION_RESULT_FOR_USE_BY - The current KM provides one or more of its evaluation results to the related KM for usage as an evaluation input.
- PASSES_THROUGH_EVALUATION_RESULT_FROM - The current KM passes through to the consumer one or more of the evaluation results obtained from the related KM.
- PROVIDES_EVALUATION_RESULT_FOR_PASS_THROUGH_BY - The current KM provides one or more of its evaluation results to the related KM for passing the evaluation result through to the consumer.
- SUPERCEDED_BY - The current KM was superceded by the related KM. That is, the related KM should be used instead of the current KM if possible.
- SUPERCEDES - The current KM supercedes the related KM. That is, the current KM should be used instead of the related KM if possible.

6.4.1.4 Knowledge Module Traits

A KM may possess specified traits. These traits are described in detail in 6.4.3, Trait.

6.4.2 Semantic Requirement

A Semantic Requirement (SemanticRequirement) is an abstract class that represents a requirement placed on all knowledge modules within a DSS instance. A DSS semantic profile specifies which semantic requirements must be fulfilled by its KMs (6.3.1.2, Profile Types). The Semantic Requirement class is defined in the metadata.semanticrequirement package. Note that many of the descriptions and explanations that follow have been adapted from Sections 2.3.4.3.1 and 6.2 of the HL7 DSS SFM. Also note that this requirement used to be referred to as KM Requirements in the HL7 DSS SFM.

6.4.2.1 Semantic Requirement Types

The table below specifies the types of semantic requirements that may be specified.

Table 6.3 - Types of semantic requirements

Semantic Requirement Type	Description
Trait set requirement	Specifies the list of traits (see 6.4.3, Trait) that will or may be associated with the DSS's knowledge modules. Traits are identified by the identifier of the trait's scoping entity, the trait identifier, and the trait version. The requirement also specifies if the trait is required or optional for knowledge modules.

Table 6.3 - Types of semantic requirements

Information model requirement	The InformationModelRequirement specifies the information models that (a) can or (b) must be used by DSS knowledge modules claiming conformance to this requirement. This information model requirement consists of one or more of the following: (i) allowedDataRequirement - specifies the superset of data requirement models and associated query models that can be used. (ii) requiredDataRequirement - specifies the data requirement models and associated query models, if any, that must be used. (iii) allowedWarningModelSSId - specifies the superset of models that can be used by the service to provide warnings regarding evaluations. (iv) allowedEvaluationResultModelSSId - specifies the superset of evaluation result models that can be used. (v) requiredEvaluationResultModelSSId - specifies the evaluation result models that must be used.
Language support requirement	Specifies the languages that are supported by the DSS.
Other semantic requirement	Uses narrative to specify a semantic requirement for the DSS.

6.4.2.2 Language specification

Language shall be specified as either a 2-character ISO 639-1 language code or a combination of a 2-character ISO 639-1 language code and a 2-character ISO 3166-1 geographical code, concatenated with a hyphen. Example valid language specifications include: “en,” “en-US,” “en-GB,” and “fr.” ISO 639-1 codes are available at http://www.loc.gov/standards/iso639-2/php/English_list.php, and ISO 3166-1 codes are available at http://www.iso.org/iso/english_country_names_and_code_elements. See Clause 3 for normative references to these standards.

6.4.2.3 Semantic Requirement Model

The Semantic Requirement class model is shown below.

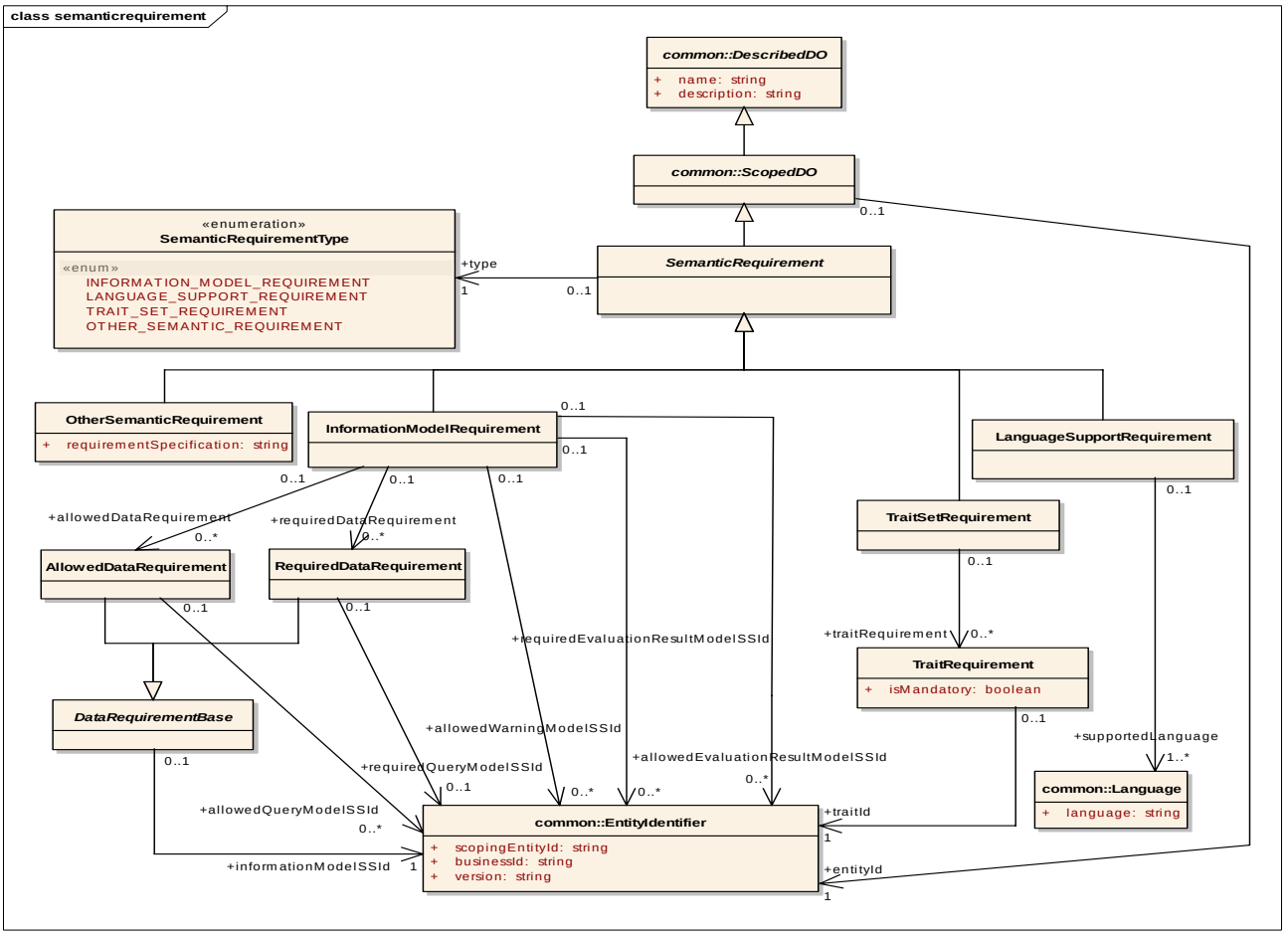


Figure 6.9 - Model for Semantic Requirement

6.4.2.4 Comprehensive Capture of Key Service Characteristics within Semantic Requirements and Semantic Profiles

In order to allow a DSS consumer to be able to fully understand the key characteristics of a DSS at the level of semantic profiles and constituent semantic requirements, the following are specified as mandatory requirements:

- All traits used within a DSS must be included within a TraitSetRequirement.
- All languages supported by a DSS must be included within a LanguageSupportRequirement.
- All evaluation result information models used within a DSS must be included within an allowedEvaluationResultModelSSId.
- All data requirement information models used within a DSS must be included within an AllowedDataRequirement.
- All warning information models used within a DSS must be included within an allowedWarningModelSSId.
- All semantic requirements within a DSS must be represented within a semantic profile.

6.4.3 Trait

A Trait is used in the context of this specification to provide metadata for KMs. This class is defined in the metadata.trait package. Traits can be used to search for KMs or to describe a given KM. Example traits include the last review date, steward organization, and keywords. The information models used to define these traits are specified using semantic signifiers. Also, a Trait may have Trait Criterion (TraitCriterion) objects that represent semantic signifier-identified information models that can be used to express query parameters for searching for KMs with specified trait values. The identifier of a trait criterion must be unique within a trait (i.e., the identifier of a trait criterion consists of the parent's trait EntityIdentifier plus an itemID that is unique within the scope of the parent trait's EntityIdentifier). If a trait is language-dependent (e.g., a descriptive text), then the trait value is provided in accordance with the client's language (see 6.4.2.2, Language specification regarding DSS support for languages, as well as 6.9, Query Interface and 6.10, Evaluation Interface regarding how individual DSS operations deal with languages).

The data models for the Trait, Trait Value, and Trait Criterion classes are shown below.

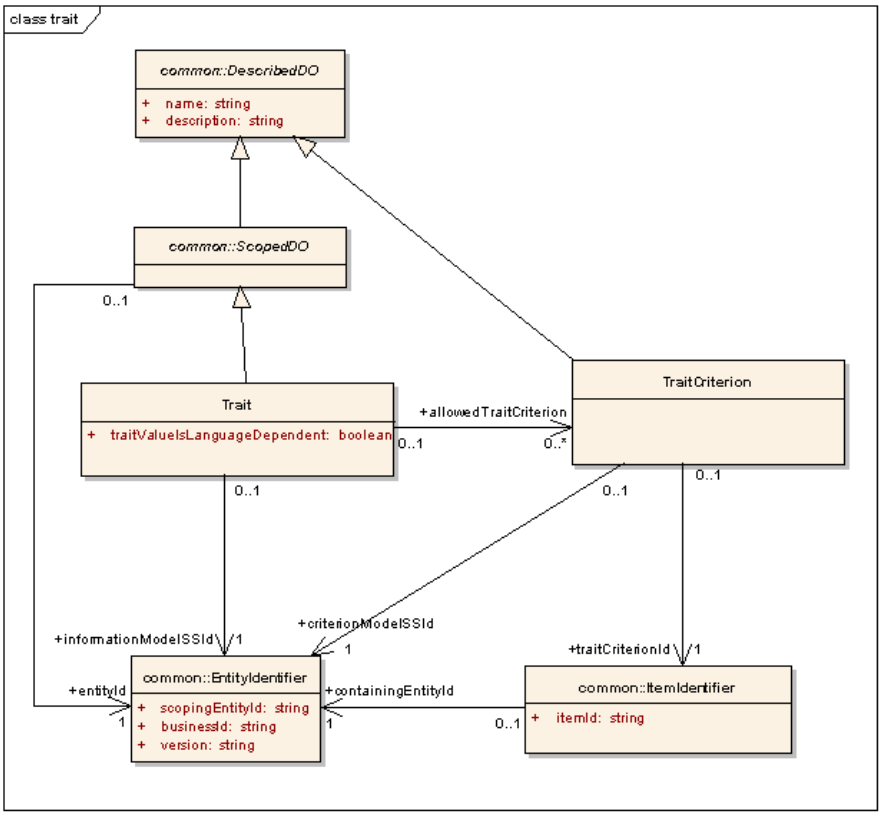


Figure 6.10 - Model for Trait

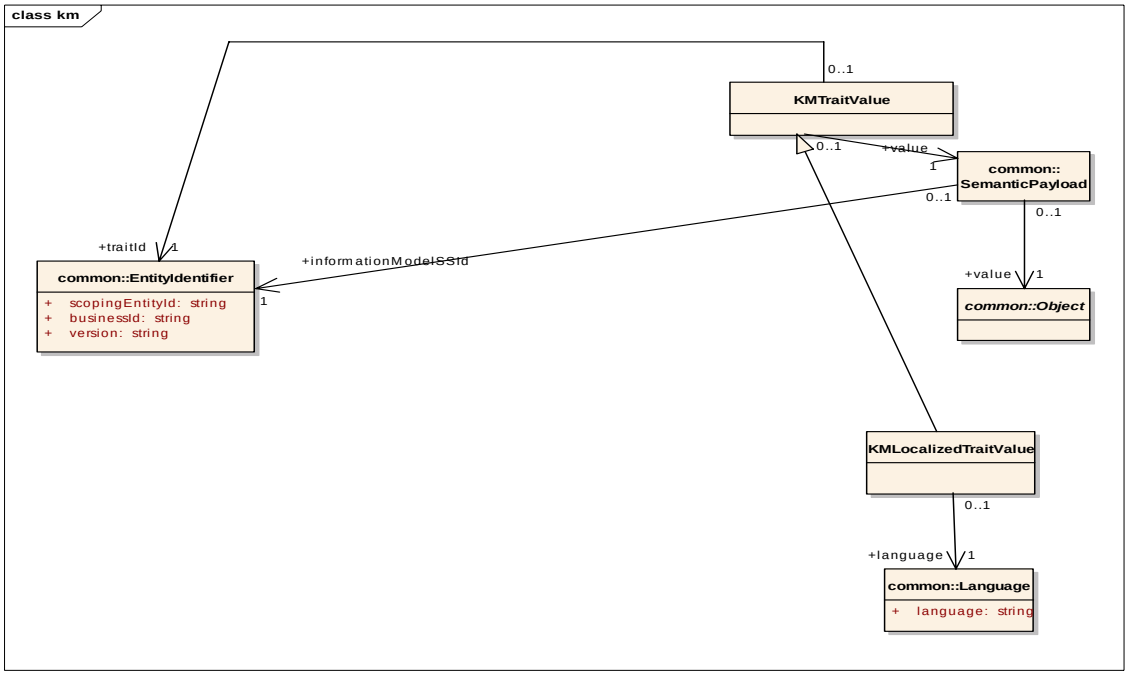


Figure 6.11 - Model for Trait Value

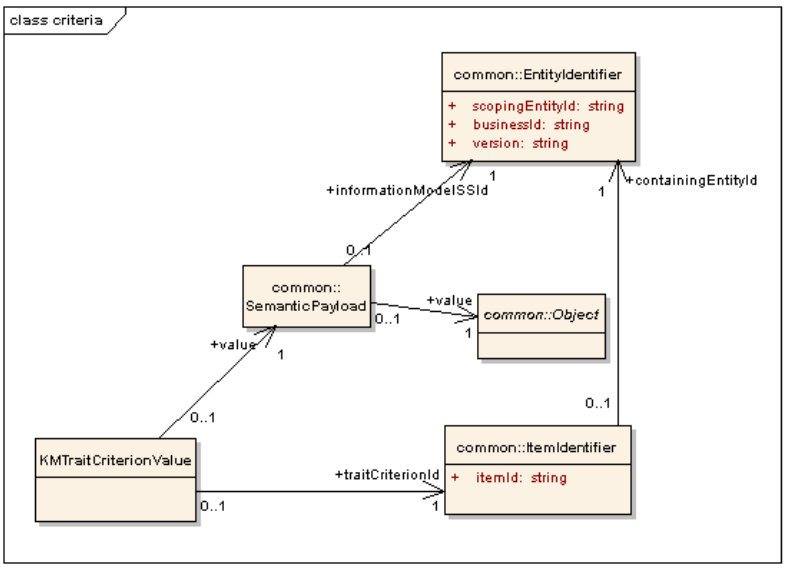


Figure 6.12 - Model for Trait Criterion

6.4.4 Knowledge Module Data Requirement Elements

KMs have data requirements for generating evaluation results. These models are provided in the query.km.dr and query.km.cpqp sections. These model elements are described below.

6.4.4.1 KM Data Requirement Item

The building block of KM data requirements are KM Data Requirement Items (KMDataRequirementItem class; model shown below). Through a semantic signifier, a KM data requirement item specifies how the data requirement item must be presented to the DSS. In addition, a KM data requirement item may optionally specify query parameters that should be used by the client to restrict the data submitted to the DSS. The information model used to define the query is identified by the query semantic signifier, and the information model-compliant query parameters are specified within the query attribute.

These KM data requirement items may also specify that certain query parameters should be specified by the client. In specifying the use of such consumer-provided query parameters (CPQPs), the identifier of the CPQP is provided along with an unambiguous specification of the path within the query model where the CPQP should be used to replace the placeholder data provided by the DSS. For the XML Web service PSM, this path shall be specified using XPath 1.0. Further details regarding CPQPs are provided in the next clause.

Of note, this specification expects patient data to be communicated using information models that utilize absolute date-times to note when a care activity occurred. Use of information models that do not utilize absolute date-times to note when a care activity occurred are outside of the scope of this specification.

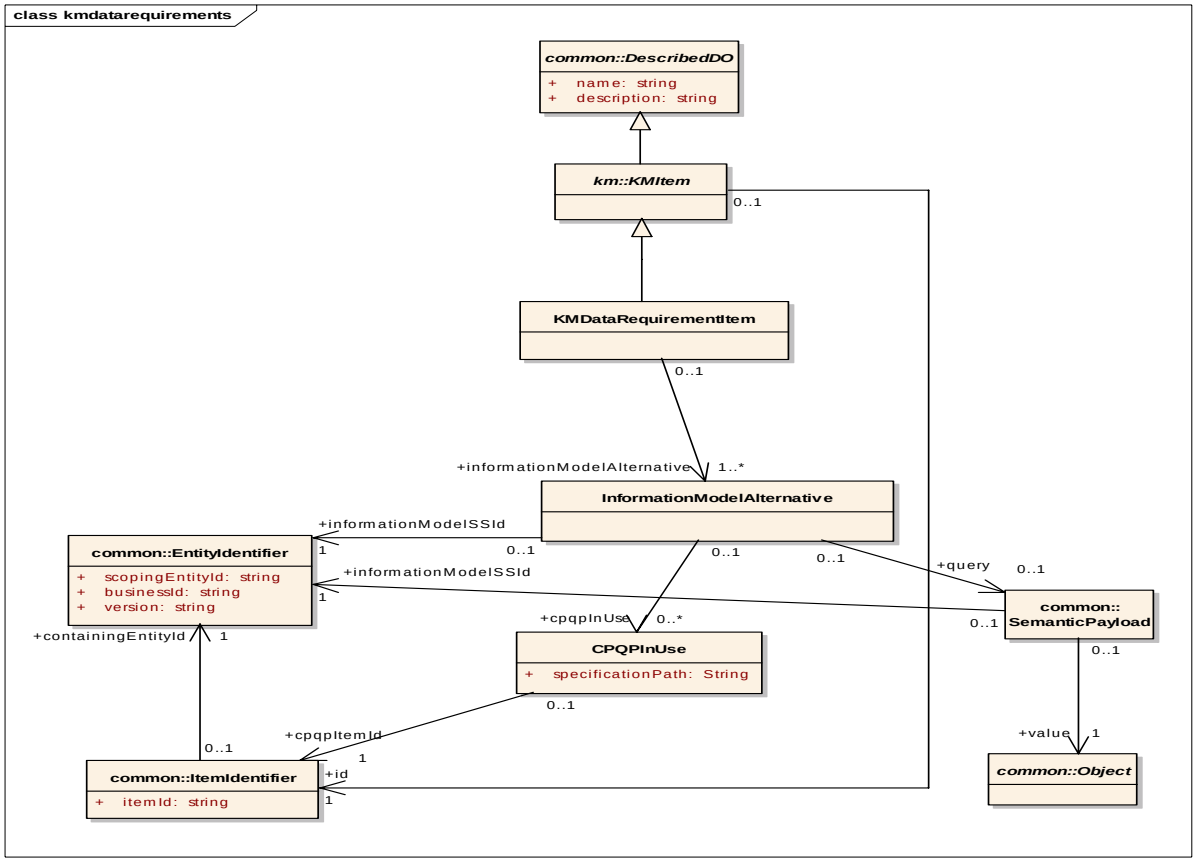


Figure 6.13 - Model for KM Data Requirement Item

6.4.4.2 Consumer-Provided Query Parameter (CPQP)

CPQPs may be necessary when a DSS KM has no way of knowing *a priori* what a certain query parameter value should be. This may be the case, for example, if a query parameter model involving a patient identifier is used, or if a query parameter model requires the specification of a specific encounter to analyze. The use of a CPQP allows a DSS to specify that certain query parameters within a KM data requirement item's query model should be specified by the client.

The model of the CPQP is provided below.

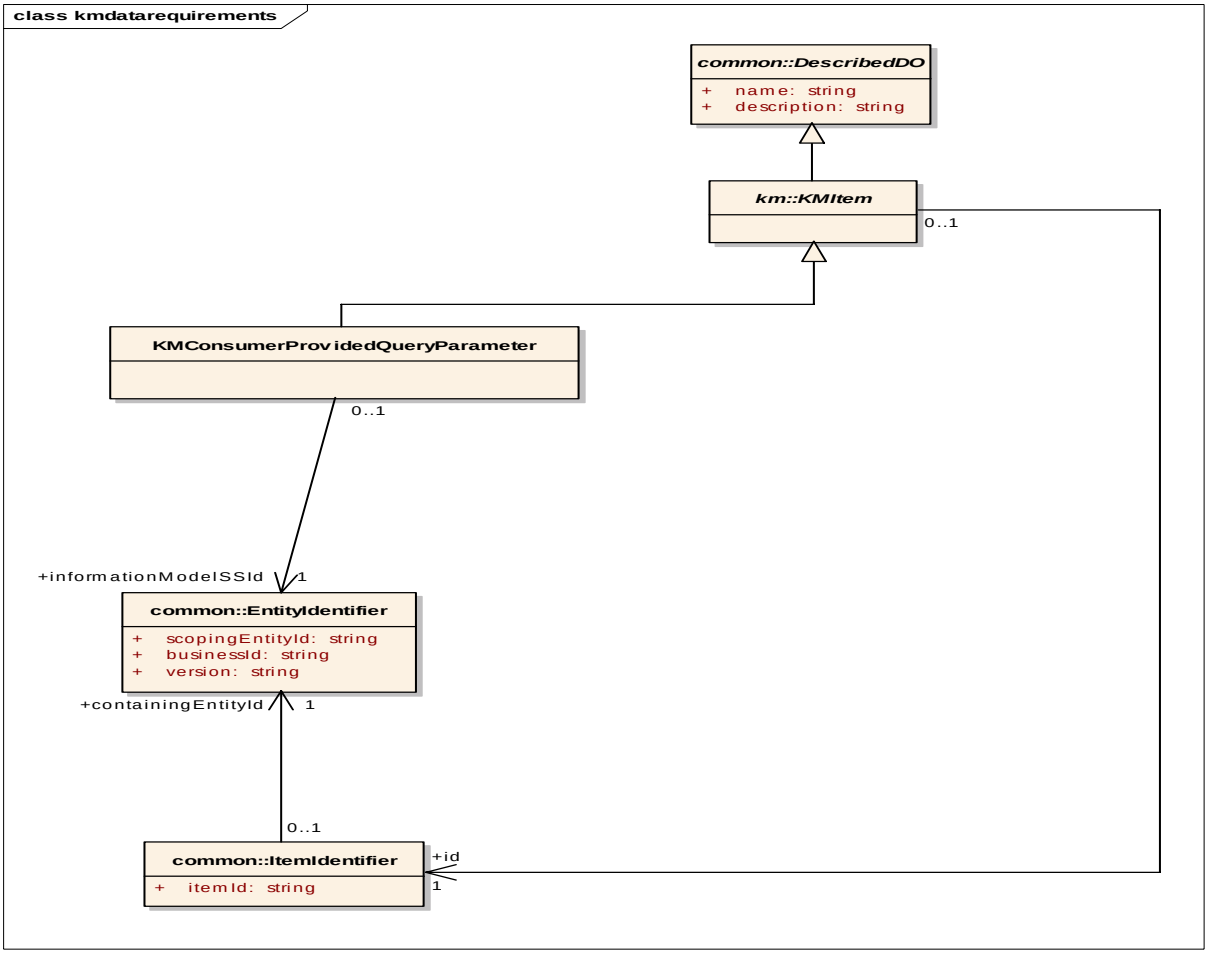


Figure 6.14 - Model for Consumer-Provided Query Parameter

6.4.4.3 KM Data Requirement Group

KM data requirement items are organized into KM data requirement groups (DRGs). DRGs contain one or more data requirement items. Each of these groups is uniquely identified within a given KM through the KM item identifier. This model is defined in the query.kmdatarequirements package and is provided below.

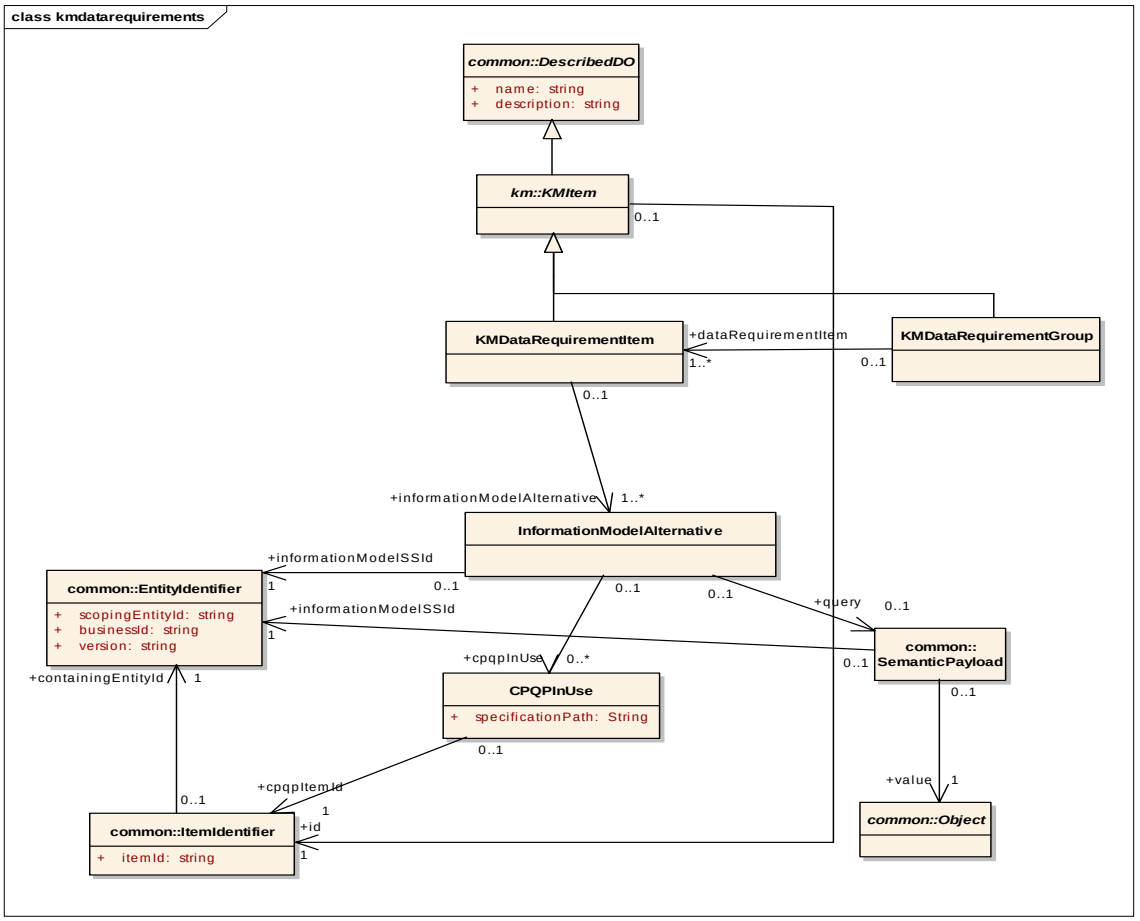


Figure 6.15 - Model for KM Data Requirement Group

6.4.4.4 KM Data Requirements

The data requirements for a KM are expressed in a manner that supports an iterative interaction model. To support such iterative interaction, the DSS expresses its data requirements in terms of DRGs that must be initially provided as well as additional DRGs that may need to be provided during a subsequent interaction, depending on the results of the initial interaction. If the client wishes to interact with the DSS using a single interaction model, a client can simply provide all of the data required by all of the DRGs. This data requirement model is defined in the query.kmdatarequirements package and explained further below.

As shown in the figure below, in expressing the data requirements for a KM, a DSS specifies one or more DRGs as needing to be provided with an initial evaluation. Also, additional DRGs that may be needed in a future interaction are specified. Furthermore, any CPQPs required within the DRGs are specified. Of note, a client may provide all of the above DRGs with an initial evaluation to ensure that a final conclusion is reached after a single interaction.

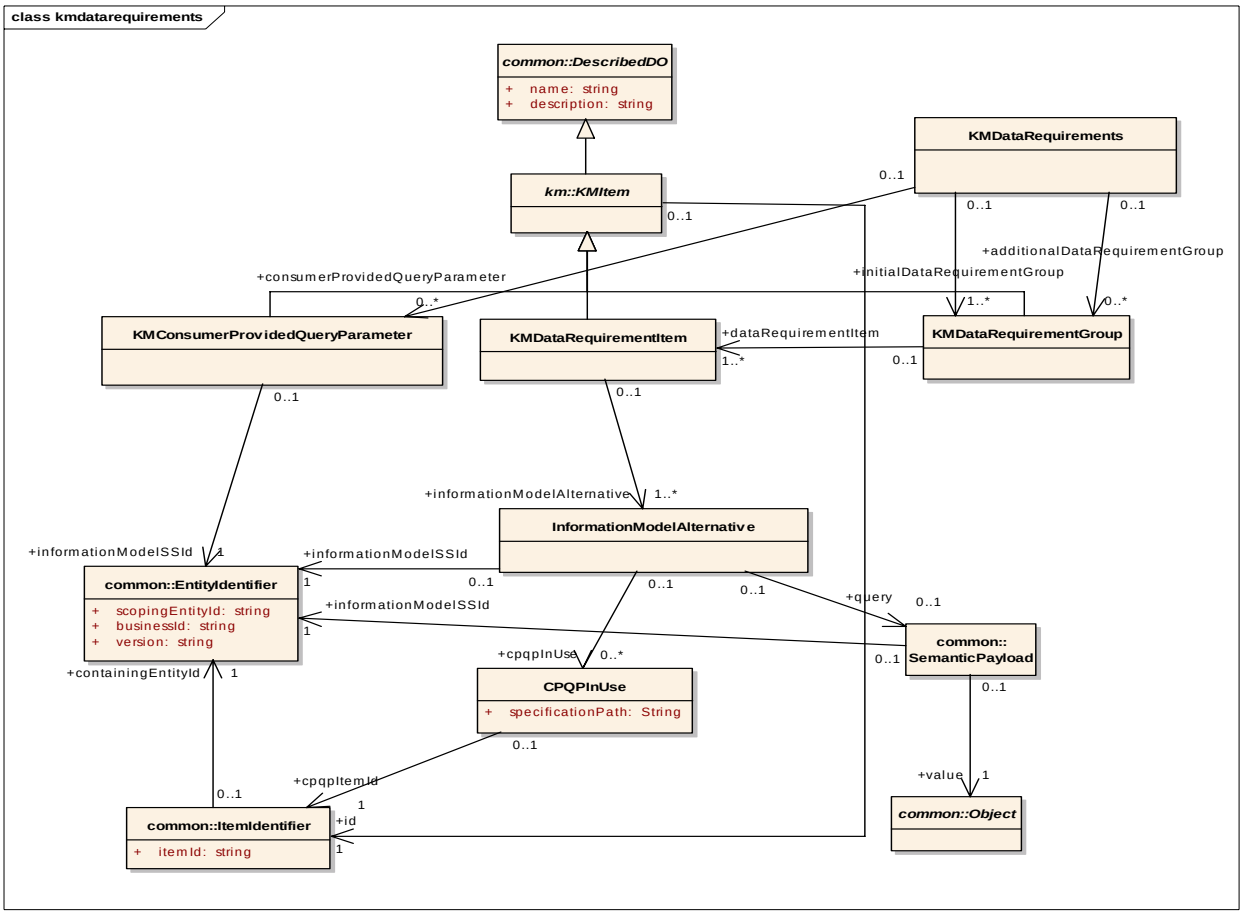


Figure 6.16 - Model for KM Data Requirements

6.4.5 KM Evaluation Result Semantics

KMs return one or more evaluation results as specified by a semantic signifier. This model is defined in the query.evaluationresult package.

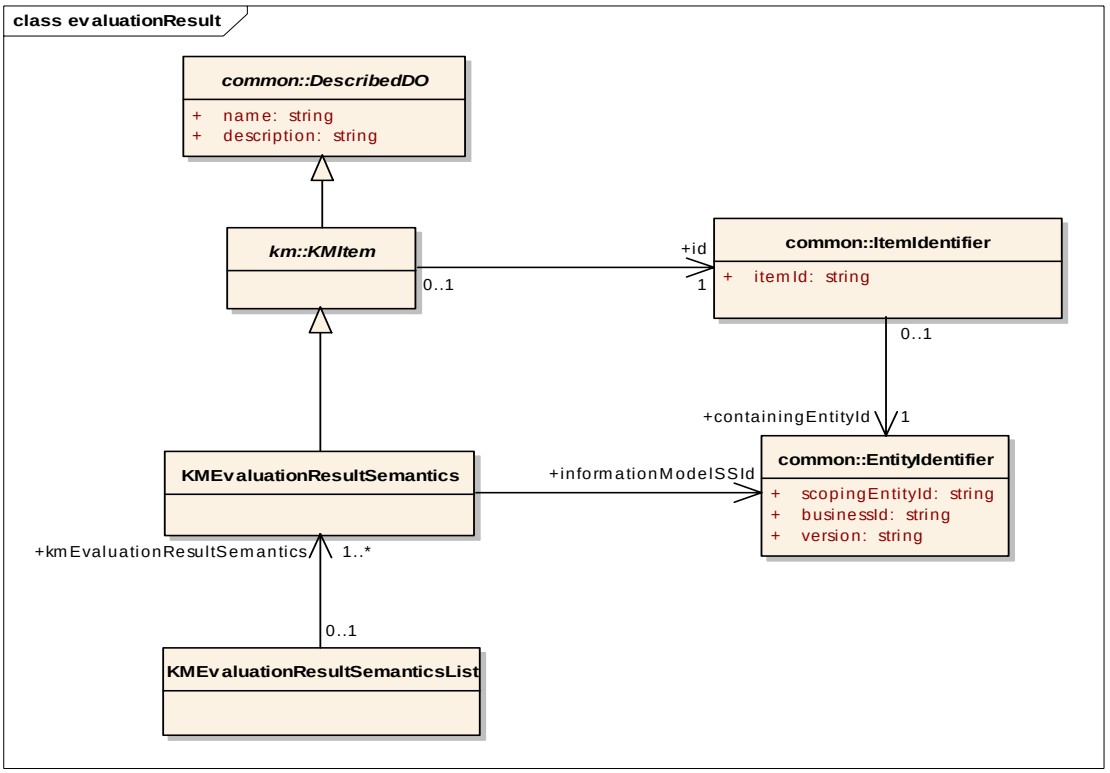


Figure 6.17 - Model for KM Evaluation Result Semantics

6.5 Exception Model Elements

Models for exceptions thrown by the service are in the common.exception, evaluation.exception, and query.exception packages. These exception models are not further described here, as they are described in the service operations themselves in 6.8, Metadata Discovery Interface, 6.9, Query Interface, and 6.10, Evaluation Interface. The three models are provided here for reference.

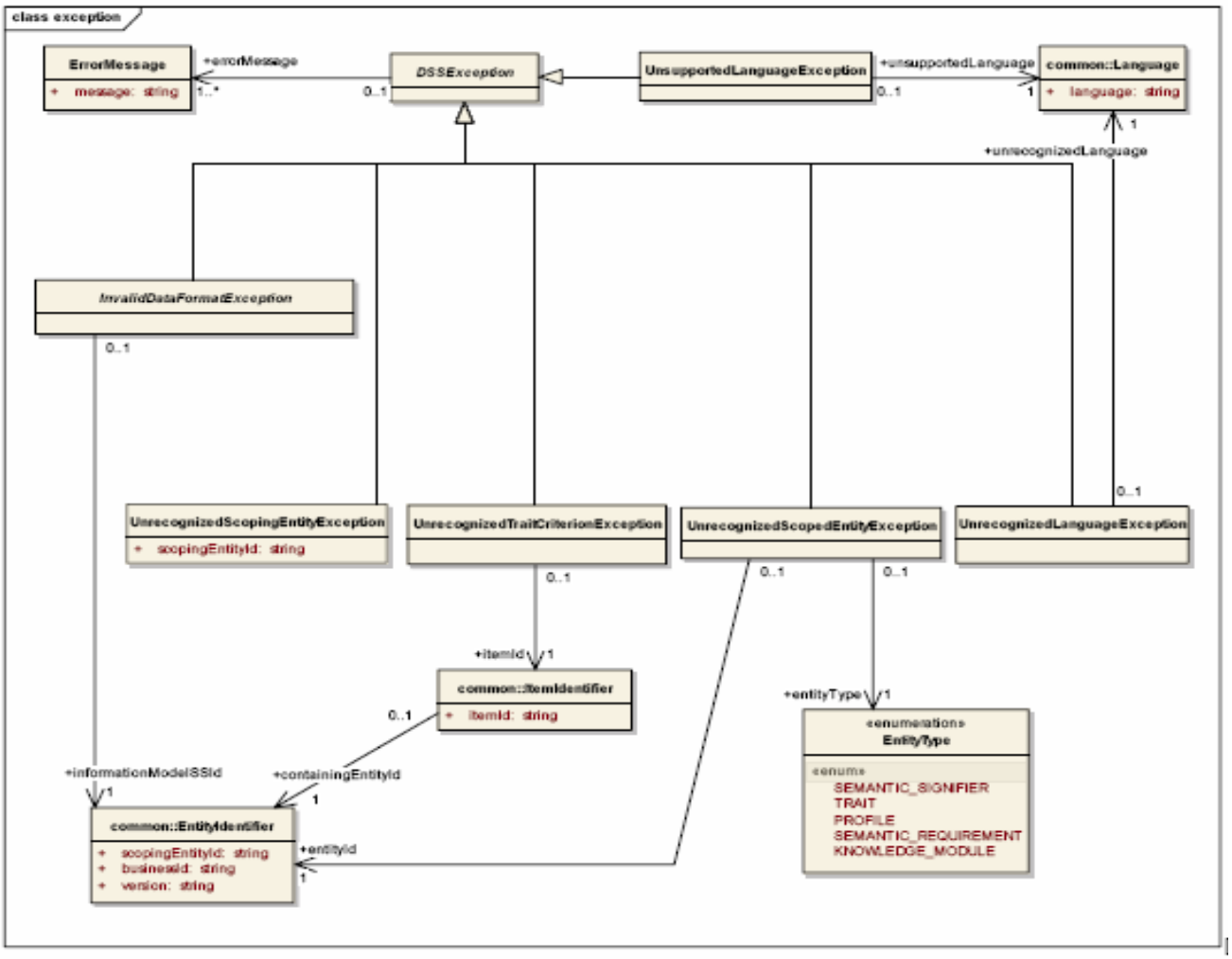


Figure 6.18 - Model for Exceptions in common.exception package

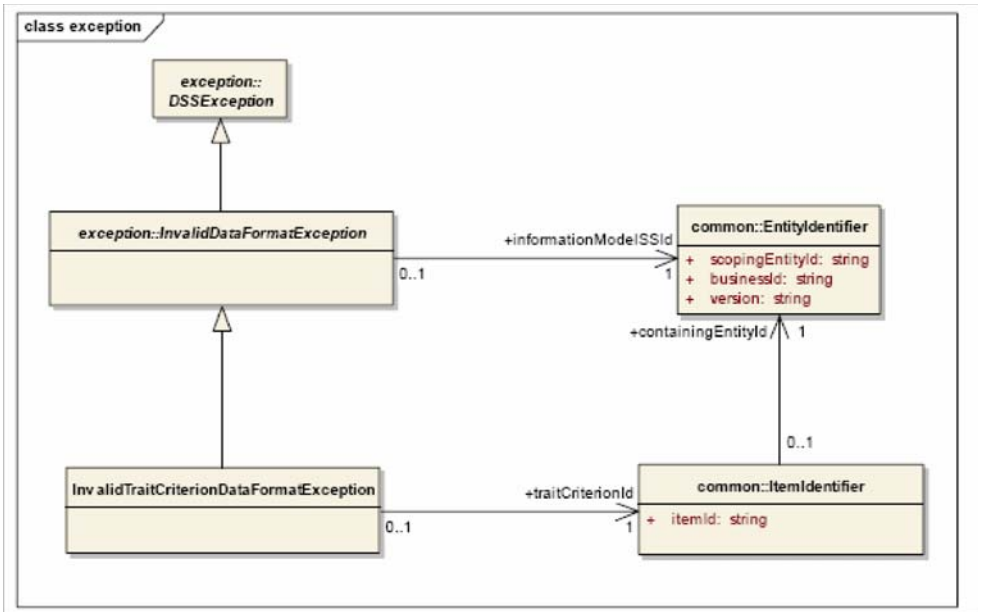


Figure 6.19 - Model for Exceptions in query.exception package

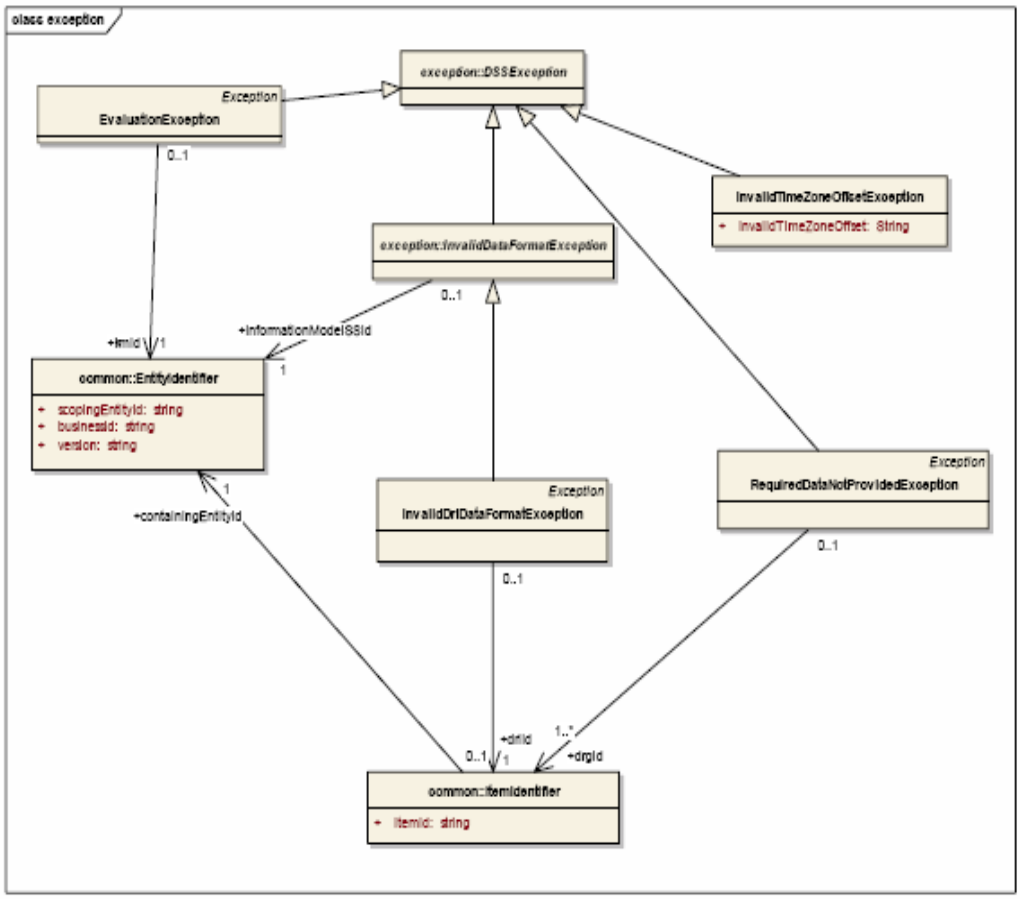


Figure 6.20 - Model for Exceptions in evaluation.exception package

6.6 KM Search Criteria Model Elements

KMs may be searched based on various search criteria. A search may identify KMs that fulfill search criteria perfectly or partially, with search results provided in a ranked list based on relevance. This model for search criteria is provided below and further described.

6.6.1 Search Criteria

The search criteria are modeled in the query.criteria packaged and provided below. Search criteria consist of the following:

- The maximum number of KMs to return in the search result (integer; minimum value of 1).
- The minimum search score required for a KM to be included in the search result (integer; value of 1 to 100). A perfect match shall have a score of 100, and a non-perfect match shall have a score of between 1 to 99. Implementations of the scoring mechanism are vendor-specific. One suggestion is to make the score the % of criteria that match.

- Search inclusion and exclusion criteria. An inclusion criterion is used to include KMs into the search result list and/or increase the KMs' search score, whereas an exclusion criterion is used to exclude KMs from the search result list and/or reduce the KMs' search score. The following criteria may be used as inclusion and/or exclusion criteria:
    - Knowledge module trait criteria
    - Knowledge module statuses. The possible knowledge module statuses are enumerated in 6.4.1.1, Knowledge Module Status.
    - Evaluation result semantics used by a knowledge module
    - Data requirement items in use by a knowledge module
    - Specified relationships to specified knowledge modules

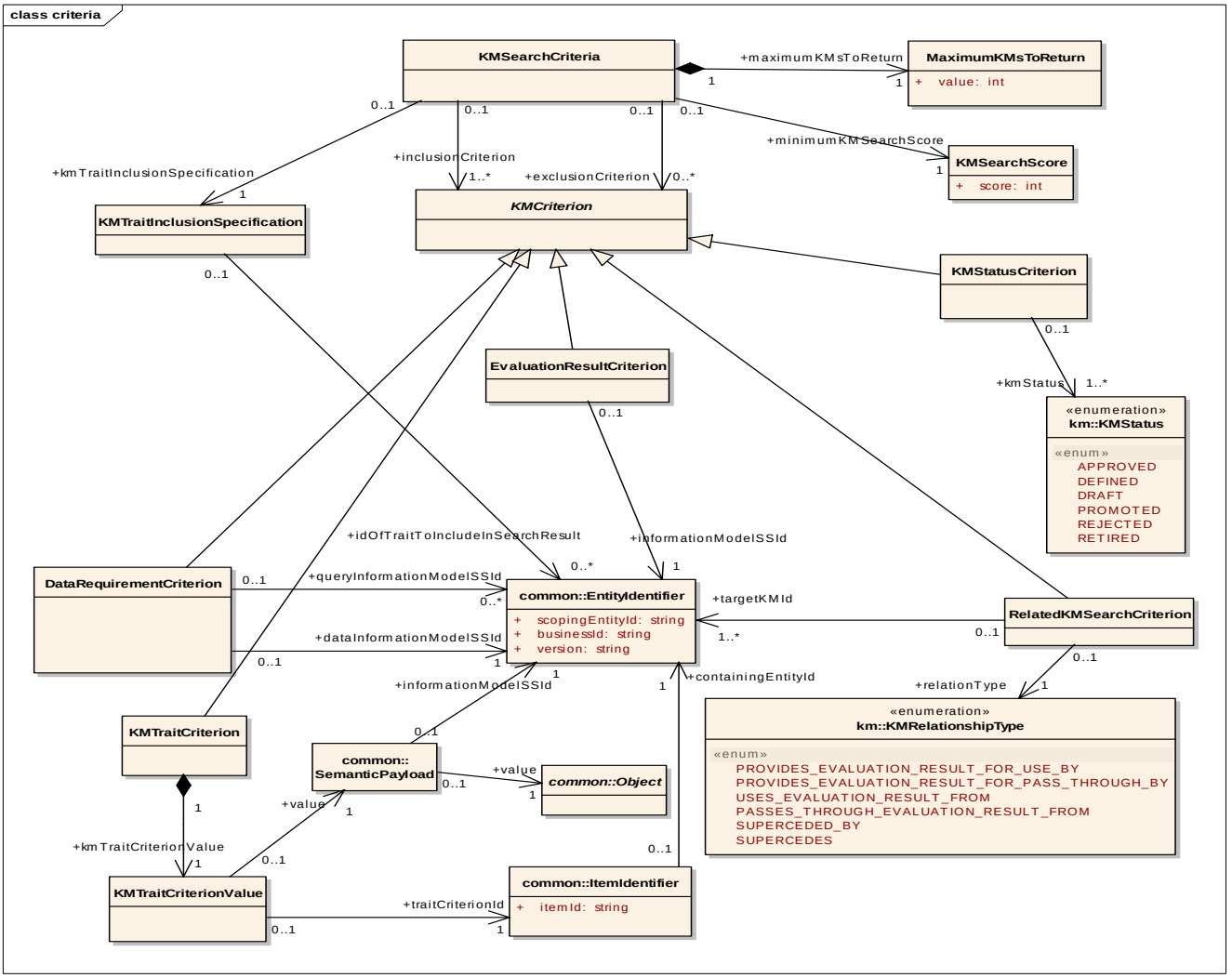


Figure 6.21 - Model for KM Search Criteria

6.7 Evaluation Payload Elements

The evaluation.request and evaluation.response packages contain model elements that represent the input and output payloads of the DSS evaluation operations. These model elements are described below.

6.7.1 Evaluation Request Model

The request model for DSS evaluations is shown below.

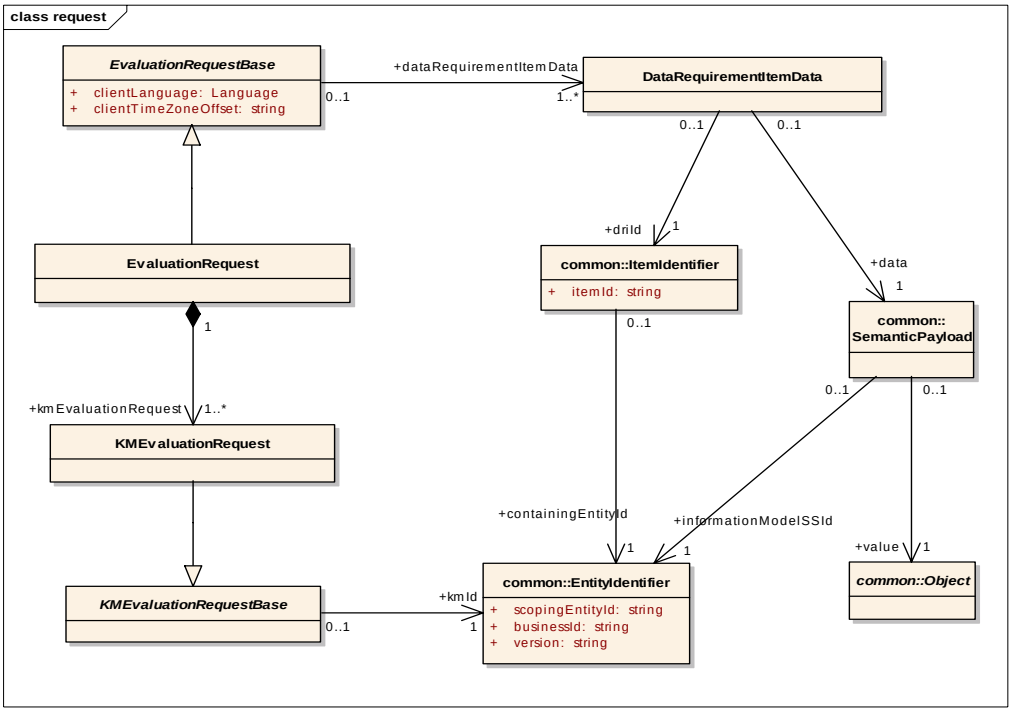


Figure 6.22 - Model for Evaluation Request

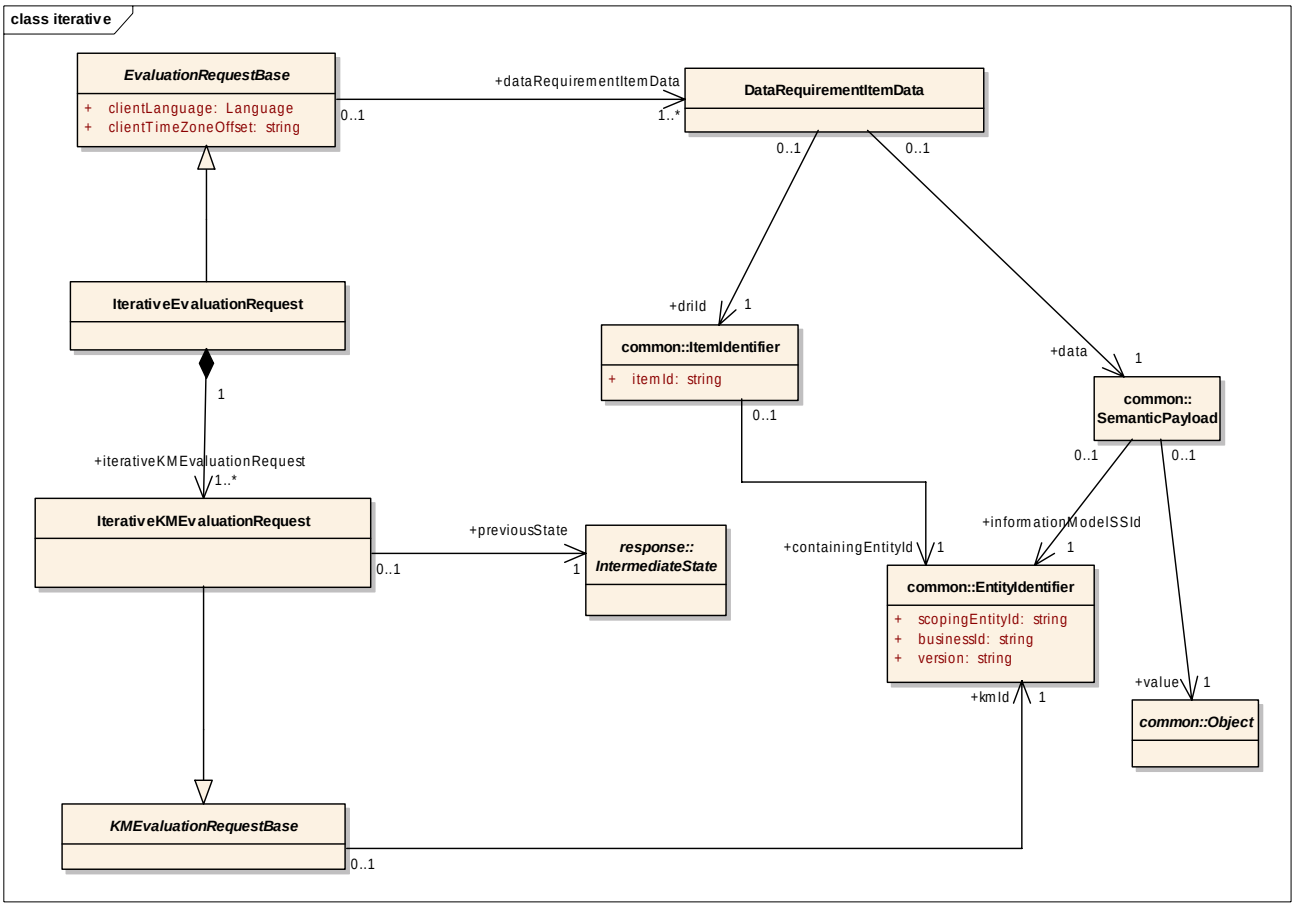


Figure 6.23 - Model for Iterative Evaluation Request

As noted, each evaluation request carries with it DataRequirementItemData, which consist of required data as specified by the knowledge modules as well as a specification of which data requirement item each data item fulfills.

Also note that the request must specify the client's time zone offset from Universal Coordinated Time (UTC). This offset is expressed as +/- hh:mm, e.g., 00:00, -05:00, +07:00. Note that the client's time zone offset cannot be used to determine a geographical time zone. Unless otherwise specified, all time-stamped data provided by the client will be assumed to have this time zone offset.

The evaluation request also contains the client's language. The language is used by the DSS to adjust the evaluation result (e.g., for narrative text included with the evaluation result).

6.7.1.1 Single-Interaction Evaluation Request

In a single-interaction evaluation request, a list of KMs to be used for the evaluation is provided along with the required data.

6.7.1.2 Iterative Interaction Evaluation Request

In the case that a DSS is used iteratively for evaluation, an evaluation request is similar to a single-interaction evaluation request, except that (i) the data provided are either the initial data required for an iterative interaction evaluation request or the data specified as being required next during subsequent iterative steps, and (ii) the intermediate state for each KM evaluation returned from the prior response is provided as a part of the request.

6.7.2 Evaluation Response Model

The response model for DSS evaluations is shown below. As noted, the base evaluation response class contains a list of 0 or more final KM evaluation results as pre-specified by the KM using semantic signifiers.

For single-interaction evaluations, the evaluation result consists of these final KM evaluation results if the required data were provided. If the required data were not provided, the evaluation result for a KM indicates which additional data requirement groups needed to have been provided in order to provide a final evaluation result.

For iterative-interaction evaluations, a final KM evaluation result is provided only when all data required for completing the evaluation has been provided. Until that condition is met, the KM evaluation response returns a specification of the data that must be provided during the next interaction, as well as intermediate state data to pass back with the next request.

Moreover, all KM evaluation responses may contain warnings. These warnings include the actual warning, as well as a specification of the information model used to communicate the warning. An example warning may specify that a retired KM was evaluated, along with the identifier of the superceding KM.

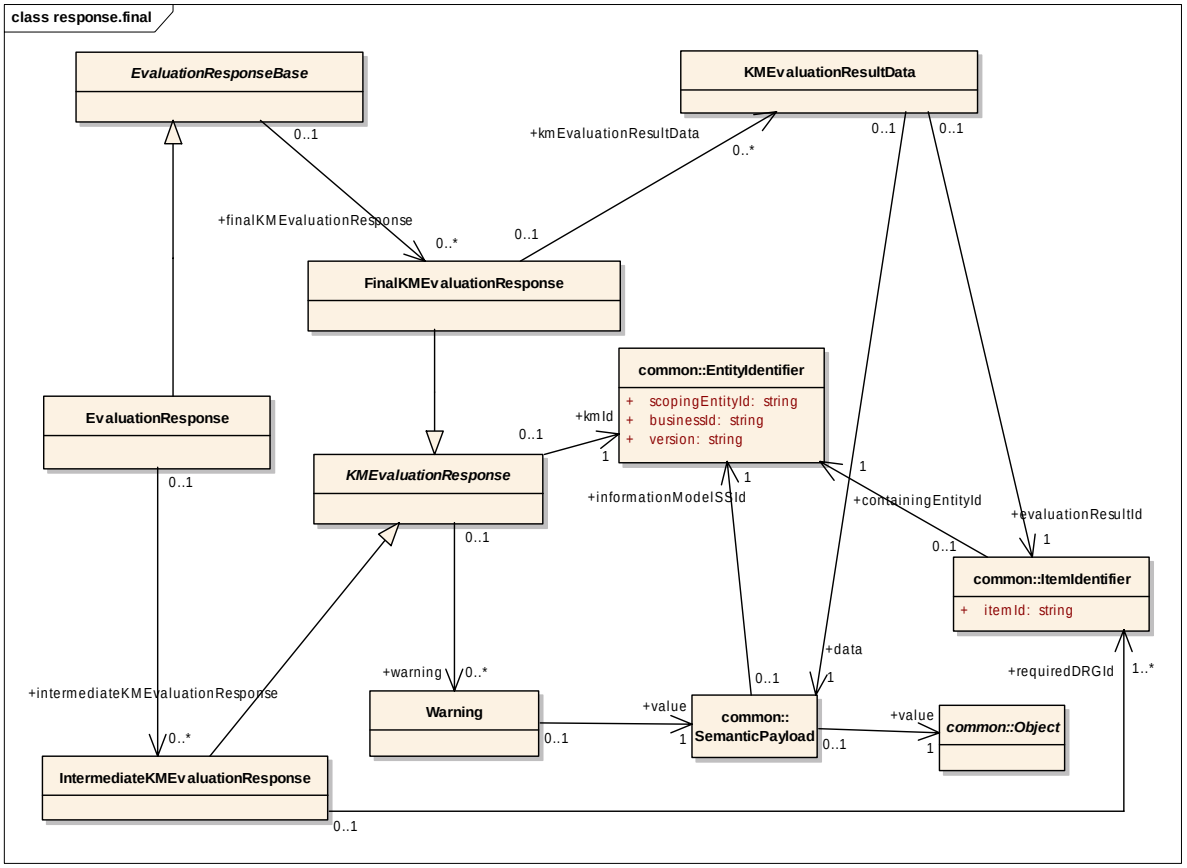


Figure 6.24 - Model for Evaluation Response

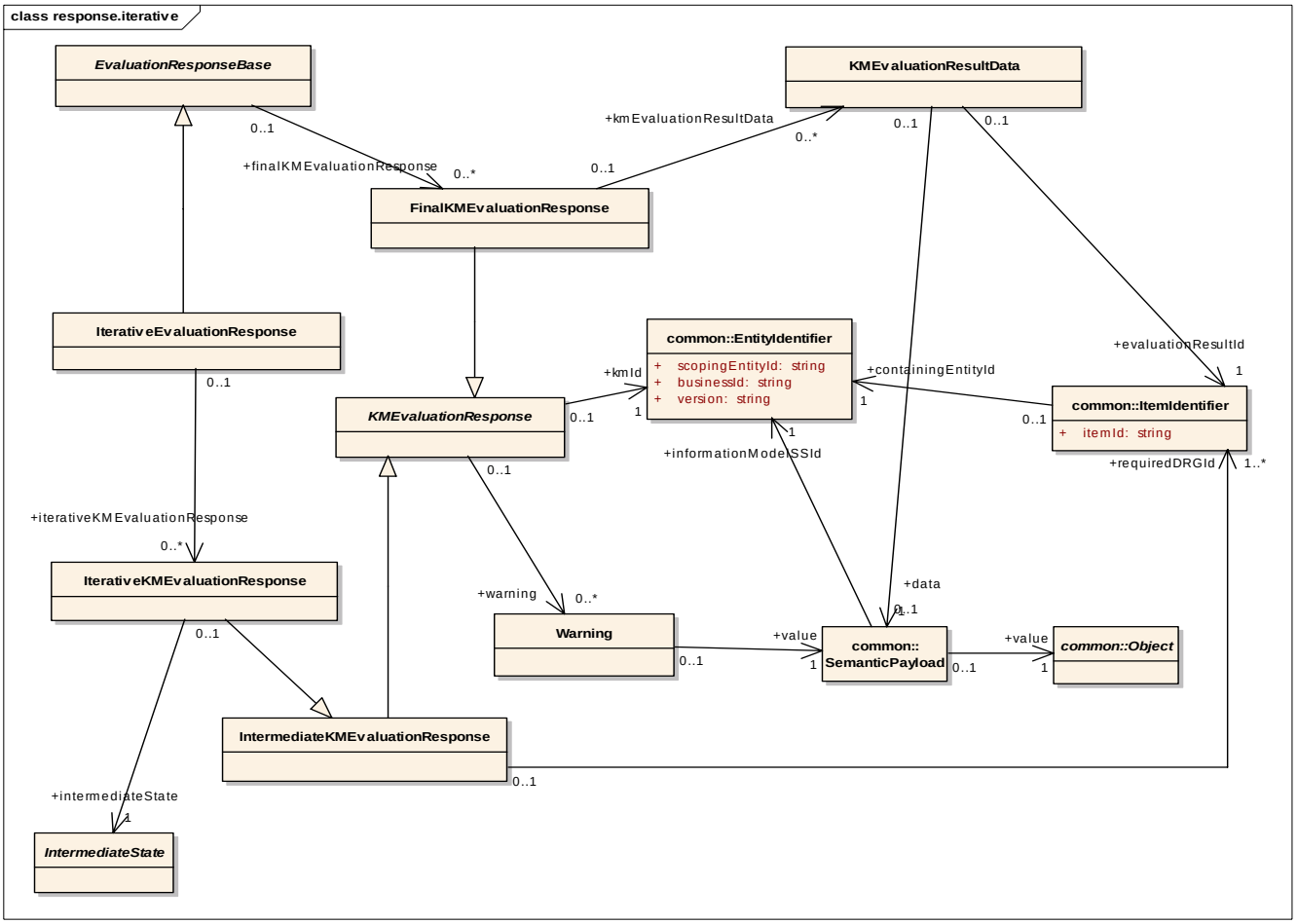
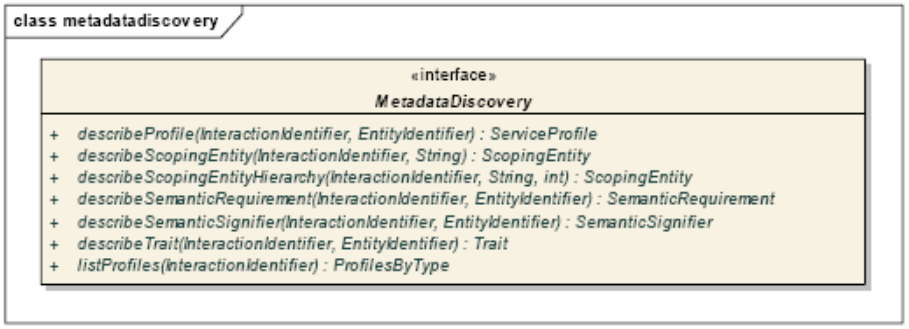


Figure 6.25 - Model for Iterative Evaluation Response

6.8 Metadata Discovery Interface

The DSS Metadata Discovery interface is defined in the service.metadatadiscovery package, and provided below. Details of each operation in the interface are then described. These operations are intended to allow a consumer to start with listProfiles and then to use the other operations to identify the capabilities of the service.



While not specified individually in the definition of DSS operations that follow in 6.8, Metadata Discovery Interface, 6.9, Query Interface, and 6.10, Evaluation Interface, **note that the following hold true for all operations across all interfaces**:

- All operations may throw an exception if the service request is syntactically invalid (e.g., for the SOAP Web service PSM, the Web service call is non-compliant with the DSS's WSDL).
- All operations have the following pre-condition: "No preconditions are assumed."
- All operations have the following post-condition: "If successful, returns output object(s). If unsuccessful, throws exception."
- All operations have the following invariant: "All operations defined are read-only, with no changes made to the DSS."
- All operations have an InteractionIdentifier as an Input. Note that if Inputs are otherwise listed as "None," this means that the sole Input is an InteractionIdentifier.

6.8.1 listProfiles

Description	Returns a list of all of the profiles supported by the service as a ProfilesByType object.
Inputs	None
Outputs	ProfilesByType (6.3.1.3, Service Profile Model): list of profiles supported by the DSS, grouped by type of profile.
Exception conditions	
Aspects left to implementers	Whether and how to sort the output. A suggestion is to order the groups alphabetically by profile type. Within profile types, a suggestion is to sort by the EntityIdentifier of the profiles according to scoping entity identifier, then business identifier, then version.

6.8.2 describeProfile

Description	Throws UnrecognizedScopedEntityException if the specified profile EntityIdentifier is not recognized by the service. If specified EntityIdentifier is recognized by the service, returns a description of the profile as a ServiceProfile object.
Inputs	EntityIdentifier of the profile (6.2.3, Entity Identifier and Interaction Identifier).
Outputs	ServiceProfile (6.3.1.3, Service Profile Model).
Exception conditions	The profile is not recognized by the service (UnrecognizedScopedEntityException)
Aspects left to implementers	

6.8.3 describeScopingEntity

Description	Throws UnrecognizedScopingEntityException if the specified scoping entity identifier is not recognized by the service. If specified, scoping entity identifier is recognized by the service, returns a description of the scoping entity as a ScopingEntity object. Returned ScopingEntity object does not include any children scoping entities.
Inputs	Scoping entity identifier (String) (6.2.2, Scoping Entity).
Outputs	ScopingEntity (6.2.2, Scoping Entity). Does not include any children scoping entities.
Exception conditions	The scoping entity is not recognized by the service (UnrecognizedScopingEntityException).
Aspects left to implementers	

6.8.4 describeScopingEntityHierarchy

Description	Throws UnrecognizedScopingEntityException if the specified scoping entity identifier is not recognized by the service. If specified scoping entity identifier is recognized by the service, returns a description of the scoping entity as a ScopingEntity object. Returned ScopingEntity object includes any descendant scoping entities, up to and including the depth specified.
Inputs	Scoping entity identifier (String) (6.2.2, Scoping Entity). Maximum depth of search (e.g., 2 could result in the inclusion of descendant scoping entities up to the grand children) (positive integer).
Outputs	ScopingEntity (6.2.2, Scoping Entity). Includes any descendant scoping entities, up to and including the depth specified.
Exception conditions	The scoping entity is not recognized by the service (UnrecognizedScopingEntityException).
Aspects left to implementers	

6.8.5 describeSemanticRequirement

Description	Throws UnrecognizedScopedEntityException if the specified semantic requirement EntityIdentifier is not recognized by the service. If specified EntityIdentifier is recognized by the service, returns a description of the semantic requirement as a SemanticRequirement object.
Inputs	EntityIdentifier of the semantic requirement (6.2.3, Entity Identifier and Interaction Identifier).
Outputs	SemanticRequirement (6.4.2.3, Semantic Requirement Model).
Exception conditions	The semantic requirement is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.8.6 describeSemanticSignifier

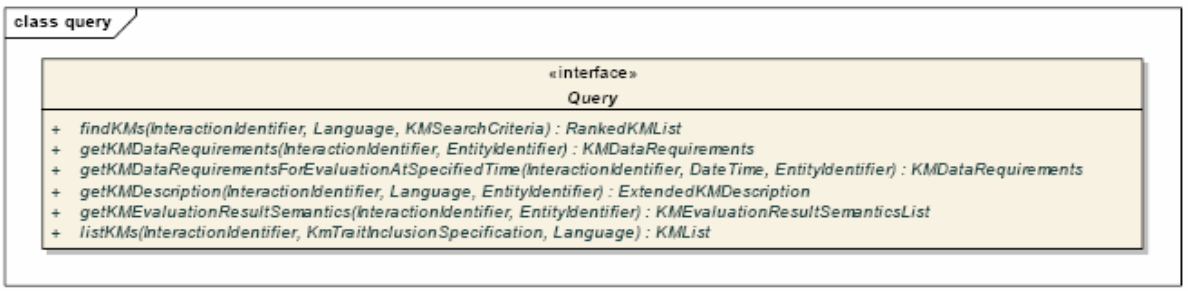
Description	Throws UnrecognizedScopedEntityException if the specified semantic signifier EntityIdentifier is not recognized by the service. If specified, EntityIdentifier is recognized by the service, returns a description of the semantic signifier as a SemanticSignifier object.
Inputs	EntityIdentifier of semantic signifier (6.2.3, Entity Identifier and Interaction Identifier).
Outputs	SemanticSignifier (6.3.2.3, Semantic Signifier Model).
Exception conditions	The semantic signifier is not recognized by the service (UnrecognizedScopedEntityException)
Aspects left to implementers	

6.8.7 describeTrait

Description	Throws UnrecognizedScopedEntityException if the specified trait EntityIdentifier is not recognized by the service. If specified, EntityIdentifier is recognized by the service, returns a description of the trait used for describing knowledge modules as a Trait object.
Inputs	EntityIdentifier of trait (6.2.3, Entity Identifier and Interaction Identifier).
Outputs	Trait (6.4.1.4, Knowledge Module Traits).
Exception conditions	The knowledge module trait is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.9 Query Interface

The DSS Query interface enables the discovery and characterization of knowledge modules. The Query interface is defined in the service.query package and provided below. Details of each operation in the interface are then described.



6.9.1 listKMs

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, returns a list of all knowledge modules hosted by the service as a KMList object. Consumers can specify which traits, if any, to include in the KM descriptions returned. Trait values are provided according to the client's specified language. Note that the language specified by the client must exactly match a language supported by the service. Each KM description includes the status of the KM and its trait values as requested by the consumer.
Inputs	• Client's Language (6.4.2.2, Language specification) • KMTraitInclusionSpecification: specification of which KM traits to include in the KM descriptions returned (6.6.1, Search Criteria).
Outputs	List of KMs (KMList; 6.4.1, Knowledge Module Description). Each KM includes a specification of the following: • KM status • KM trait values for specified traits, localized according to client language.
Exception conditions	• The client's specified Language is not recognized (UnrecognizedLanguageException). • The client's specified Language is recognized but not supported. Note that the language specified by the client must exactly match a language supported by the service in order to avoid this exception (UnsupportedLanguageException). • A knowledge module trait included in the KMTraitInclusionSpecification is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	Whether and how to sort the output. A suggestion is to order the KMs by the EntityIdentifier according to scoping entity identifier, then business identifier, then version.

6.9.2 findKMs

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, returns knowledge modules fulfilling client search criteria as a RankedKMList object. A search may identify KMs that fulfill search criteria perfectly or partially. Search results are provided in a ranked list, with more relevant KMs listed first. KMs included in the search result must have a relevance score of 1 to 100. A KM meeting all client search criteria shall have a score of 100, while a KM that does not meet all client search criteria shall not have a score of 100. Implementations of the scoring mechanism are vendor-specific. One suggestion is to make the score the % of criteria that match. For KMs with the same score, relative ordering in the result list denotes their relative relevance. Consumers can specify which traits, if any, to include in the KM descriptions returned. Trait values are provided according to the client's specified language. Note that the language specified by the client must exactly match a language supported by the service. Each KM description includes the status of the KM and its trait values as requested by the consumer.
Inputs	• Client's Language (6.4.2.2, Language specification) • KMSearchCriteria (6.6.1, Search Criteria)
Outputs	List of knowledge modules fulfilling the search criteria (RankedKMList; 6.4.1, Knowledge Module Description).
Exception conditions	• A knowledge module trait included in the KMTraitInclusionSpecification is not recognized by the service (UnrecognizedScopedEntityException). • A knowledge module specified as the target of a relationship-based search is unrecognized (UnrecognizedScopedEntityException). • A trait criterion identifier is not recognized (UnrecognizedTraitCriterionException). • A trait criterion value has an invalid data format (InvalidTraitCriterionDataFormatException). • An evaluation result semantic signifier specified as a search criterion is not recognized (Unrecognized ScopedEntityException). • A semantic requirement specified as a search criterion is not recognized (Unrecognized ScopedEntityException). • A semantic signifier used to specify a data requirement criterion is not recognized (UnrecognizedScopedEntityException). • The client's specified Language is not recognized (UnrecognizedLanguageException).
Aspects left to implementers	

6.9.3 getKMDescription

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, returns a description of the specified knowledge module as an ExtendedKMDescription object. When language-dependent trait values are available, returns trait values using the client's specified language. Note that the language specified by the client must exactly match a language supported by the service.
Inputs	• EntityIdentifier of knowledge module (6.2.3, Entity Identifier and Interaction Identifier) • Client's Language (6.4.2.2, Language specification)
Outputs	ExtendedKMDescription (6.4.1, Knowledge Module Description)
Exception conditions	• The client's specified Language is not recognized (UnrecognizedLanguageException). • The client's specified Language is recognized but not supported. Note that the language specified by the client must exactly match a language supported by the service in order to avoid this exception (UnsupportedLanguageException). • The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.9.4 getKMEvaluationResultSemantics

Description	Throws UnrecognizedScopedEntityException if the specified knowledge module EntityIdentifier is not recognized by the service. If specified EntityIdentifier is recognized by the service, returns a specification of the information model(s) that will be used by the knowledge module when returning an evaluation result as a KMEvaluationResultSemanticsList object.
Inputs	EntityIdentifier of the knowledge module (6.2.3, Entity Identifier and Interaction Identifier).
Outputs	KMEvaluationResultSemanticsList (6.4.5, KM Evaluation Result Semantics).
Exception conditions	The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.9.5 getKMDataRequirements

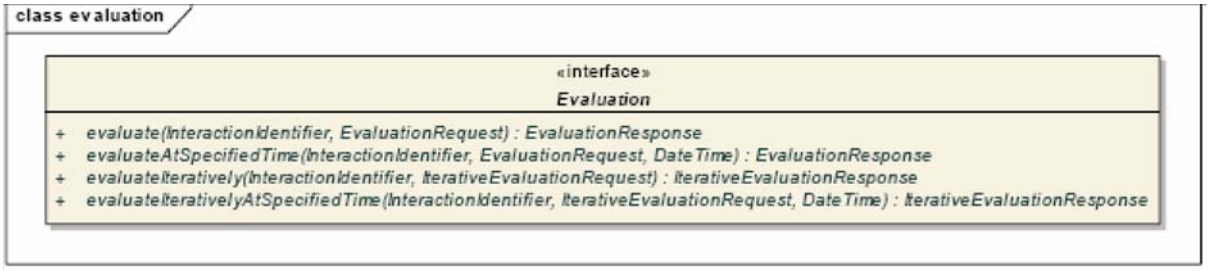
Description	Throws UnrecognizedScopedEntityException if the specified knowledge module EntityIdentifier is not recognized by the service. If specified EntityIdentifier is recognized by the service, returns a specification of the data required by the knowledge module for conducting an evaluation as a KMDataRequirements object.
Inputs	EntityIdentifier of knowledge module (6.2.3, Entity Identifier and Interaction Identifier).
Outputs	KMDataRequirements (6.4.4.4, KM Data Requirements).
Exception conditions	The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.9.6 getKMDataRequirementsForEvaluationAtSpecifiedTime

Description	Throws UnrecognizedScopedEntityException if the specified knowledge module EntityIdentifier is not recognized by the service. If specified, EntityIdentifier is recognized by the service, returns a specification of the data required by the knowledge module for conducting an evaluation as a KMDataRequirements object. If there are any query parameters that use absolute date-times (e.g., search 1/1/09 to 7/1/09) instead of relative date-times (e.g., search past 6 months), then these absolute date-time parameters will be populated to be appropriate for an evaluation at the specified date-time. Note that if a DSS provider does not use absolute date-time query parameters, then the DSS provider can implement this operation by simply calling the getKMDataRequirements operation.
Inputs	• DateTime of intended evaluation • EntityIdentifier of knowledge module (6.2.3, Entity Identifier and Interaction Identifier)
Outputs	KMDataRequirements (6.4.4.4, KM Data Requirements).
Exception conditions	The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.10 Evaluation Interface

The DSS Evaluation interface enables data evaluation using knowledge modules. The Evaluation interface is defined in the service.evaluation package and provided below. Details of each operation in the interface are then described.



6.10.1 evaluate

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, evaluates in a non-iterative fashion one or more knowledge modules using the data provided as an EvaluationRequest object and returns the result(s) of the evaluation as an EvaluationResponse object. All time-stamped data are considered to have the time zone offset specified by the client, unless otherwise noted. The provision of excessive data (i.e., unrequired DataRequirementItemData) shall be ignored without leading to an exception. However, a warning may be provided.
Inputs	EvaluationRequest object (6.7.1, Evaluation Request Model)
Outputs	EvaluationResponse object (6.7.2, Evaluation Response Model)
Exception conditions	• The specified time zone offset is invalid (InvalidTimzeZoneOffsetException). • The client's specified Language is not recognized (UnrecognizedLanguageException). • The client's specified Language is recognized but not supported. Note that the language specified by the client must exactly match a language supported by the service in order to avoid this exception (UnsupportedLanguageException). • A requested knowledge module does not exist (UnrecognizedScopedEntityException). • Required data not provided. This exception specifies the data requirement group(s) for which data were required but not provided (RequiredDataNotProvidedException). • Required data were not provided in the correct format (InvalidDriDataFormatException). • An exception occurred during the evaluation process (EvaluationException).
Aspects left to implementers	

6.10.2 evaluateAtSpecifiedTime

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, evaluates in a non-iterative fashion one or more knowledge modules using the data provided as an EvaluationRequest object and returns the result(s) of the evaluation as an EvaluationResponse object. Conducts evaluation as if it was currently the specified date and time. All time-stamped data are considered to have the time zone offset specified by the client, unless otherwise noted. The provision of excessive data (i.e., unrequired DataRequirementItemData) shall be ignored without leading to an exception. However, a warning may be provided.
Inputs	• EvaluationRequest object (6.7.1, Evaluation Request Model) • DateTime of the evaluation
Outputs	EvaluationResponse object (6.7.2, Evaluation Response Model)
Exception conditions	• The specified time zone offset is invalid (InvalidTimzeZoneOffsetException). • The client's specified Language is not recognized (UnrecognizedLanguageException). • The client's specified Language is recognized but not supported. Note that the language specified by the client must exactly match a language supported by the service in order to avoid this exception (UnsupportedLanguageException). • A requested knowledge module does not exist (UnrecognizedScopedEntityException). • Required data not provided. This exception specifies the data requirement group(s) for which data was required but not provided (RequiredDataNotProvidedException). • Required data was not provided in the correct format (InvalidDriDataFormatException). • An exception occurred during the evaluation process (EvaluationException).
Aspects left to implementers	

6.10.3 evaluateIteratively

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, evaluates the data provided by the client using one or more knowledge modules and returns the result(s) of the evaluation. Conducts evaluation iteratively, returning intermediate state data and specification of additional required data if final conclusions cannot be initially reached. All time-stamped data are considered to have the time zone offset specified by the client, unless otherwise noted. The provision of excessive data (i.e., unrequired DataRequirementItemData) shall be ignored without leading to an exception. However, a warning may be provided.
Inputs	IterativeEvaluationRequest object (6.7.1, Evaluation Request Model)
Outputs	IterativeEvaluationResponse object (6.7.2, Evaluation Response Model)
Exception conditions	• The specified time zone offset is invalid (InvalidTimzeZoneOffsetException). • The client's specified Language is not recognized (UnrecognizedLanguageException). • The client's specified Language is recognized but not supported. Note that the language specified by the client must exactly match a language supported by the service in order to avoid this exception (UnsupportedLanguageException). • A requested knowledge module does not exist (UnrecognizedScopedEntityException). • Required data not provided. This exception specifies the data requirement group(s) for which data were required but not provided (RequiredDataNotProvidedException). • Required data were not provided in the correct format (InvalidDriDataFormatException). • An exception occurred during the evaluation process (EvaluationException).
Aspects left to implementers	

6.10.4 evaluateIterativelyAtSpecifiedTime

Description	Throws one of the exceptions if an exception condition is present. If none of the exception conditions are present, evaluates the data provided by the client using one or more knowledge modules and returns the result(s) of the evaluation. Conducts evaluation iteratively, returning intermediate state data and specification of additional required data if final conclusions cannot be initially reached. Conducts evaluation as if it was currently the specified date and time. All time-stamped data are considered to have the time zone offset specified by the client, unless otherwise noted. The provision of excessive data (i.e., unrequired DataRequirementItemData) shall be ignored without leading to an exception. However, a warning may be provided.
Inputs	• IterativeEvaluationRequest object (6.7.1, Evaluation Request Model) • Date and time of the evaluation
Outputs	IterativeEvaluationResponse object (6.7.2, Evaluation Response Model)
Exception conditions	• The specified time zone offset is invalid (InvalidTimzeZoneOffsetException). • The client's specified Language is not recognized (UnrecognizedLanguageException). • The client's specified Language is recognized but not supported. Note that the Language specified by the client must exactly match a language supported by the service in order to avoid this exception (UnsupportedLanguageException). • A requested knowledge module does not exist (UnrecognizedScopedEntityException). • Required data not provided. This exception specifies the data requirement group(s) for which data was required but not provided (RequiredDataNotProvidedException). • Required data was not provided in the correct format (InvalidDriDataFormatException). • An exception occurred during the evaluation process (EvaluationException).
Aspects left to implementers	

6.11 Profiles and Semantic Requirements Specified as a Part of this Specification

6.11.1 Overview

This specification specifies several profiles and semantic requirements to ensure a minimum level of interoperability among DSSs. These profiles and semantic requirements are derived from, and extend, those profiles and semantic requirements specified in Section 6 of the HL7 DSS SFM.

This sub clause defines these normative specifications that consist of three functional profiles (the HSSP Minimum DSS Functional Profile, the HSSP Core DSS Functional Profile, and the HSSP Advanced Time Handling DSS Functional Profile), one semantic profile (the HSSP Minimum DSS Semantic Profile), one semantic requirement (the HSSP Minimum DSS Trait Set Requirement), and three conformance profiles (the HSSP Minimum DSS Conformance Profile, the HSSP Core DSS Conformance Profile, and the HSSP Advanced Time Handling DSS Conformance Profile). These specifications are defined below. Moreover, Table 6.4 provides a summary of the operations supported by the functional profiles, and Figure 6.26 outlines the relationships between the profiles specified in this specification.

Table 6.4 - Operations supported by functional profiles

Operation	Supported by HSSP Minimum DSS Functional Profile	Supported by HSSP Core DSS Functional Profile	Supported by HSSP Advanced Time Handling DSS Functional Profile
describeProfile	X	X	X
describeScopingEntity	X	X	X
describeScopingEntityHierarchy	X	X	X
describeSemanticRequirement	X	X	X
describeSemanticSignifier	X	X	X
describeTrait	X	X	X
listProfiles	X	X	X
findKMs		X	X
getKMDataRequirements	X	X	X
getKMDataRequirementsForEvaluation AtSpecifiedTime			X
getKMDescription	X	X	X
getKMEvaluationResultSemantics	X	X	X
listKMs	X	X	X
evaluate	X	X	X
evaluateAtSpecifiedTime			X
evaluateIteratively		X	X
evaluateIterativelyAtSpecifiedTime			X

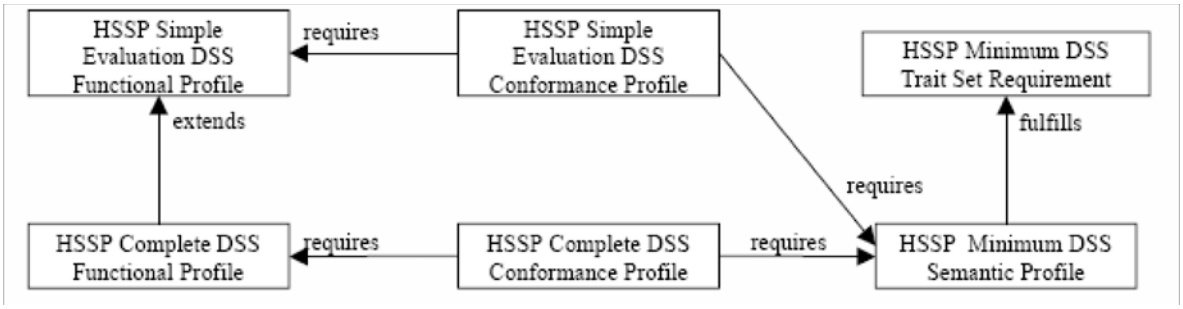


Figure 6.26 - Relationships between profiles

6.11.2 HSSP Simple Evaluation DSS Functional Profile, Version 1.0

To claim conformance to the HSSP Simple DSS Functional Profile, version 1.0, a DSS must implement and support the following service operations defined in this specification:

From the Evaluation interface:

evaluate

The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Simple_Evaluation_DSS_Functional_Profile

version: 1.0

type: FUNCTIONAL_PROFILE

6.11.3 HSSP Complete DSS Functional Profile, Version 1.0

To claim conformance to the HSSP Complete DSS Functional Profile, version 1.0, a DSS must implement and support all service operations defined in this specification. The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Complete_DSS_Functional_Profile

version: 1.0

type: FUNCTIONAL_PROFILE

6.11.4 HSSP Minimum DSS Semantic Profile, Version 1.0

To claim conformance to the HSSP Minimum DSS Semantic Profile, version 1.0, the service must fulfill the HSSP Minimum DSS Trait Set Requirement, which is specified in sub clause 6.11.3. The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Minimum_Meta_Data_DSS_Semantic_Profile

version: 1.0

type: SEMANTIC_PROFILE

6.11.5 HSSP Minimum DSS Trait Set Requirement, Version 1.0

To claim conformance to this trait set requirement, all knowledge modules in the DSS must support the traits and trait criteria specification in sub clause 6.11.5.1 and 6.11.5.2. The relevant identifying parameters for this semantic requirement shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Minimum_DSS_Trait_Set_Requirement

version: 1.0

type: TRAIT_SET_REQUIREMENT

6.11.5.1 Knowledge Module Traits Required by Trait Set Requirement

Please note that all schemas referenced in this sub clause are included as supplemental files with this specification. Referenced HL7 version 3 schemas were obtained from the 2008 HL7 version 3 normative edition standard, available at http://www.hl7.org/memonly/downloads/v3edition.cfm#V32008. Note that the voc.xsd file must be taken from the cda folder in the ballot package. Relative path modifications were applied to the schemas' "include" statements as necessary.

The "root global element name" defined below is specific to the XML Web Service PSM and corresponds to the semantic signifier's xsdRootGlobalElementName attribute in the XSDComputableDefinition class (see 6.3.2.3, Semantic Signifier Model).

6.11.5.1.1 StewardOrganization

Description:

The organization acting as the steward of the KM

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: StewardOrganization

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: StewardOrganization

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.11.5.1.2 CreationDate

Description:

Date KM was first created

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: CreationDate

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: CreationDate

Trait attributes:

Is mandatory: true

Trait value is localized: false

6.11.5.1.3 LastReviewDate

Description:

Date when KM was last reviewed for accuracy

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: LastReviewDate

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: LastReviewDate

Trait attributes:

Is mandatory: true

Trait value is localized: false

6.11.5.1.4 AuthorList

Description:

A list of the KM's authors. May be empty.

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: AuthorList

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: AuthorList

Trait attributes:

Is mandatory: true

Trait value is localized: false

6.11.5.1.5 FreeTextKeywordList

Description:

A list of free text keywords that characterize the KM. May be empty.

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: FreeTextKeywordList

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: FreeTextKeywordList

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.11.5.1.6 CodedValueKeywordList

Description:

A list of coded value keywords that characterize the KM. May be empty. Use of SNOMED CT encouraged.

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: CodedValueKeywordList

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: CodedValueKeywordList

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.11.5.1.7 Purpose

Description:

The purpose of a KM, intended for a medical informaticist

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: Purpose

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: Purpose

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.11.5.1.8 Explanation

Description:

An explanation of how the KM uses the required data to generate evaluation results, intended for a medical informaticist.

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: Explanation

Version: 1.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: Explanation

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.11.5.2 Knowledge Module Trait Criteria that Must be Available to Query for KMs Based on Trait Value

6.11.5.2.1 ReviewedOnOrAfter

Parent trait: LastReviewDate (Section 6.11.5.1.3, LastReviewDate)

Description:

Specifies that LastReviewDate must have been on or after the specified date

Trait criterion identifier:

Containing entity identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: LastReviewDate

Version: 1.0

Item identifier: ReviewedOnOrAfter

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: ReviewedOnOrAfter

6.11.5.2.2 ReviewedWithinLastXDays

Parent trait: LastReviewDate (6.11.5.1.3)

Description:

Specifies that LastReviewDate must have occurred within specified number of days.

Trait criterion identifier:

Containing entity identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: LastReviewDate

Version: 1.0

Item identifier: ReviewedWithinLastXDays

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: ReviewedWithinLastXDays

6.11.5.2.3 FreeTextKeywordContainsString

Parent trait: FreeTextKeywordList (6.11.5.1.5)

Description:

Specifies that at least one free text keyword must contain the specified string.

Trait criterion identifier:

Containing entity identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: FreeTextKeywordList

Version: 1.0

Item identifier: FreeTextKeywordContainsString

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: FreeTextKeywordContainsString

6.11.5.2.4 CodedValueKeywordExists

Parent trait: CodedValueKeywordList (6.11.5.1.6)

Description:

Specifies that the specified code exists as a coded value keyword. Note that because the HL7 version 3 Coded Value with Equivalents schema element is used, the search concept may be specified using multiple vocabularies. A match on any of the equivalent concept codes shall be considered a keyword match.

Trait criterion identifier:

Containing entity identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: CodedValueKeywordList

Version: 1.0

Item identifier: CodedValueKeywordExists

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: CodedValueKeywordExists

6.11.5.2.5 CodedValueKeywordOrKeywordDescendantExists

Parent trait: CodedValueKeywordList (6.11.5.1.6)

Description:

Specifies that the specified concept or a descendant concept exists as a coded value keyword. Note that because the HL7 version 3 Coded Value with Equivalents schema element is used, the search concept may be specified using multiple vocabularies. A match on any of the equivalent concept codes shall be considered a keyword match. Also, note that because a DSS provider may have limited and/or different capabilities for fulfilling this trait search criterion, a client may wish to instead use the CodedValueKeywordExists trait criterion instead (6.11.5.2.4) and specify all of the descendant concepts of interest explicitly.

Trait criterion identifier:

Containing entity identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: CodedValueKeywordList

Version: 1.0

Item identifier: CodedValueKeywordOrKeywordDescendantExists

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema

Version: 1.0

Root global element name: CodedValueKeywordOrKeywordDescendantExists

6.11.6 HSSP Simple Evaluation DSS Conformance Profile, Version 1.0

To claim conformance to this profile, a DSS must be conformant with the HSSP Simple Evaluation DSS Functional Profile, version 1.0 (6.11.2, HSSP Simple Evaluation DSS Functional Profile, Version 1.0) and the HSSP Minimum DSS Semantic Profile, version 1.0 (6.11.4, HSSP Minimum DSS Semantic Profile, Version 1.0). The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Complete_DSS_Conformance_Profile

version: 1.0

type: CONFORMANCE_PROFILE

6.11.7 HSSP Complete DSS Conformance Profile, Version 1.0

To claim conformance to this profile, a DSS must be conformant with the HSSP Complete DSS Functional Profile, version 1.0 (6.11.3, HSSP Complete DSS Functional Profile, Version 1.0) and the HSSP Minimum DSS Semantic Profile, version 1.0 (6.11.5, HSSP Minimum DSS Trait Set Requirement, Version 1.0). The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Complete_DSS_Conformance_Profile

version: 1.0

type: CONFORMANCE_PROFILE

6.12 Minimal Requirement for Claiming Conformance to HSSP DSS Standard

To claim conformance to the HSSP DSS standard, a DSS must be conformant with the HSSP Simple Evaluation DSS Conformance Profile, version 1.0 (6.11.6, HSSP Simple Evaluation DSS Conformance Profile, Version 1.0).

6.13 Future Specifications of Profiles and Semantic Requirements

It is anticipated that many more semantic profiles and semantic requirements will be specified in the future. These specifications are expected to take the form of HL7 and OMG-defined specifications as well as specifications defined by other entities, such as individual vendors.

7 DSS Platform Specific Model for XML Web Services

7.1 General

The Platform Specific Model (PSM) for XML Web services is derived from the platform independent model specified in Clause 6. The PSM is defined in the accompanying normative WSDL and associated XSD.

Note that, for obvious reasons, the actual URL address of the service (specified in the WSDL as www.exampleLocation.com/evaluation, www.exampleLocation.com/query, and www.exampleLocation.com/metadata) are non-normative and should be replaced by the implementer. Also note that security handling is outside of the scope of this specification, but should be considered. Typical approaches to handling security may include the use of the WS-Security protocol and Transport Layer Security (TLS). At a minimum, implementers should ensure transport security for patient-identifiable information provided by clients. Implementers should also consider transport security, authentication, and authorization for all service calls. Of note, an implementer may extend the provided WSDLs to incorporate WS-Security conformance and still be considered compliant with the specification.

The source Enterprise Architect .EAP model used to generate the XSD, as well as the XMI derived from the .EAP model, are provided on a non-normative, reference basis.

Also, please note that there has been significant interest in a RESTful Web service PSM for the DSS. Therefore, a RESTful Web service PSM in under consideration for future specification.

7.2 PSM-Specific Conformance Criteria

The PSM conformance criteria correspond to the conformance criteria defined for the PIM in Clause 6. Three separate WSDLs are provided to correspond with the two functional profiles defined in this specification, as follows:

- HSSP Simple Evaluation DSS Functional Profile, Version 1.0: dssEvaluate.wsdl
- HSSP Complete DSS Functional Profile, Version 1.0: dss.wsdl

Of note, a third WSDL (dssBaseComponents.wsdl) defines common components and is used by the two WSDLs noted above.

A - Non-normative Content

A.1 Problem Addressed by the Specification

The problem addressed by the specification is the need for a standardized approach for leveraging machine-executable medical knowledge in an application-independent manner.

Further elaboration on the problem addressed by this specification is provided below. Note that this text was taken from the OMG DSS RFP section 6.1, which in turn was derived from the HL7 DSS Service Functional Model (SFM) section 2.1.1.

In recent years, research has emerged showing that the healthcare delivered in many industrialized nations falls short of optimal, evidence-based care. In the United States, a recent nationwide audit assessing 439 quality indicators found that American adults receive only about half of recommended care,[1] and the U.S. Institute of Medicine has estimated that up to 98,000 Americans die each year as the result of preventable medical errors.[2] In the United Kingdom, a recent retrospective analysis at two London hospitals found that 10.8% of admitted patients experienced adverse events, of which 48% were judged to be preventable and of which 8% led to death.[3] Similarly in Australia, a review of medical records from 28 hospitals identified adverse events in 16.6% of admissions, of which 51% were deemed preventable and of which 4.9% led to death.[4]

One of the most promising strategies for addressing this crisis in care quality is the use of clinical decision support (CDS) systems, which are systems that provide physicians and other healthcare stakeholders with patient-specific assessments or recommendations in order to aid in clinical decision making. Examples of CDS systems include outpatient systems that attach care reminders to the charts of patients in need of specific preventive care services, computerized provider order entry (CPOE) systems that provide patient-specific recommendations as part of the order entry process, and laboratory alerting systems that page physicians when critical lab values are detected.

CDS systems can be highly effective at improving care quality and ensuring patient safety. In a recent systematic review, for example, CDS systems possessing four critical features were found to significantly improve clinical practice in 94% of randomized controlled trials.[5] Despite these promising results, however, the availability of decision support capabilities remains limited in most health care facilities in the U.S. and elsewhere. Although many barriers contribute to this limited use of decision support systems, one important barrier is the difficulty and cost associated with implementing effective decision support systems.

As with other types of applications, a CDS system could be more easily implemented and maintained if software services were available to provide functionality required by the application. Table A.1 lists some of the services that may be useful for the implementation of a CDS system, including: (i) a decision support service (DSS), which uses patient data to draw machine-interpretable conclusions regarding patients; (ii) a common terminology service (CTS), which provides access to various terminology operations; (iii) an entity identification service (EIS), which enables the identification of entities (e.g., patients) across systems; (iv) a record locator and access service (RLAS), which facilitates the retrieval of patient records across systems, and which also allows for fine-grained queries for patient data; (v) a patient record update service (PRUS), which

1. McGlynn EA, Asch SM, Adams J, et al. The quality of health care delivered to adults in the United States. *N Engl J Med*. 2003;348:2635-2645.
2. Kohn LT, Corrigan JM, Donaldson MS, eds. *To Err is Human: Building a Safer Health System*. Washington, DC: National Academy Press; 1999.
3. Vincent C, Neale G, Woloshynowych M. Adverse events in British hospitals: preliminary retrospective record review. *BMJ*. 2001;322:517-519.
4. Wilson RM. The quality in Australian Health Care Study. *Medical Journal of Australia*. 1995;163:458-71.
5. Kawamoto K, Houlihan CA, Balas EA, Lobach DF. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. *BMJ*. 2005;330:765-772.

allows the service client to update the patient record; and (vi) an electronic health record (EHR) action brokering service (EABS), which permits the service client to invoke various actions within an EHR. Of note, the patient data query service component of the RLAS, the PRUS, and the EABS comprise the primary services that an EHR would need to implement in order to provide what is known as a virtual medical record (vMR) interface.[1] *Currently, specifications for the CTS, EIS, and DSS are being actively worked on by members of the Healthcare Services Specification Project (HSSP). Also, the HSSP Retrieve, Locate, and Update Service (RLUS) encompasses the functionality of the RLAS and PRUS.*

All of the services just described facilitate the implementation of a CDS system, as they allow a CDS system to fulfill many of its functional requirements by making requests to existing services. Specifically with regard to the Decision Support Service (DSS), the service allows a CDS system to reach conclusions regarding a patient by making requests to one or more DSSs. Furthermore, the service allows a single DSS to simultaneously fulfill the patient evaluation requirements of multiple decision support applications. Because the specification and updating of machine-executable decision logic represents one of the most expensive aspects of developing and maintaining a decision support system, this arrangement could significantly reduce the effort required for a CDS system implementation. This reduction in the effort required to implement and maintain a CDS system is the primary business purpose for the DSS. It is hoped that the DSS standard will facilitate the more widespread adoption of CDS systems, which in turn should result in higher quality care and improved patient safety.

Table A.1 - Services potentially useful for the implementation of a CDS system

Service	Description	Example of Service Use by a CDS system
Decision Support Service (DSS)	Provides machine-interpretable, patient-specific assessments and recommendations given requisite data.	When a patient checks into an outpatient clinic, the clinic's EHR sends relevant patient data to the DSS, receives back the patient's care needs (e.g., overdue preventive care procedures, medication incompatibilities), and informs the clinician regarding those care needs.
Common Terminology Service (CTS)	Provides access to various terminology operations (e.g., translation of a code between vocabularies, identification of semantic relationships between codes).	When authoring a rule regarding beta-blocker use following a myocardial infarction, a knowledge engineer provides the CTS with the SNOMED CT code for the beta-blocker drug class and requests all SNOMED CT codes that are subsumed by (i.e., are descendants of) the provided code. The engineer also makes a request to the CTS to translate the SNOMED CT codes to FDA NDC codes. The SNOMED CT and NDC codes indicative of beta-blockers are used to determine whether a patient who has suffered a myocardial infarction is currently prescribed a beta-blocker.
Identity Cross Reference Service (IXS)	Allows the service client to identify entities (e.g., patients) across systems.	When determining whether a patient is in need of an influenza vaccine, a CDS system associated with Health System A uses IXSs to identify that the patient has a medical record number with the local health department, as well as with Clinic B. The CDS system provides these system-specific record numbers to the RLASs of the health department and of Clinic B, and the CDS system requests that the RLASs retrieve data on the influenza vaccination procedures the patient has received at these sites over the past year. Through this interaction, the CDS system is able to determine that the patient received a flu shot this year at the local health department. As a result, the CDS system correctly concludes that the patient is not in need of a flu shot.

1. Johnson PD, Tu SW, Musen MA, Purves I. A virtual medical record for guideline-based decision support. *Proc AMIA Symp.* 2001;294-8.

Table A.1 - Services potentially useful for the implementation of a CDS system

Record Locator and Access Service (RLAS)	Allows the service client to locate and retrieve records for a patient across systems. Allows for fine-grained record retrieval (e.g., query for lab tests for a patient from the past 3 months with LOINC codes A, B, or C).	See example above for IXS.
Patient Record Update Service (PRUS)	Allows the service client to update a patient's record.	When the hematocrit is entered into the clinical data repository for a patient being treated in the hospital, a CDS system detects that the hematocrit is critically low and sends a page to the intern responsible for the patient's care. The CDS system makes a request to the PRUS to record into the clinical data repository the details regarding the alert (e.g., when it was sent, to whom it was sent, why it was sent).
EHR Action Brokering Service (EABS)	Allows the service client to request that the EHR performs pre-specified actions.	A clinician consults a decision support module in an EHR to decide on a medication regimen for a patient with hypertension. The CDS system determines that additional data are required to reach a conclusion. The CDS system makes a request to the EABS to collect the required data from the clinician; upon receiving the request, the EABS asks the clinician for the required information through the EHR user interface. The EABS then returns the information to the CDS system so that a conclusion can be reached.

A.2 Functional Capabilities of a Decision Support Service (DSS)

A DSS can be conceptually understood as the guardian of one or more modules of medical knowledge, wherein each DSS knowledge module is capable of utilizing coded patient data to arrive at machine-interpretable conclusions regarding the patient under evaluation. To support this capability for evaluating patients using knowledge modules, a DSS also provides supplemental operations for clients to (i) identify knowledge modules meeting their business needs, (ii) to obtain information on the data required for evaluating a patient using the specified DSS knowledge modules; and (iii) to obtain a specification of the meaning and format of the patient evaluation results that will be returned by the specified DSS knowledge modules. These functional capabilities of a DSS are further elaborated in the text below. Note that this text was adapted from the HL7 DSS Service Functional Model (SFM) section 2.1.2.

A DSS can be conceptually understood as the guardian of one or more modules of medical knowledge, wherein each DSS knowledge module is capable of utilizing coded patient data to arrive at machine-interpretable conclusions regarding the patient under evaluation. The scope of a typical DSS knowledge module is the assessment of a single patient in a specified topic area. The topic area may be narrow (e.g., the need for a glycated hemoglobin test for a patient with diabetes) or broad (e.g., the existence of contraindications to any medications prescribed or about to be prescribed for a patient).

A DSS is used by a DSS client, which is alternatively referred to as a "client" or as a "client system" in this specification. A DSS client is any external entity that interacts with a DSS to obtain its services. Examples of DSS clients include a DSS query system used by an engineer to find and explore knowledge modules at design time or an operational CDS system that interacts with a DSS at run-time.

When requesting a patient evaluation, a client CDS system specifies the knowledge modules to use for the evaluation, and the CDS system also submits the patient data required by the knowledge modules. In return, the DSS returns inferences regarding the patient in a format that has been pre-defined for that knowledge module. For example, an online immunization registry might submit data on a patient's allergies and on her past immunizations to a DSS and request that the patient be evaluated using the service's immunization knowledge module. In return, the DSS might return a list of the vaccines for which the

patient is ineligible due to contraindications, a list of the vaccines for which the patient is up-to-date, and a list of the vaccines for which the patient is due.

Of note, a DSS knowledge module may or may not have a one-to-one correspondence with an underlying computational construct. For example, the immunization knowledge module just described may be implemented using one computational construct (i.e., a single construct that checks for the need for a number of vaccines) or multiple computational constructs (e.g., one construct that checks for the need for a flu vaccine, a second construct that checks for the need for a pneumococcal vaccine, etc.).

Table A.2 provides examples of the types of inferences that could be made by a DSS.

Table A.2 - Example inferences that could be made by a DSS

Sample Evaluation Input	Sample Evaluation Output
Patient age, gender, past health maintenance procedures	List of health maintenance procedures due or almost due
Medication identifier, age, gender, weight, serum creatinine level	Recommended maximum and minimum doses for medication given patient's estimated renal function
Age, gender, co-morbidities, chief complaint	Admission order set in HL7 format
Insurance provider, data relevant to prescription	Whether the prior authorization criteria for prescribing the medication are met

In order to acquire patient evaluations in this manner, a client must be able to obtain several supplemental pieces of information from a DSS. These supplemental information needs consist of the need to (i) identify the knowledge modules that could be used to meet client needs; (ii) know what patient data must be submitted to the DSS in order to obtain an accurate evaluation; and (iii) know the meaning and format of any results that will be returned by the DSS following a patient evaluation. Table A.3 lists these supplemental client information needs; a brief description is also provided for the DSS operations that meet these information needs.

Table A.3 - Supplemental information required for obtaining patient evaluations using a DSS, and brief descriptions of the service operations that provide the required information

Supplemental Information Need	Operation Providing Required Information	Description of Service Operation	Typical Usage Context
Identification of knowledge modules meeting client needs.	Find Knowledge Modules	Identifies the service's knowledge modules that meet client search criteria. It is anticipated that the search for appropriate knowledge modules will generally occur at design time.	Design-time
Information on the data required for evaluating a patient using the specified DSS knowledge modules.	Get Knowledge Module Data Requirements	Explicitly specifies the data required for evaluating a patient using the selected knowledge modules.	Design-time and run-time
Specification of the meaning and format of the patient evaluation results that will be returned by the specified DSS knowledge modules.	Get How Knowledge Module Evaluation Results Will be Returned	Provides a description of the specified knowledge module, including the content and structure of the results that will be returned when the module is used to evaluate a patient.	Design-time

Through the use of these supplemental operations, a service client is able to identify the knowledge modules that are available from one or more DSSs for meeting the service client's CDS needs. Furthermore, the service client is able to determine what data are needed for requesting a patient evaluation, as well as what will be returned by the DSS as a result of the patient evaluation request. Thus, when the need for a patient evaluation arises in a CDS system, the CDS system is able to (i) obtain the required patient data from its clinical data repositories, (ii) provide the requisite data to the DSS and request that the patient be evaluated using the specified knowledge modules, (iii) obtain machine-interpretable decision support results regarding the patient, and (iv) parse and use the results as appropriate in meeting the functional requirements of the application.

Figure A.1 illustrates this interaction graphically. Of note, all of the core information exchanged in the illustrated interactions could potentially be represented using HL7 v3 content.

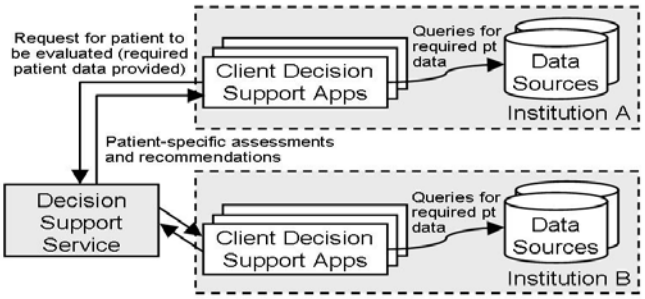


Figure A.1 - Schematic representation of interaction between clients and a DSS

As an optional feature, a DSS may allow the client to specify an analysis time other than the present when requesting a patient evaluation. This feature is useful, for example, when outpatient care reminder sheets need to be printed in batch during the business day prior to the actual clinic session. Furthermore, the ability to designate any time in the past or the future as the evaluation time significantly facilitates testing, as static test cases will not become obsolete with the passage of time. This ability to specify the time at which a knowledge module evaluation is to take place is similar to how a HL7 v3 RIM Act can be scheduled to occur at a desired point in time through the use of the "intent" mood and the specification of the relevant activityTime.

A.3 Relationship to HL7 DSS Service Functional Model (SFM)

The DSS specification is the product of a collaborative effort between Health Level 7 (HL7) and the Object Management Group (OMG). Within HL7, members of the SOA Work Group produced a functional specification document (known as the Service Functional Model – SFM). The HL7 DSS SFM upon which this OMG specification is based is Release 1 of the SFM, available at http://www.hl7.org/v3ballot2009sep/html/infrastructure/dss/dss.htm. Following adoption of the present OMG DSS technical specification, it is anticipated that the HL7 DSS SFM will be updated to be fully harmonized with the present OMG specification. The latest version of the HL7 DSS SFM is expected to be available at http://www.hl7.org/v3ballot/html/infrastructure/dss/dss.htm (note that this link currently points to Release 1 of the SFM, as specified above).

The HL7 DSS SFM was used as the basis for the requirements expressed in the RFP. Many of the RFP requirements explicitly referred to sections within the SFM, especially the definition of the capabilities required in Section 5 of the SFM. Although this specification was developed in direct response to the issued RFP, readers and viewers are advised to become familiar with the SFM as well.

A.4 Relationship to Existing OMG Specifications

This specification is related to the following OMG specifications listed in Table A.4.

Table A.4 - OMG specifications related to this specification

OMG Specification	OMG Document Number and/or URL	Relationship to Current Specification
Unified Modeling Language (UML)	http://www.omg.org/spec/UML	Used to define PIM.
XML Metadata Interchange Specification (XMI)	http://www.omg.org/spec/XMI	Used to exchange the UML models that define the PIM.
OMG Retrieve, Locate, and Update Service (RLUS) Technical Specification	http://www.omg.org/spec/RLUS	RLUS is an HSSP service for locating, retrieving, and updating clinical data. A DSS implementations' data requirements *may* be fulfilled using RLUS implementations, although this is not required. See Section 2.4.3 of the HL7 DSS SFM for a discussion of this topic.
OMG Identity Cross Reference Service (IXS) Technical Specification	http://www.omg.org/spec/IXS	IXS is an HSSP service for identifying entities (e.g., patients) across systems. An IXS *may* be used to facilitate the collection of patient data required by a DSS from across various data sources. Note that HL7 now refers to IXS simply as an Identification Service (IS).

A.5 Related Activities, Documents, and Standards

The table below summarizes some of the standards and reference content relevant to this specification, as well as the relationship of these works to the DSS standard. Note that this table is derived from Appendix I of the HL7 DSS SFM.

Table A.5 - Relevant standards, activities, and reference content and their relationships to the DSS standard

Category	Standard, Activity, or Reference Content	Relationship to the DSS Standard
Reference content – relevant prior work	SEBASTIAN	The development of the DSS SFM was informed by a Web service for clinical decision support known as SEBASTIAN (an acronym for System for Evidence-Based Advice through Simultaneous Transaction with an Intelligent Agent across a Network).[1]
Relevant standard – HL7	Version 3 Reference Information Model (RIM) and RIM-Derived Domain Content	HL7 version 3 content can be specified as service input or output parameters through the use of semantic signifiers. See Section 2.1.4 of the HL7 DSS SFM for an in-depth discussion of the use of HL7 v3 domain content by DSS implementations.

Table A.5 - Relevant standards, activities, and reference content and their relationships to the DSS standard

Relevant standard – HL7	Arden Syntax	The Arden Syntax is a HL7 standard for representing executable medical knowledge. A DSS implementation could potentially use Arden Syntax Medical Logic Modules (MLMs) to analyze patient data and generate patient-specific inferences.
Relevant standard – HL7	HSSP Service Specification Framework (SSF)	Main guide for generating HL7 SFMs. Adaptation of the HL7 Development Framework (HDF) for the purpose of generating functional service specifications.
Relevant standard – HL7	Retrieve, Locate, and Update Service (RLUS) Service Functional Model [RLUS-SFM]	The RLUS SFM is an HSSP functional model for locating, retrieving, and updating clinical data. The DSS SFM is specified so that DSS implementations' data requirements can be fulfilled in a straightforward manner by using RLUS implementations. See Section 2.4.3 of the HL7 DSS SFM for a discussion of this topic.
Relevant standard – HL7	CDS TC GELLO standard	Medical knowledge encoded in GELLO could potentially be exposed to clients using a DSS interface.
Relevant standard – HL7	CDS TC standard for context-sensitive reference information retrieval (Infobutton standard)	The capabilities of this standard could be exposed through a DSS interface. See the business scenario in Sections 3.3.2.1 and 7.2.2.1 of the HL7 DSS SFM for details. Also, work is currently ongoing within HL7 to specify a DSS profile for supporting this capability.
Relevant standard – ASTM International	Continuity of Care Record (CCR) standard	A DSS implementation could specify that patient data should be provided as service inputs using ASTM International's CCR.
Relevant standard – ASTM International and HL7	Continuity of Care Document (CCD) implementation guide for HL7 Clinical Document Architecture (CDA)	A DSS implementation could specify that patient data should be provided as service inputs using CCD.
Relevant standards development activity – OMG	OMG Decision Model and Notation standard specification project (overview presentation available to OMG members at http://www.omg.org/members/cgi-bin/doc?bmi/09-06-09.pdf)	This specification could potentially be of use for a DSS implementer.

1. Kawamoto K and Lobach DF. Design, Implementation, Use, and Preliminary Evaluation of SEBASTIAN, a Standards-Based Web Service for Clinical Decision Support. *Proc AMIA Symp*. 2005;380-4.

A.6 Overall Design Rationale

The primary design decisions regarding the DSS have already been made previously during the specification of the HL7 DSS SFM. During its specification, primary design principles included the following:

- Make DSS an application-independent service with as few external dependencies as possible. As a corollary, design the DSS to be compatible with other services (e.g., OMG Retrieve, Locate, and Update Service), but do not turn such potential for coordination into a dependency for the DSS.

- Be minimalist in design. Specifically, do not include within the DSS capabilities that can be handled by other independent services (e.g., RLUS, IXS).
- Allow flexibility for DSS semantics, but specify a mechanism for constraining these semantics within a given interoperability context (realized as DSS semantic profiles).
- Do not constrain how machine-executable medical knowledge is represented within a service.
- Standardize only what is necessary; allow flexibility for implementers where possible unless such flexibility would hinder interoperability.
- Enable a DSS to be utilized across regional and national boundaries.
- Ensure ease of implementation and use whenever feasible.
- Ensure that one conformant service implementation can be replaced with another meeting the same service specification while maintaining functionality of the system.

The above principles were carried forward in the present specification activities.

A.7 Proof of Concept

Several submitting vendors have implemented operational Decision Support Services from which the current consensus specification was derived. The current specification was designed to allow these existing implementations to be efficiently adapted to the standard specification. Several submitting vendors have implemented, or are actively implementing, clinical decision support services compliant with the standard interfaces defined in this specification.

B - Description of Associated Machine Consumable Files

File(s)	Status	Description
Normative Content\PIM\dss_xmi.xml	Normative	XMI file of DSS PIM
Normative Content\PSM\dss.wsdl	Normative	WSDL file of DSS PSM for SOAP XML Web services. Supports the complete functional profile.
Normative Content\PSM\ dssEvaluate.wsdl	Normative	WSDL file of DSS PSM for SOAP XML Web services. Supports the simple evaluation functional profile.
Normative Content\PSM\baseWsdl\ dssBaseComponents.wsdl	Normative	Abstract base WSDL file containing WSDL type and message definitions. Used by the WSDLs above.
Normative Content\PSM\ baseWsdl\OmgDssSchema.xsd	Normative	XSD file of DSS PSM, for use by DSS WSDL
Files in Normative Content\Schemas\hl7v3schemas	Normative	XSD files for normative Health Level 7 version 3 information models obtained from http://www.hl7.org/memonly/downloads/v3edition.cfm#V32008 and used by the HSSP Minimum DSS Trait Set Requirement, Version 1.0 (see Section 6.10.5 of primary specification for details).
Normative Content\Schemas\hsspschemas\ OmgDssTraitSchema.xsd	Normative	XSD file used the by HSSP Minimum DSS Trait Set Requirement, Version 1.0 (see Section 6.10.5 of primary specification for details)
Informative Content\PIM\DSS.EAP	Informative	Enterprise Architect UML model for PIM used to generate normative XMI file for PIM
Informative Content\PSM\ DSS_XML_PSM.EAP	Informative	Enterprise Architect UML model for PSM used to generate normative XSD file for DSS PSM
Informative Content\PSM\ DSS_XML_PSM_XMI.xml	Informative	XMI file generated from DSS_XML_PSM.EAP
Informative Content\PSM\ RDDLNamespaceDocuments\dss.html	Informative	RDDL file describing the http://www.omg.org/spec/CDSS/201012/dss namespace
Informative Content\PSM\RDDLNamespaceDocuments\dss Wsdl.html	Informative	RDDL file describing the http://www.omg.org/spec/CDSS201012/dssWsdl namespace
Informative Content\PSM\ RDDLNamespaceDocuments\dssTraits.html	Informative	RDDL file describing the http://www.omg.org/spec/CDSS/201012/dssTraits namespace