V3_DSS_R2_2013SEP

HL7 Version 3 Standard: Decision Support Service (DSS), Release 2

DSTU Ballot, September 2013

Project Lead and Editor:
Kensaku Kawamoto, MD, PhD; University of Utah

Contributors:
Ohad Young; dbMotion
David Boaz; dbMotion
David Shields; University of Utah
Bryn Rhodes; Veracity Solutions
Brett Esler; Pen Computer Systems
Ken Rubin; Hewlett-Packard
Manfred Koethe, MS; 88solutions
Alean Kirnak; Software Partners
Clayton Curtis, MD, PhD; Veterans Health Administration
Robert Hausam, MD; OntoReason
Alan Honey, PhD; ii4sm

Project Sponsor:
HL7 CDS Work Group

Table of Contents

1 Preface

A decision support service (DSS) receives patient data as the input and returns patient-specific conclusions as the output. As such, it can significantly facilitate the implementation of systems that require patient-specific inferencing, such as clinical decision support (CDS) systems and quality reporting systems. The present DSS specification represents one member of a family of healthcare service specifications being jointly developed by HL7 and the Object Management Group (OMG) through the Healthcare Services Specification Project (HSSP).[1,2] As per the HSSP process, the required capabilities of a DSS were identified in an HL7 Draft Standard for Trial Use (DSTU) in December 2006 (the *HL7 DSS Service Functional Model, Version 1.0*).[3] Subsequently, a fully implementable DSS specification based on the HL7 DSTU was developed by the OMG Healthcare Domain Task Force. This OMG DSS specification was first adopted as an OMG draft standard and then recommended for adoption as a normative OMG standard in December 2010 (the *OMG CDSS Specification, Version 1.0*).[4,5] Within OMG, the DSS standard is known as the Clinical Decision Support Service (CDSS) standard to indicate its focus on the clinical domain. In this document, the terms DSS and CDSS are used interchangeably when referring to the OMG DSS specification. The OMG DSS standard was developed in close consultation with members of the HL7 CDS Work Group to ensure that the DSS specification could be brought back into the HL7 community following completion of the OMG standardization process.

This present specification is a normative HL7 specification that is based on the normative OMG DSS standard, with minor revisions that are clearly noted in Section 2. These minor revisions are expected to be introduced back into the OMG standard, as the intent is for the HL7 and OMG DSS standards to be semantically interoperable. Like the OMG standard, the present DSS specification includes a platform-independent model (PIM) for the DSS as well as a platform-specific model (PSM) for SOAP XML Web services. The original HL7 DSS DSTU was a functional specification that described the behavior of a DSS but was intentionally not specified at a technical level required for implementation. The present HL7 DSS normative specification includes technical details required for implementation, including WSDLs and XSDs for implementing a DSS as a SOAP Web service. A catalogue of these machine-consumable files can be found in Appendix I of this document.

Of note, an open-source implementation of this specification will be made available through the multi-institutional OpenCDS effort (http://www.opencds.org).

[1] Kawamoto, K., Honey, A. and Rubin, K. (2009). The HL7-OMG Healthcare Services Specification Project: motivation, methodology, and deliverables for enabling a semantically interoperable service-oriented architecture for healthcare. *Journal of the American Medical Informatics Association* 16, 874-81.

[2] Healthcare Services Specification Project (HSSP) Homepage. Available at http://hssp.wikispaces.com. Accessed August 1, 2013.

[3] Health Level 7 (HL7). HL7 Service Functional Model Specification – Decision Support Service (DSS). HL7 Draft Standard for Trial Use - Release 1. Available at http://www.hl7.org/dstucomments/ex_showdetail.cfm?dstuid=12. Accessed August 1, 2013.

[4] Object Management Group (OMG). OMG Clinical Decision Support Service Specification, Version 1.0. Available at http://hssp-dss.wikispaces.com/omg_specification. Accessed August 1, 2013.

[5] Object Management Group (OMG). Machine-Readable Files for OMG Clinical Decision Support Service Specification, Version 1.0. Available at http://hssp-dss.wikispaces.com/omg_specification. Accessed August 1, 2013.

2 Revision History

2.1 Revisions of HL7 DSS Release 1 Specification Compared to OMG CDSS Version 1.0 Specification

At the core of this specification are the Platform Independent Model (PIM) and Platform Specific Model (PSM) for the DSS, along with the accompanying machine-readable files. These core aspects of the present specification are semantically identical to the OMG CDSS Version 1.0 specification, except as specified below. It is expected that these revisions will be brought back into the OMG community in the future to ensure full alignment of the OMG and HL7 DSS standards.

- Revision 1: addition of DSSRuntimeException to all operations
    - In order to account for runtime errors not otherwise covered by other named exceptions, a DSSRuntimeException has been added to all DSS operations.

- Revision 2: correction of business IDs of trait and trait criteria semantic signifiers
    - The semantic signifiers of trait and trait criteria specified in Section 6.10.5 have been corrected so that their business IDs include root global element names, which had been omitted in error. For example, for the Explanation trait, its semantic signifier's business ID is now HsspDssTraitSchema.Explanation instead of HsspDssTraitSchema.

- Revision 3: clarification of KMItem description
    - The description of KMItem has been clarified to note that its scoping entity may be a knowledge module or another entity. The exact change is as follows:
    - Original description:
        - The superclass of all knowledge module sub-items. For example, Data Requirement Group or Item. It contains the item identifier, which consists of the identifier of the knowledge module as well as a unique identifier within the knowledge module. It inherits name and description information from DescribedDO.
    - Revised description:
        - The superclass of all knowledge module sub-items. For example, Data Requirement Group or Item. It contains the item identifier, which consists of the identifier of the scoping entity as well as a unique identifier within the scoping entity. The scoping entity may be the knowledge module within which the KMItem resides, a different knowledge module, or an entity other than the knowledge module. This approach enables items such as data requirements to be decoupled from specific knowledge modules and reused across knowledge modules. This class inherits its name and description information from DescribedDO.

- Revision 4: updating of versioned namespaces of machine-consumable files
    - Where the above changes resulted in updates to machine-consumable files, versioned namespaces have been updated to use the date-based version of "201105" instead of "201012".

Furthermore, while the OMG CDSS Version 1.0 specification includes informative Resource Directory Description Language (RDDL) files to describe its XML namespaces, this specification does not include RDDL files. RDDL files are not included as the use of such resource descriptors are expected within the OMG community but not within the HL7 community.

2.2 Revisions of HL7 DSS Release 2 Specification Compared to HL7 DSS Release 1 Specification

- Added a responseId element to the iterative and final response messages.
    - This enables logging of outputs to be traced separately for each requested Knowledge Module.

- Added a new profile to support RESTful Web services for the Evaluation interface as an alternative to the original SOAP interface.
    - No changes were made to any defined operations, and the new RESTful interface is functionally identical to the SOAP interface.

- Extended the DSS Evaluate Profile to include both the evaluate operation and the evaluateAtSpecifiedTime operation.

- Added a requestId and responseId to all evaluation response messages.
    - The requestId is identical to the interactionId provided as part of the associated request.
    - The responseId is a system-generated identifier for tracking and trouble-shooting purposes.

- Updated HL7 version 3 schemas referenced in the specification to the 2012 Normative Edition.

- Added optional CDS output specification to KM Evaluation Result Semantics. Updated Semantic Requirement model to reflect this change (EvaluationResultRequirementBase, AllowedEvaluationResultRequirement, and RequiredEvaluationResultRequirement).

- Updated profile version numbers and dates in schema namespaces.

- Added to responses to all operations a requestId and a responseId.

3 Acronyms

There are a number of acronyms used in this document.

The following is a brief list of what the most common ones stand for.

Acronym	Full Name
ANSI	American National Standards Institute (U.S.A.)
CDS	Clinical decision support
CDSS	Clinical Decision Support Service (synonymous with DSS)
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
RM-ODP	Reference Model of Open Distributed Processing defined by ISO
SDO	Standards Development Organization
SFM	Service Functional Model
UML	Unified Modeling Language

4 Service Overview and Business Case

4.1 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 this targeted problem is provided below.

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 nationwide audit assessing 439 quality indicators found that American adults receive only about half of recommended care,[6] 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.[7] In the United Kingdom, a 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.[8] 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.[9]

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.[10] 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 4.1** lists some of the services that may be useful for the implementation of a CDS system, including: (i) a 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 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 HL7-OMG Healthcare Services Specification Project (HSSP) has developed or is developing standards for a number of these services.[11]

[6] 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.
[7] Kohn LT, Corrigan JM, Donaldson MS, eds. *To Err is Human: Building a Safer Health System*. Washington, DC: National Academy Press; 1999.
[8] Vincent C, Neale G, Woloshynowych M. Adverse events in British hospitals: preliminary retrospective record review. *BMJ*. 2001;322:517-519.
[9] Wilson RM. The quality in Australian Health Care Study. *Medical Journal of Australia* 163:458-71, 1995.
[10] 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.
[11] Healthcare Services Specification Project (HSSP) Homepage. Available at http://hssp.wikispaces.com. Accessed August 1, 2013.

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 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 4.1. 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.
Entity Identification Service (EIS)	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 EISs 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.
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 EIS.

Service	Description	Example of Service Use by a CDS system
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.

4.2 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. 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 4.2 provides examples of the types of inferences that could be made by a DSS.

Table 4.2. 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 4.3 lists these supplemental client information needs; a brief description is also provided for the primary DSS operations that meet these information needs.

Table 4.3. Supplemental information required for obtaining patient evaluations using a DSS, and brief descriptions of the primary service operations that provide the required information.

Supplemental Information Need	Primary Operation Providing Required Information	Description of Service Operation
Identification of knowledge modules meeting client needs	findKMs	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.
Information on the data required for evaluating a patient using the specified DSS knowledge modules	getKMDataRequirements	Explicitly specifies the data required for evaluating a patient using the selected knowledge modules
Specification of the meaning and format of the patient evaluation results that will be returned by the specified DSS knowledge modules	getKMEvaluationResultSemantics	Specifies how evaluation results will be returned when the module is used to evaluate a patient

KM = knowledge module

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 4.1** illustrates this interaction graphically.

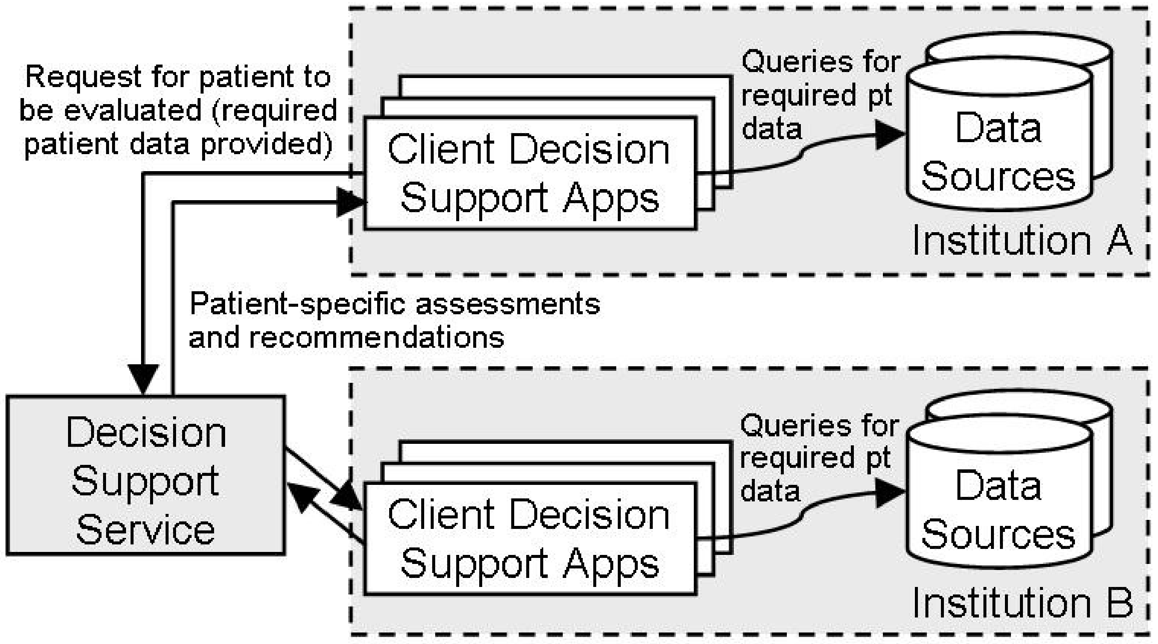

Request for patient to be evaluated (required patient data provided)
Client Decision Support Apps
Queries for required pt data
Data Sources
Institution A
Patient-specific assessments and recommendations
Decision Support Service
Client Decision Support Apps
Queries for required pt data
Data Sources
Institution B

Figure 4.1. 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.

4.3 Overall Potential Scope of the Service

The primary functionality provided by a DSS is the receipt of patient data as the input and the return of patient-specific conclusions as the output. A DSS also provides supplemental operations to support this patient evaluation functionality (**Table 4.3**).

Of note, a DSS could be used to evaluate entities other than individual patients (e.g., patient populations). For example, a DSS could be used to evaluate a group of patients to assess whether there are any indications of an emerging infectious disease outbreak within that population. However, the primary focus of this specification lies in the use of a DSS to evaluate individual patients.

As described earlier in Section 4.2, the scope of a DSS knowledge module may be narrow (e.g., an infant’s need for a varicella vaccination) or broad (e.g., a patient’s need for any general health maintenance procedures). With regard to the data required for generating the inferences, a DSS knowledge module may require the provision of various types of data. These data requirements may include, but are not limited to, demographic data, data on healthcare acts (e.g., procedures), and data on context (e.g., whether the patient is currently being seen in an outpatient or inpatient setting, or whether a specific diagnostic test can be performed at the current health care facility).

A DSS is permitted to return evaluation results using a variety of information constructs. Information constructs that may be used for communicating decision support results may include, but are not limited to, RIM acts (e.g., an HL7 medication entity with a mood code indicating that the medication should be ordered), dates (e.g., the date that a test was last performed, or the date at which a test will be due), and Boolean values (e.g., whether a patient is in need of a pneumococcal vaccine).

Of note, for both DSS inputs and outputs, the HL7 Virtual Medical Record (vMR) standard may be used as DSS payloads.[12]

With regard to the types of applications intended to be supported by the specification, the DSS is designed primarily to facilitate the implementation and maintenance of systems that assist patient-specific decision making by clinicians (i.e., CDS systems targeted to clinicians). However, the DSS could be used to support other types of applications as well. For example, the inferences obtained from a DSS could be used to generate care reminder letters for patients. Also, patient evaluation requests could be repeated across a patient population in order to obtain population-level statistics. For example, by asking a DSS provider to evaluate each diabetic patient in a clinic with regard to the patient's compliance with diabetes care guidelines, a reporting system would be able to easily calculate the proportion of diabetic patients in the clinic in compliance with the recommended care metrics. Finally, a DSS could be adapted so that it returns reference information relevant to the care of a patient rather than an assessment or a recommendation regarding the patient.

Of note, several aspects of the DSS are considered to be out of scope in terms of formal specification and standardization. To begin, what a client does with the evaluation result provided by a DSS is considered out of scope for the purposes of standardization. In addition, the mechanism used by a client to obtain the data required for evaluating a patient is also considered to be out of scope. Finally, the mechanism used by a DSS to generate patient-specific evaluation results is also considered out of scope for the purposes of standardization. As long as a DSS meets the functional requirements of the service, it is free to use whatever knowledge representation formalism it believes is most appropriate when implementing its decision support capabilities.

4.4 Reason Why the Service Specification is Needed

4.4.1 Explanation of Why the Service is Needed

As discussed in section 4.1, the rationale for creating the DSS specification is as follows: (i) there is a great need to improve the quality and safety of health care; (ii) CDS systems represent one of the most promising strategies for improving care quality, but their use is limited; (iii) one important reason for this limited utilization is the difficulty and cost associated with implementing effective CDS systems; (iv) the widespread adoption of the DSS standard should reduce the cost of implementing and maintaining a CDS system, thereby increasing the utilization of CDS systems in clinical care; and (v) increased utilization of CDS systems should help to improve care quality and to ensure patient safety.

4.4.2 Explanation of Why a Standard is Needed for the Service

Without a commonly agreed upon standard for the DSS, service clients would need to implement different interfaces when dealing with different DSSs or when switching DSSs. Similarly, the lack of a standard would result in service providers reaching only a small fraction of potential clients, as clients would need to invest in provider-specific interfaces before being able to make use of the functionality offered by a DSS.

A commonly accepted standard for the DSS would make it more attractive for service clients to invest in the infrastructure required for using the DSS to meet its decision support needs, as they would be able to use the same interface to interact with multiple service vendors. From the vendor's perspective as well, a standard for the DSS is needed to increase the pool of potential customers and to reduce the risk involved with investing in the service framework.

[12] Health Level 7. HL7 Virtual Medical Record (vMR) Project Wiki. Available at http://wiki.hl7.org/index.php?title=Virtual_Medical_Record_(vMR). Accessed August 1, 2013.

4.4.3 Vendor Viewpoint and Potential Business Opportunity or Niche

From the perspective of a vendor, the acceptance of the DSS as an HL7 standard would provide several important benefits. First, a DSS standard should lead to a significant expansion of their potential client base. This expansion should occur due to the fact that clients should be more inclined to invest in the interface required for interacting with a vendor, as the interface would no longer be vendor-specific and would be re-usable for accessing knowledge from other DSS providers. Given this expansion of the client base, and given the highly scalable nature of the service framework, a DSS provider that achieves economies of scale would be able to increase its revenues and earnings, maintain a high quality of service, and lower the price that it charges on a per-client basis.

Also, because the DSS can be implemented as a service wrapper around existing capabilities, a DSS provider that has an established client base would be able to continue providing knowledge services using existing approaches as well (e.g., via existing application programming interfaces). Furthermore, because the DSS does not dictate how knowledge should be represented, the DSS provider can continue to encode medical knowledge using the approach that it deems to be most appropriate for its purposes.

Finally, a vendor with a superior DSS implementation would be able to license its implementation of the framework to other knowledge vendors or to other entities interested in sharing its patient evaluation capabilities via a DSS. Non-vendor entities that may wish to become DSS providers include federal or state organizations with an interest in finding more effective ways of getting their recommendations implemented in practice, such as Medicare, Medicaid, and the Agency for Healthcare Research and Quality in the United States. Also, health systems may be interested in supplementing the knowledge available from commercial DSS providers by acting as a DSS provider itself.

4.4.4 Consumer Viewpoint and the Value Offered by the Work Product

DSS consumers would potentially include any entity that wishes to facilitate the implementation and maintenance of decision support systems. These service consumers may include EHR, CPOE, e-Prescribing, and hospital information system (HIS) vendors, as well as healthcare institutions and their clinical departments.

From the viewpoint of these DSS customers, the primary value offered by the work product is the ability to leverage the decision support capabilities of multiple DSSs through a common service interface. By leveraging the capabilities of the DSS, the service customer should be able to reduce the cost and difficulty associated with developing and maintaining decision support applications.

Another advantage of the DSS framework is that it is highly secure. The DSS client rather than the DSS retrieves all patient data, and there is usually no need for a DSS to know who is being evaluated. Consequently, patient identifiers do not need to be sent to a DSS. Also,the DSS does not need to be authorized to access the client's data repositories. As a result, a high level of security is attained by the fact that the client controls all read/write calls to its clinical repositories.

A final advantage to the DSS framework is that it significantly facilitates CDS system maintenance, as the medical decision logic used to reach patient evaluations are encapsulated in discrete knowledge modules that are tagged with meta-data, version-controlled, and maintained centrally on behalf of multiple clients.

5 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*

Web Services Description Language (WSDL) 2.0 W3C 26 June 2007 *http://www.w3.org/ns/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, 2012 Edition. Available free to the public at http://www.hl7.org/implement/standards/product_brief.cfm?product_id=282.

HL7 Decision Support Service, Release 1, Service Functional Model Specification. Available free to the public at http://www.hl7.org/dstucomments/ex_showdetail.cfm?dstuid=12.

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

6 DSS Platform Independent Model

The Platform Independent Model (PIM) for 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 section to clarify and provide guidance on this model.

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

6.1 Foundational Model Elements

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

6.1.1 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).

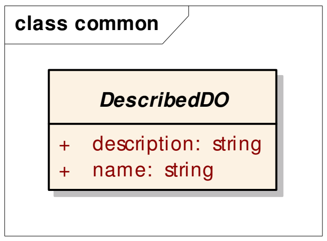


Figure 6.1. Model for Described Data Object.

6.1.2 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 uniquely identified, 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 Section 5, 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.

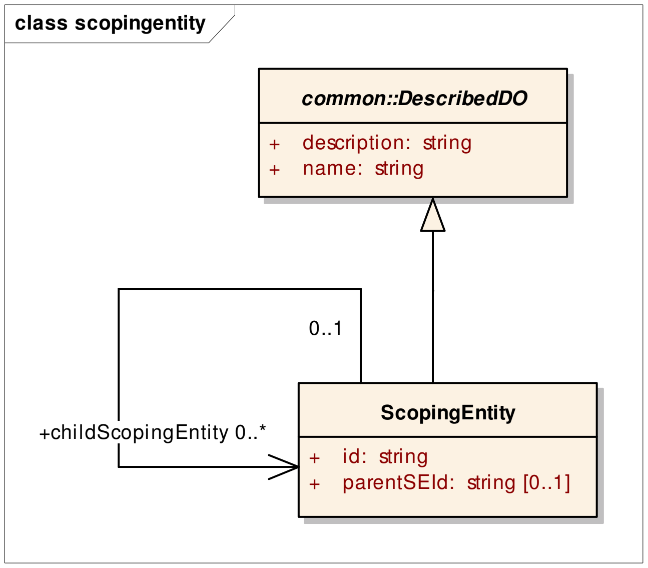


Figure 6.2. Model for Scoping Entity.

6.1.3 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 Section 6.3.1.2, Knowledge Module Version. This class is defined in the common package.

The Interaction Identifier (InteractionIdentifier) represents information that is transmitted as a part of an interaction with a DSS to identify that interaction for logging and debugging purposes. The InteractionIdentifier consists of a scopingEntityId, as well as an interactionId that is unique within the scopingEntityId. A submissionTime is also provided to help identify the interaction. This class is defined in the common package.

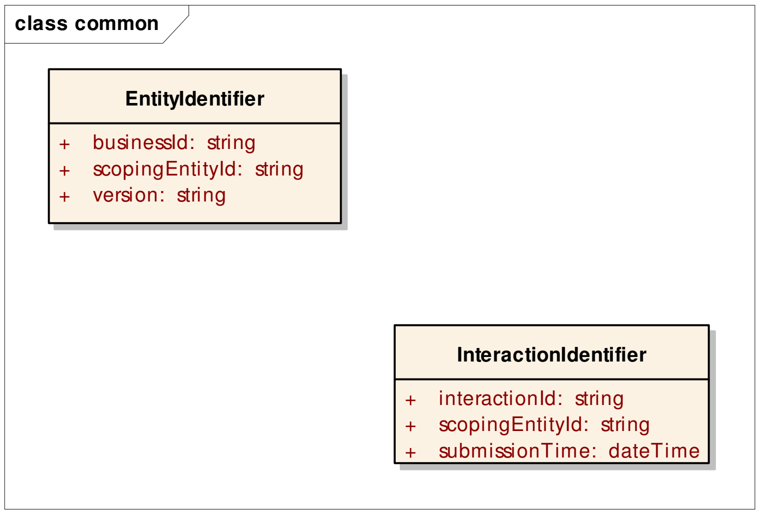


Figure 6.3. Model for Entity Identifier and Interaction Identifier.

6.1.4 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.

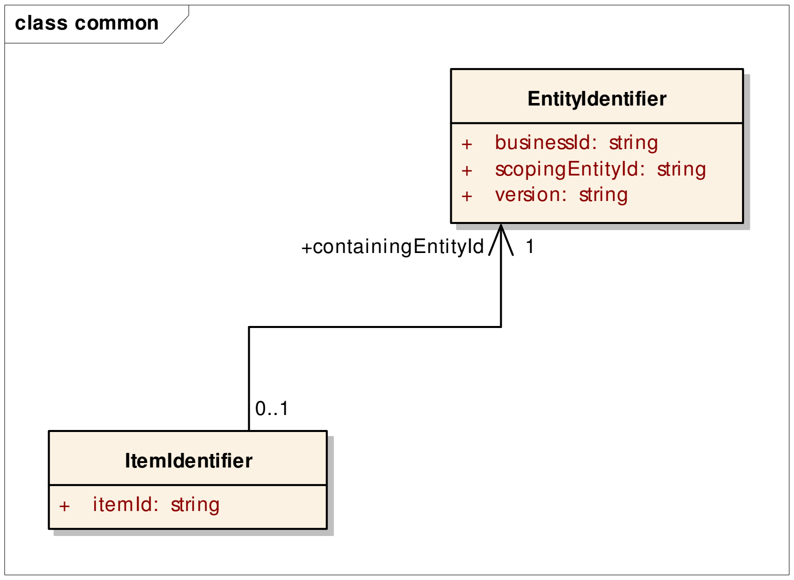


Figure 6.4. Model for Item Identifier.

6.1.5 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.

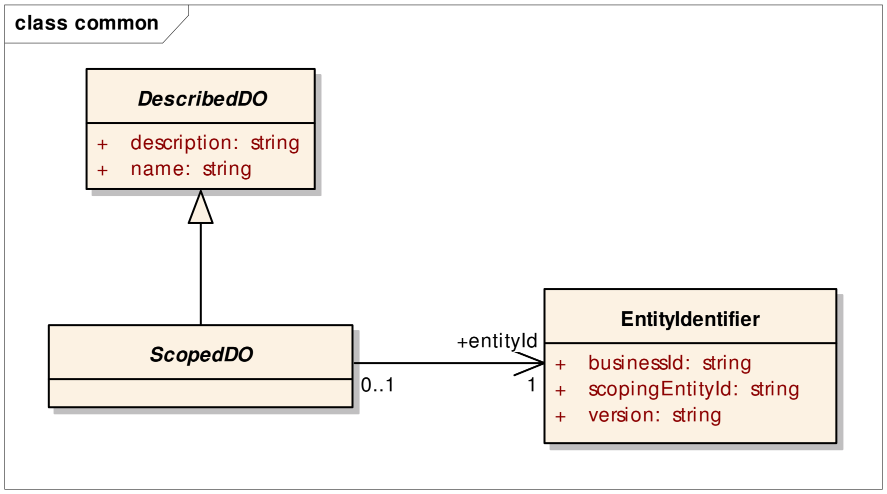


Figure 6.5. Model for Scoped Data Object.

6.2 Metadata Model Elements

This section describes the model elements related to service metadata.

6.2.1 Service Profile

The Service Profile (ServiceProfile) is an abstract class that represents a service profile. This class is defined in the metadata.profile package.

6.2.1.1 Overview of Service Profiles

By design, this specification is designed as a generic service framework which 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.

6.2.1.2 Profile Types

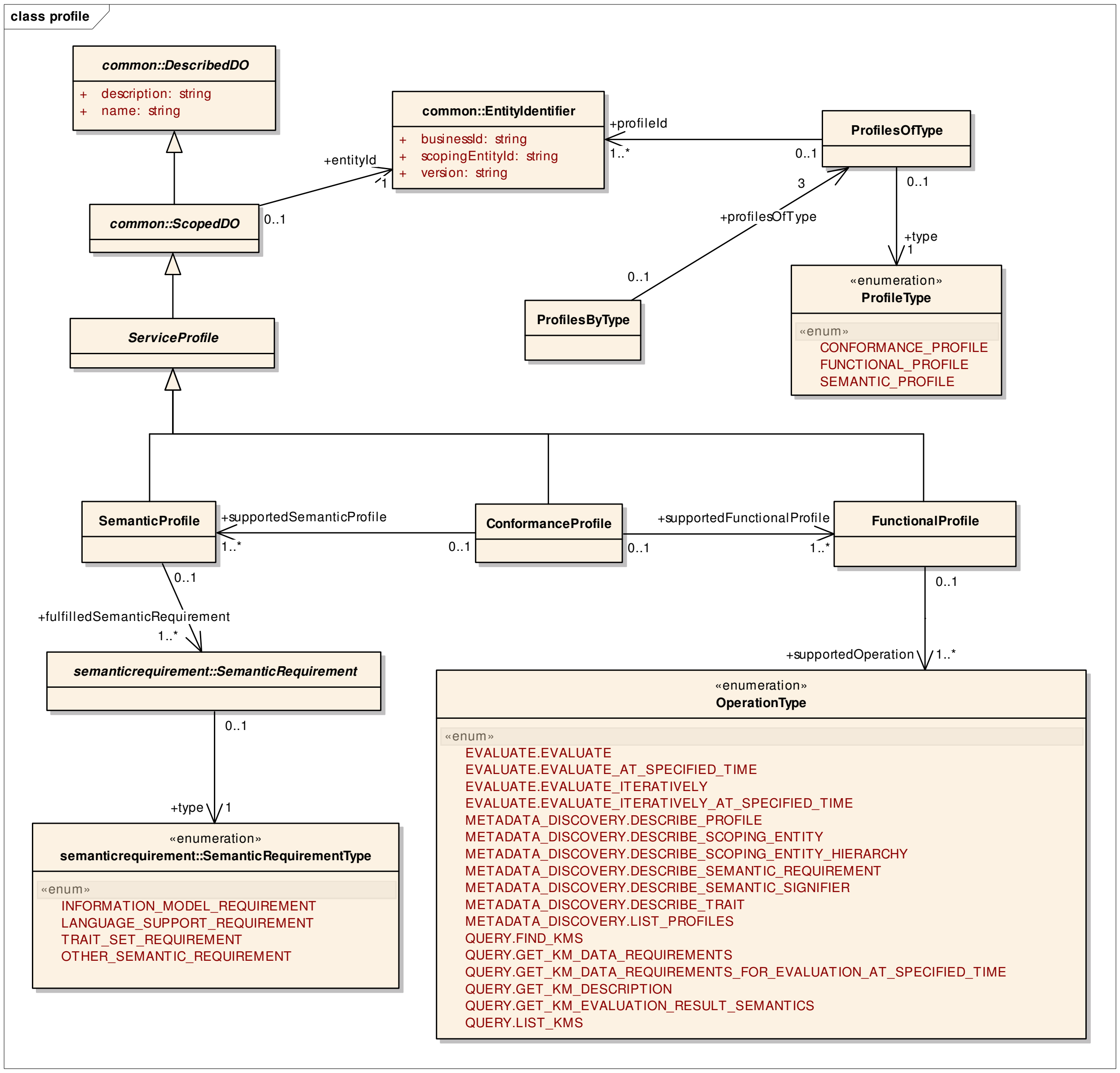
Table 6.1 summarizes the types of profiles that may be specified.

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 Section 6.3.2, Semantic Requirement).
Conformance profile	Specifies a list of one or more supported functional profiles and one or more supported semantic profiles.

6.2.1.3 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.



class profile
common::DescribedDO
+ description: string
+ name: string
common::EntityIdentifier
+ businessId: string
+ scopingEntityId: string
+ version: string
+entityId
1
0..1
+profileId
1..*
0..1
ProfilesOfType
3
+profilesOfType
0..1
+type
1
common::ScopedDO
ServiceProfile
ProfilesByType
0..1
«enumeration»
ProfileType
«enum»
CONFORMANCE_PROFILE
FUNCTIONAL_PROFILE
SEMANTIC_PROFILE
SemanticProfile
+supportedSemanticProfile
1..*
0..1
ConformanceProfile
+supportedFunctionalProfile
0..1
1..*
FunctionalProfile
0..1
+fulfilledSemanticRequirement
1..*
0..1
+supportedOperation
1..*
semanticrequirement::SemanticRequirement
0..1
+type
1
«enumeration»
semanticrequirement::SemanticRequirementType
«enum»
INFORMATION_MODEL_REQUIREMENT
LANGUAGE_SUPPORT_REQUIREMENT
TRAIT_SET_REQUIREMENT
OTHER_SEMANTIC_REQUIREMENT
«enumeration»
OperationType
«enum»
EVALUATE.EVALUATE
EVALUATE.EVALUATE_AT_SPECIFIED_TIME
EVALUATE.EVALUATE_ITERATIVELY
EVALUATE.EVALUATE_ITERATIVELY_AT_SPECIFIED_TIME
METADATA_DISCOVERY.DESCRIBE_PROFILE
METADATA_DISCOVERY.DESCRIBE_SCOPING_ENTITY
METADATA_DISCOVERY.DESCRIBE_SCOPING_ENTITY_HIERARCHY
METADATA_DISCOVERY.DESCRIBE_SEMANTIC_REQUIREMENT
METADATA_DISCOVERY.DESCRIBE_SEMANTIC_SIGNIFIER
METADATA_DISCOVERY.DESCRIBE_TRAIT
METADATA_DISCOVERY.LIST_PROFILES
QUERY.FIND_KMS
QUERY.GET_KM_DATA_REQUIREMENTS
QUERY.GET_KM_DATA_REQUIREMENTS_FOR_EVALUATION_AT_SPECIFIED_TIME
QUERY.GET_KM_DESCRIPTION
QUERY.GET_KM_EVALUATION_RESULT_SEMANTICS
QUERY.LIST_KMS

Figure 6.6. Model for Service Profile.

6.2.2 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.

6.2.2.1 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).

6.2.2.2 Use of Semantic Signifier within DSS

In this specification, semantic signifiers are used for the following purposes: (1) to specify the semantics by which data should be provided to the DSS for evaluating patients using a knowledge module; (2) to specify the semantics by which the query conditions for knowledge module data requirements are expressed by the DSS; (3) to specify the semantics by which patient evaluation results will be returned by the DSS; (4) to specify the semantics related to the traits and trait search criteria of DSS knowledge modules; and (5) to specify the semantics by which warnings are provided related to knowledge module evaluations.

6.2.2.3 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 Section 5, Normative References for normative references to the XSD, Schematron, and URL standards.

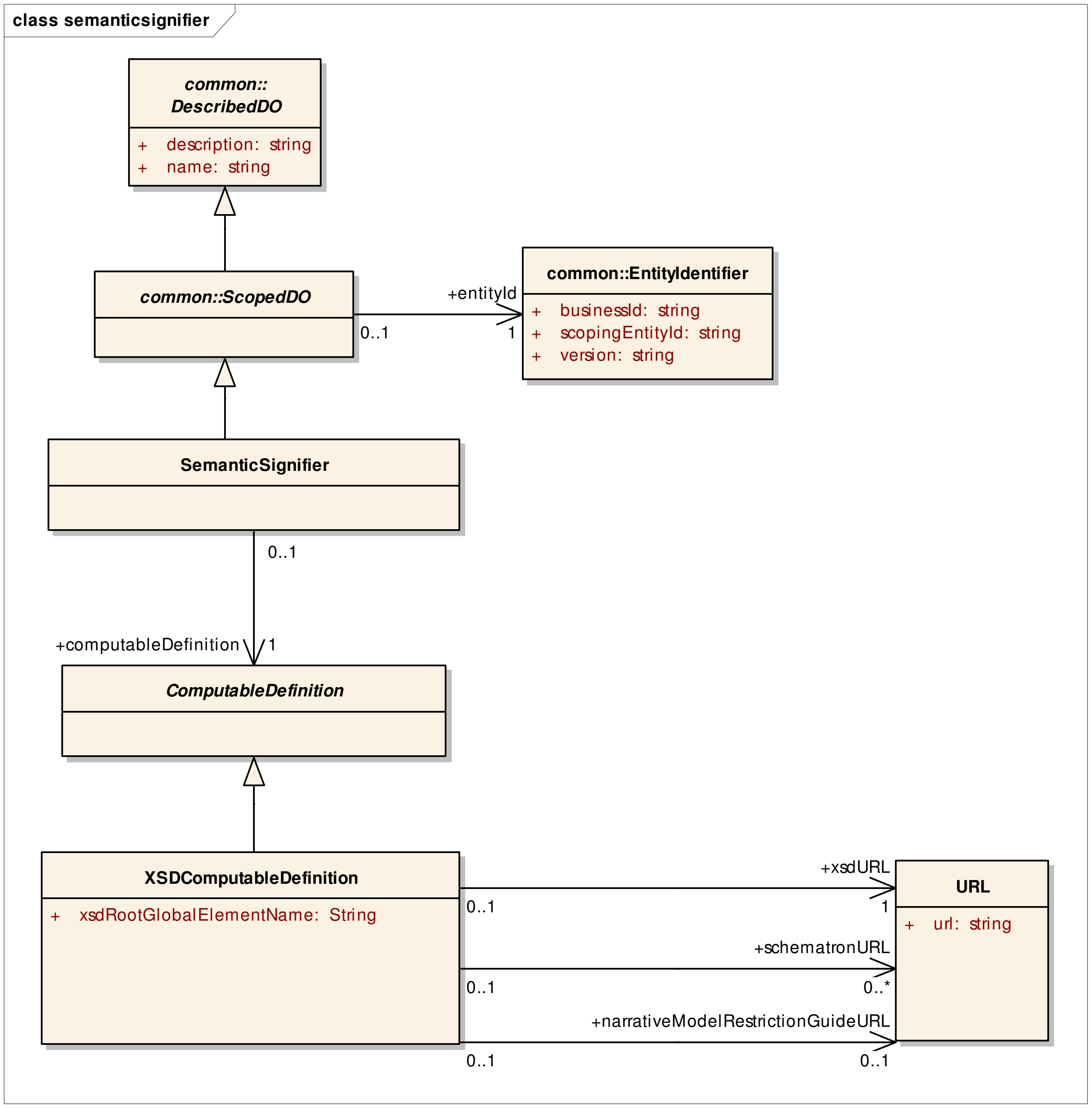


Figure 6.7. Model for Semantic Signifier.

6.2.2.4 Convention for Referring to HL7 Version 3 Semantic Signifiers

For referring to HL7 version 3 semantic signifiers, the use of the following convention is recommended. Making these recommendations mandatory will be considered in the future.

- For the scoping entity identifier, use “org.hl7”
- For the business identifier, use the following:
  - First part: “v3” followed by the folder tree in which the schema is included in the HL7 Normative Edition of the Version 3 standards. For example, in the HL7 Version 3 Normative Edition for 2012, the first part of the business identifier would be as follows:
    - **v3.infrastructure.cda** for CDA.xsd, because its folder path is “infrastructure” and then “cda”
    - **v3.processable.multicacheschemas** for HL7 version 3 messaging schemas, because their folder path is “processable” and then “multicacheschemas”
  - Second part: 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 this part of 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.
- An example using this convention is as follows:

Schema	COCT_HD010000UV01
Schema Location within HL7 Normative Edition	processable\multicacheschemas
Complex Type of Interest	COCT_HD010000UV01.Encounter
Scoping Entity Identifier	org.hl7
Business Identifier	v3.processable.multicacheschemas.COCT_HD010000UV01.COCT_HD010000UV01.Encounter

6.3 *Knowledge Module Model Elements*

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

6.3.1 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.

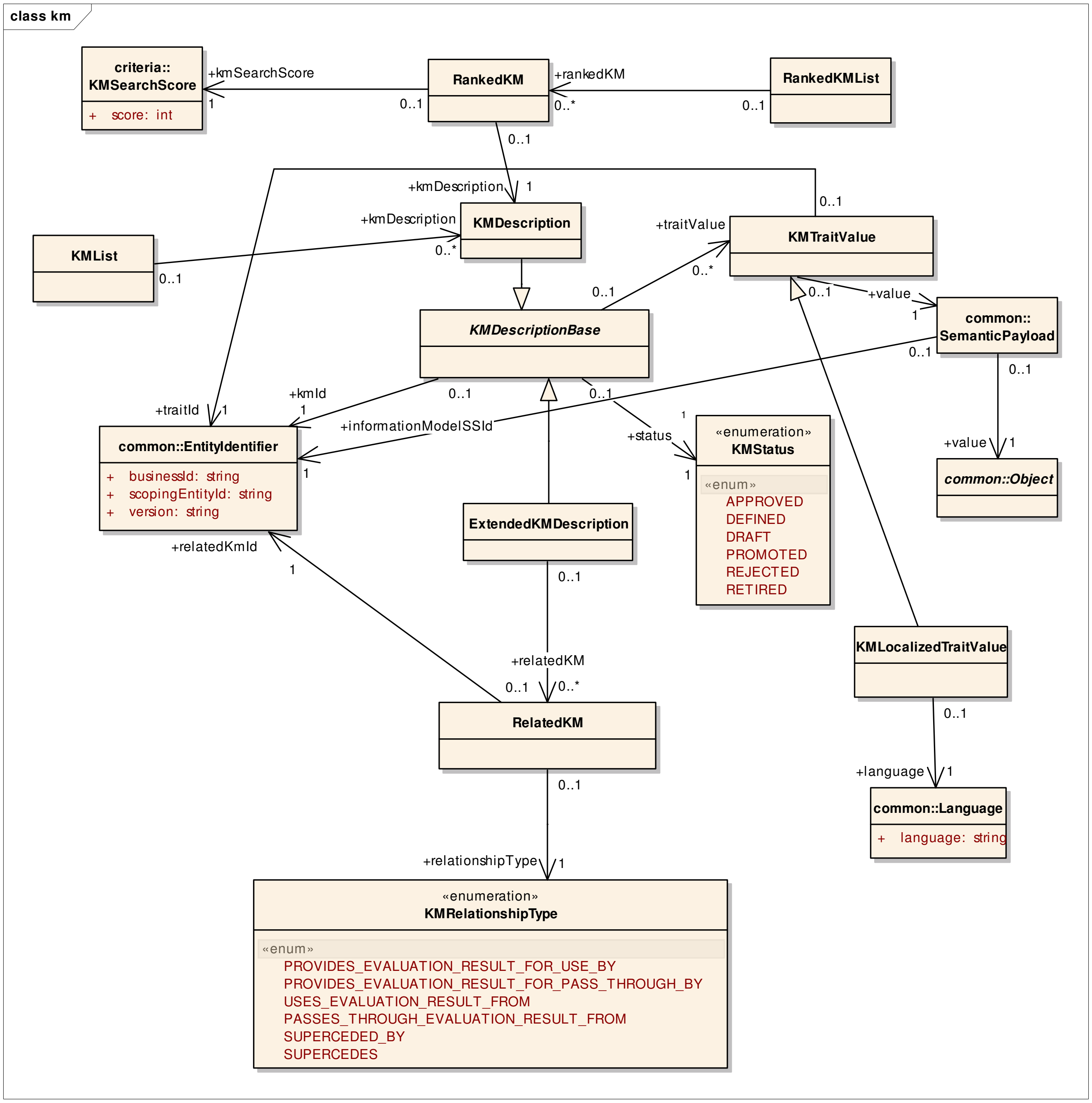


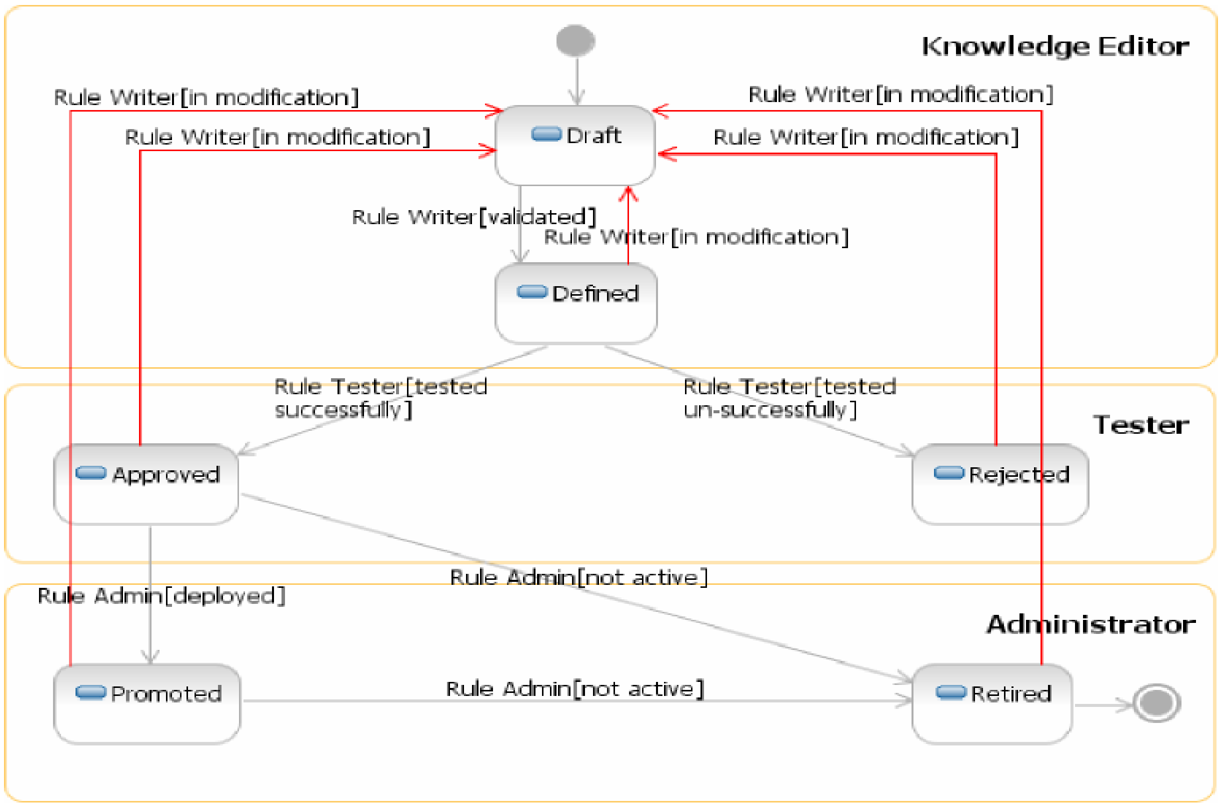
Figure 6.8. Model for Knowledge Module Description.

6.3.1.1 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.


Knowledge Editor
Rule Writer[in modification]
Rule Writer[in modification]
Draft
Rule Writer[in modification]
Rule Writer[in modification]
Rule Writer[validated]
Rule Writer[in modification]
Defined
Rule Tester[tested successfully]
Rule Tester[tested un-successfully]
Tester
Approved
Rejected
Rule Admin[not active]
Rule Admin[deployed]
Administrator
Promoted
Rule Admin[not active]
Retired

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 which require different levels of licenses to access.

6.3.1.2 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					
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				

	Change May Be Reflected In Version Part*:				
KM Change	**Maj**	**Min**	**Rev**	**None**	**Comments**
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 Section 6.9, 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).

6.3.1.3 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.

6.3.1.4 Knowledge Module Traits

A KM may possess specified traits. These traits are described in detail in Section 6.3.3, Trait.

6.3.2 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 (Section 6.2.1.2, Profile Types). The Semantic Requirement class is defined in the metadata.semanticrequirement package. Note that this requirement used to be referred to as KM Requirements in the HL7 DSS SFM DSTU.

6.3.2.1 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 Section 6.3.3, 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.
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) allowedEvaluationResultRequirement - specifies the superset of evaluation result models that can be used and associated CDS output specification models that can be used. (v) requiredEvaluationResultRequirement - specifies the evaluation result models and associated CDS output specification models, if any, 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.

6.3.2.2 Language Specification

Language shall be specifiied 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 Section 5 for normative references to these standards.

6.3.2.3 Semantic Requirement Model

The Semantic Requirement class model is shown below.

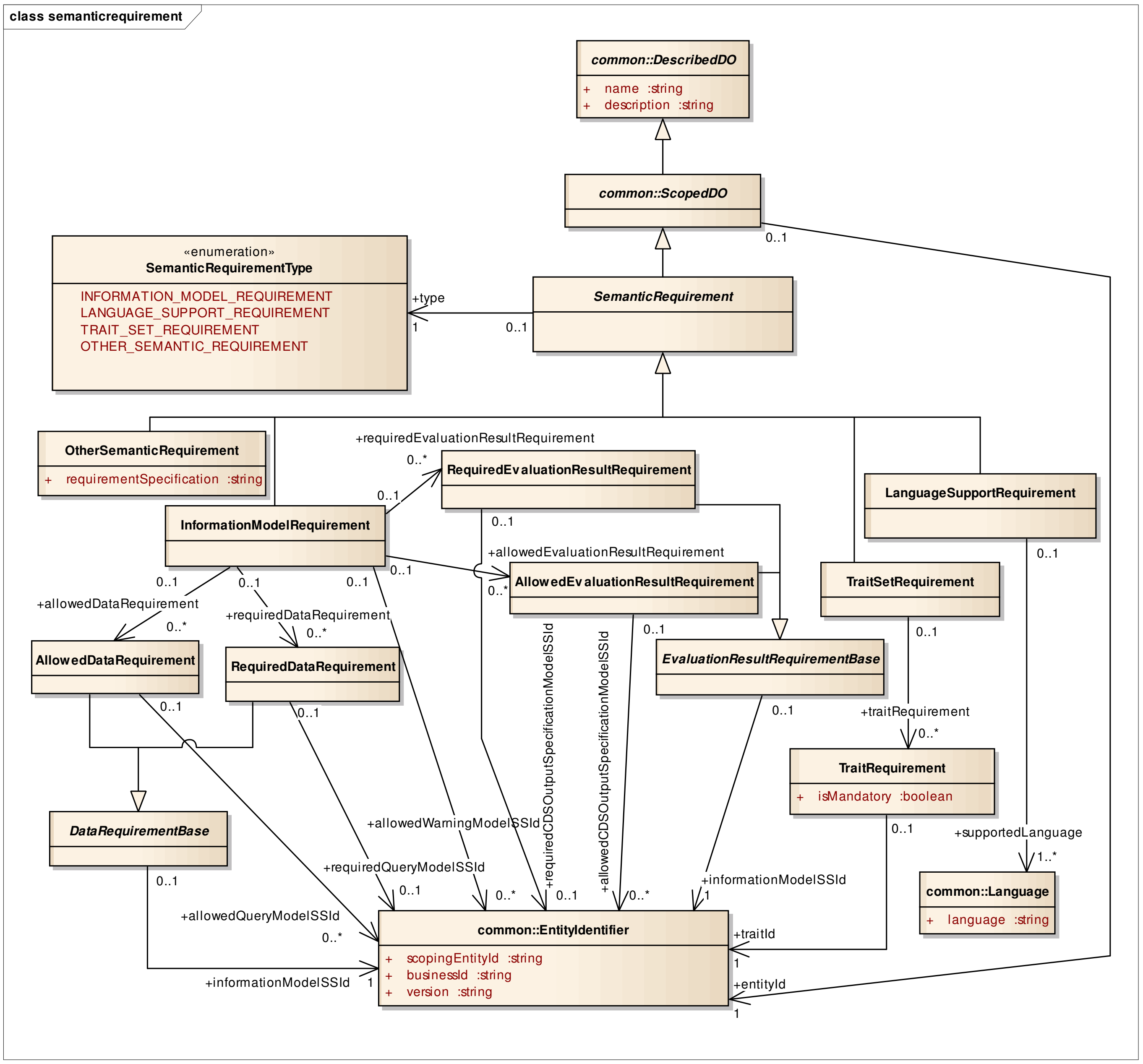


Figure 6.9. Model for Semantic Requirement.

6.3.2.4 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 allowedEvaluationResultRequirement.
- 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.

6.3.3 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 trait's 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 Section 6.3.2.2, Language Specification regarding DSS support for languages, as well as Section 6.8, Query Interface and Section 6.9, 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.

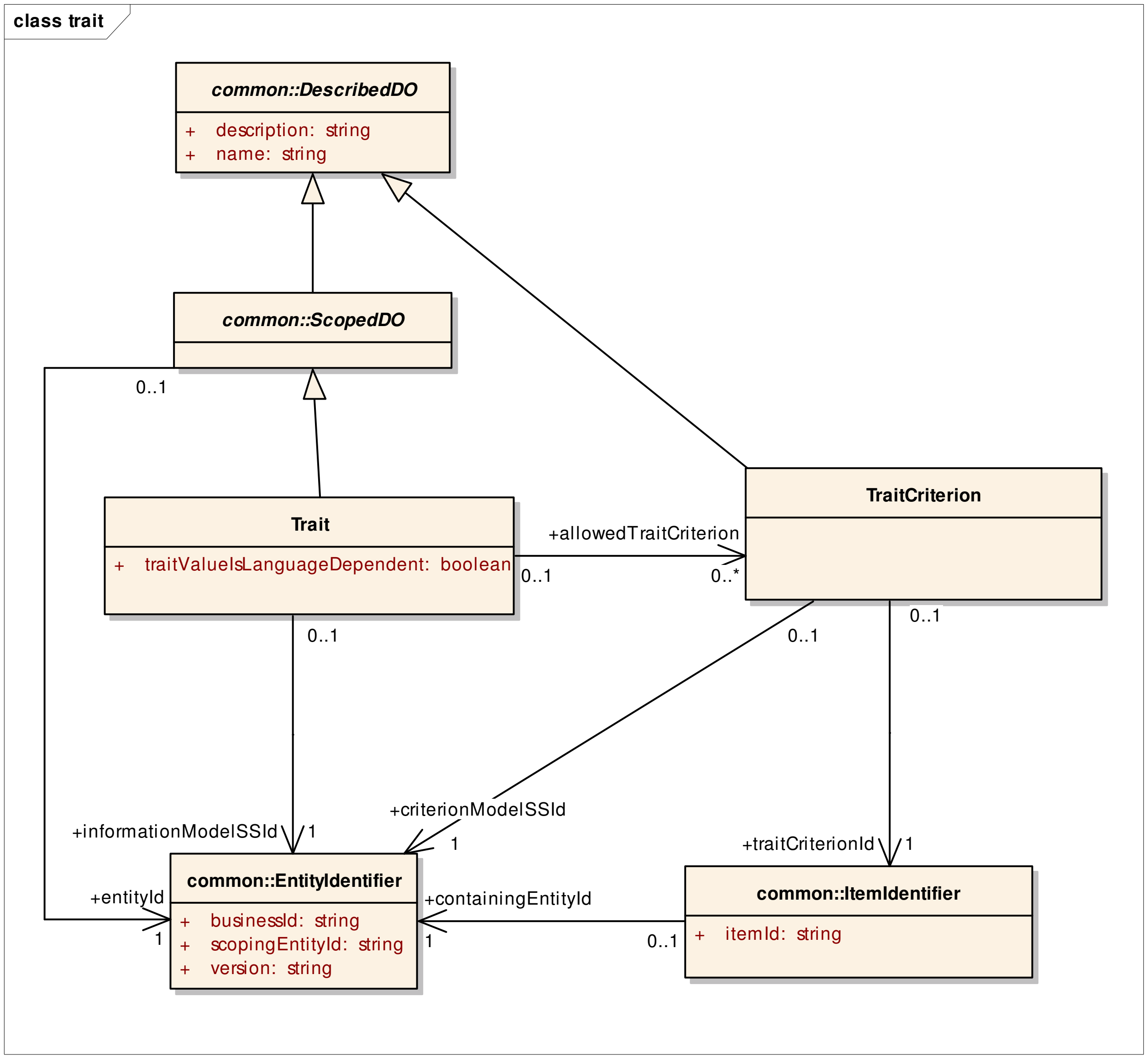


Figure 6.10. Model for Trait.

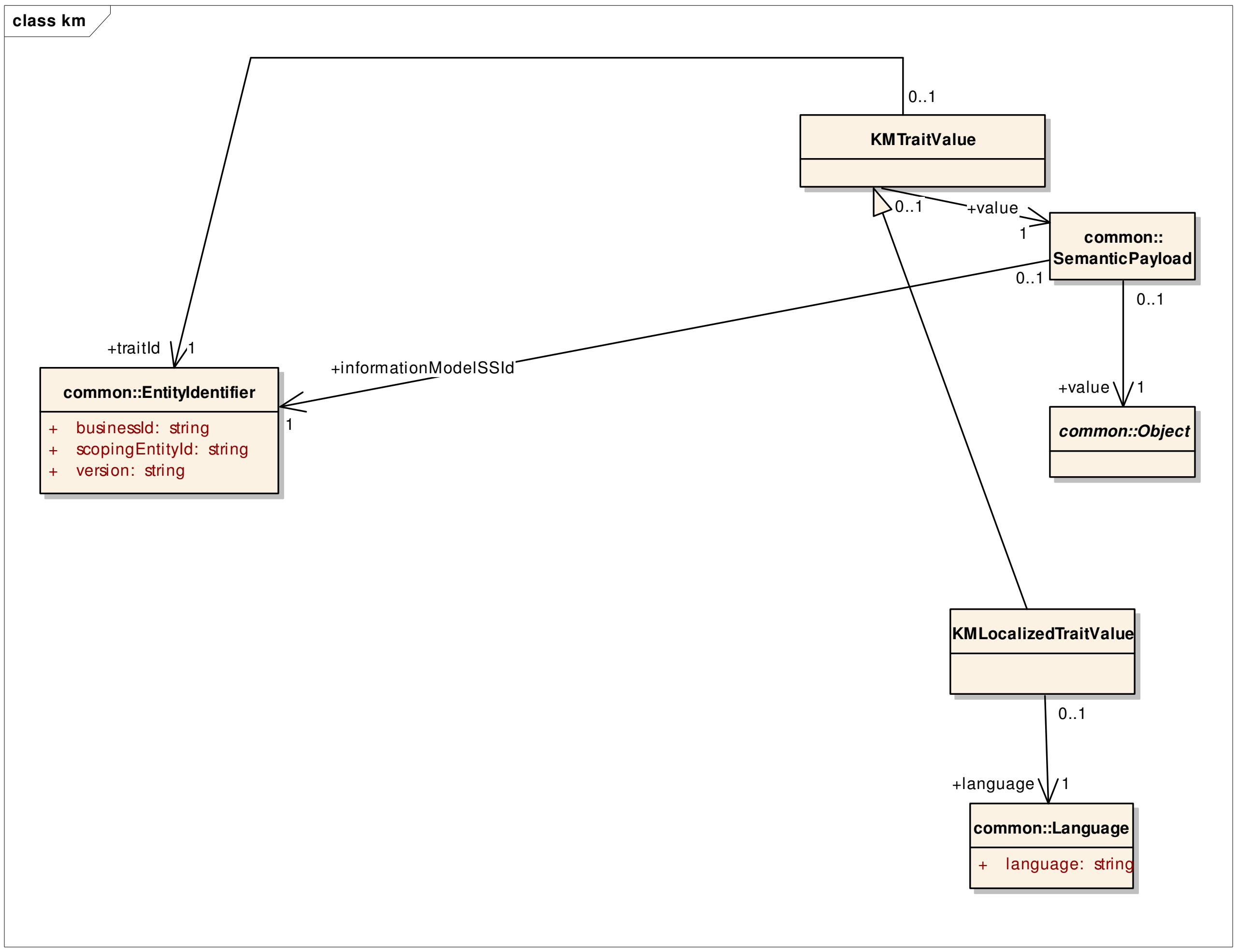


Figure 6.11. Model for Trait Value.

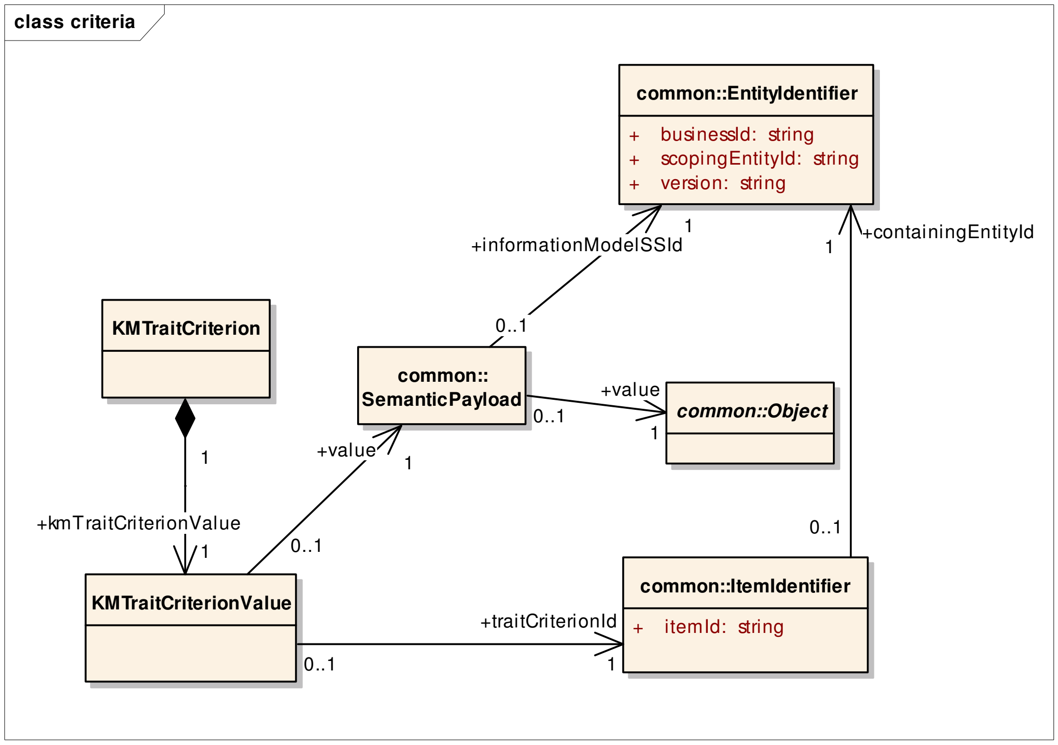


Figure 6.12. Model for Trait Criterion.

6.3.4 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.

6.3.4.1 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 section.

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.

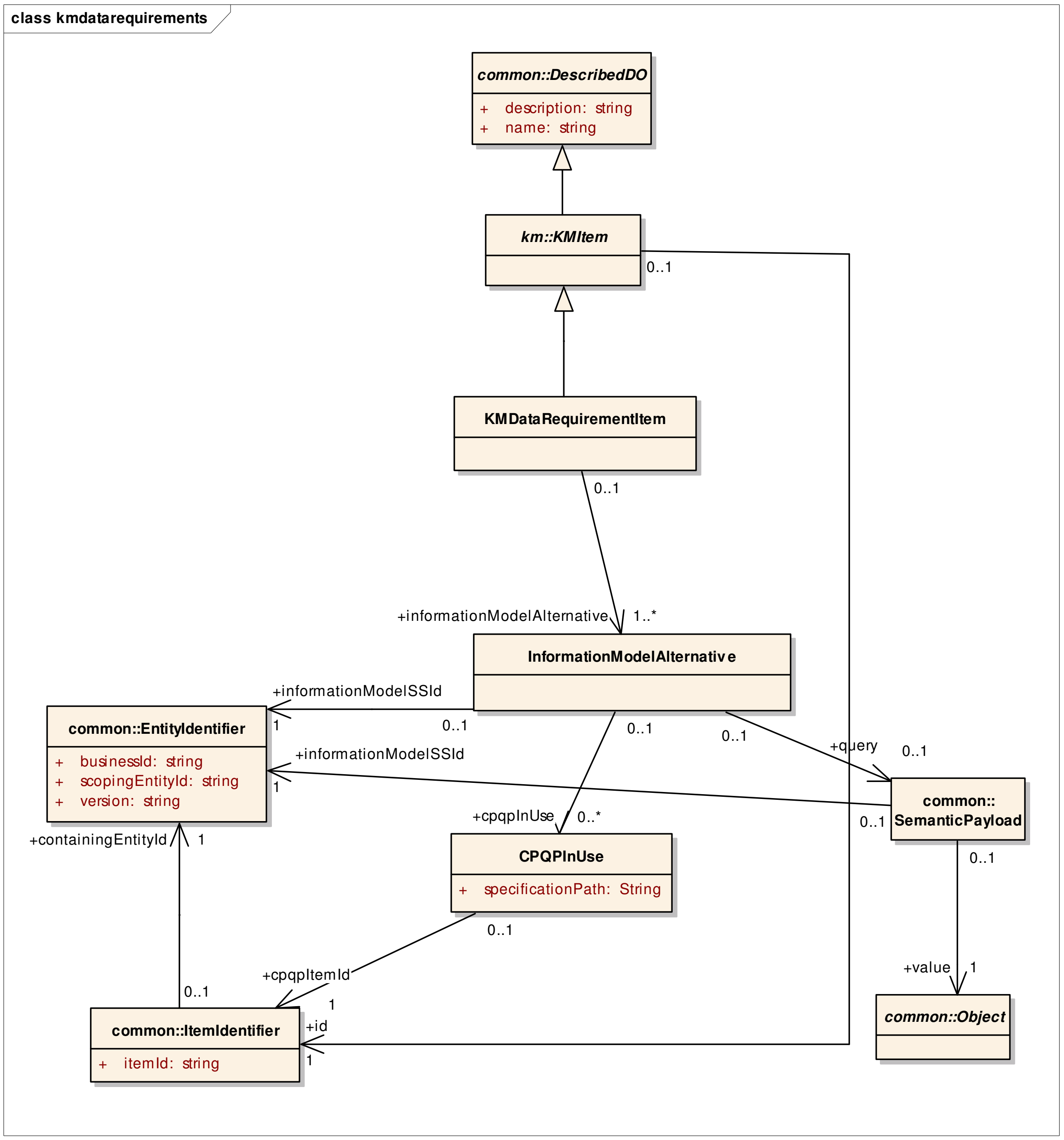


Figure 6.13. Model for KM Data Requirement Item.

6.3.4.2 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.

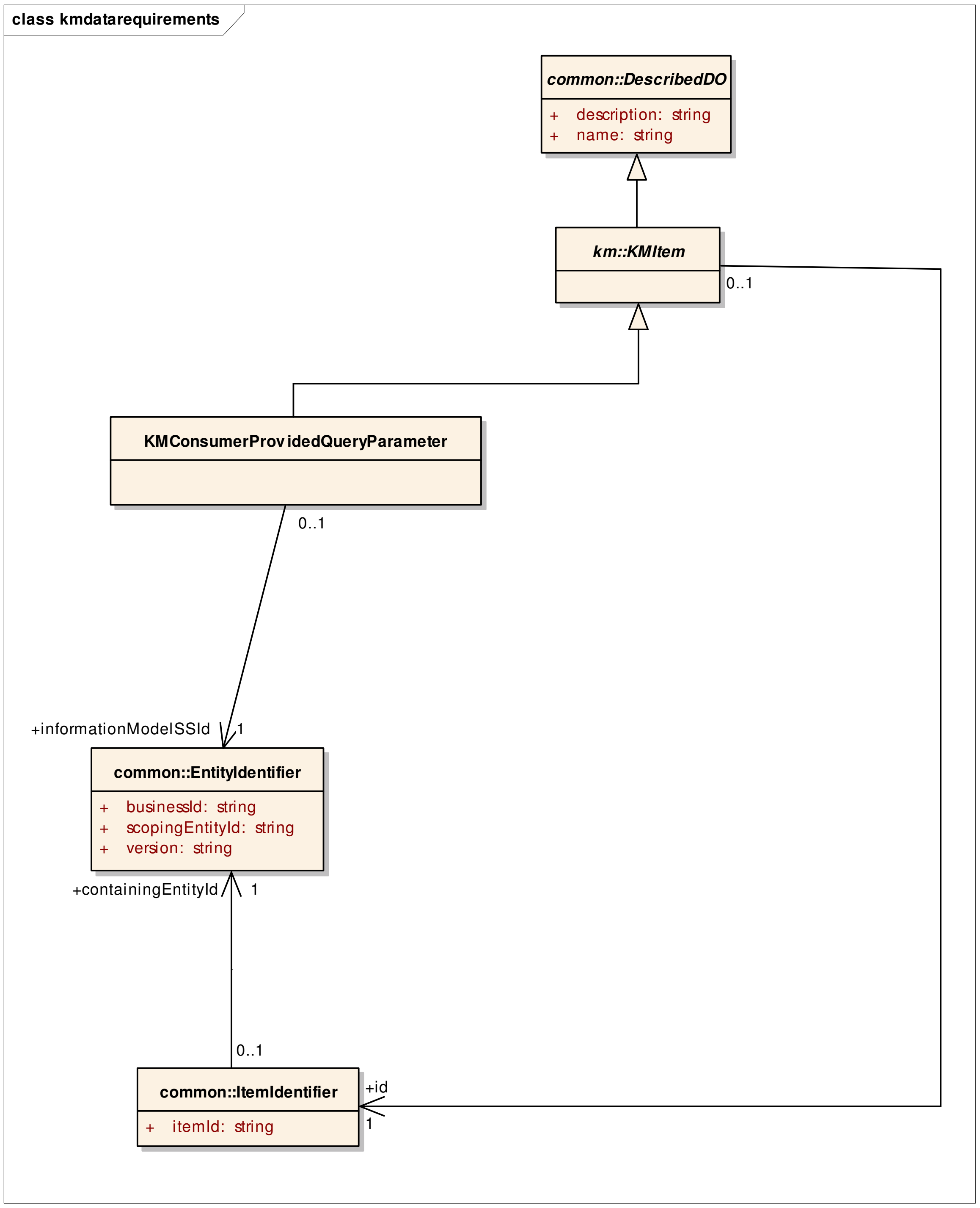


Figure 6.14. Model for Consumer-Provided Query Parameter.

6.3.4.3 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.

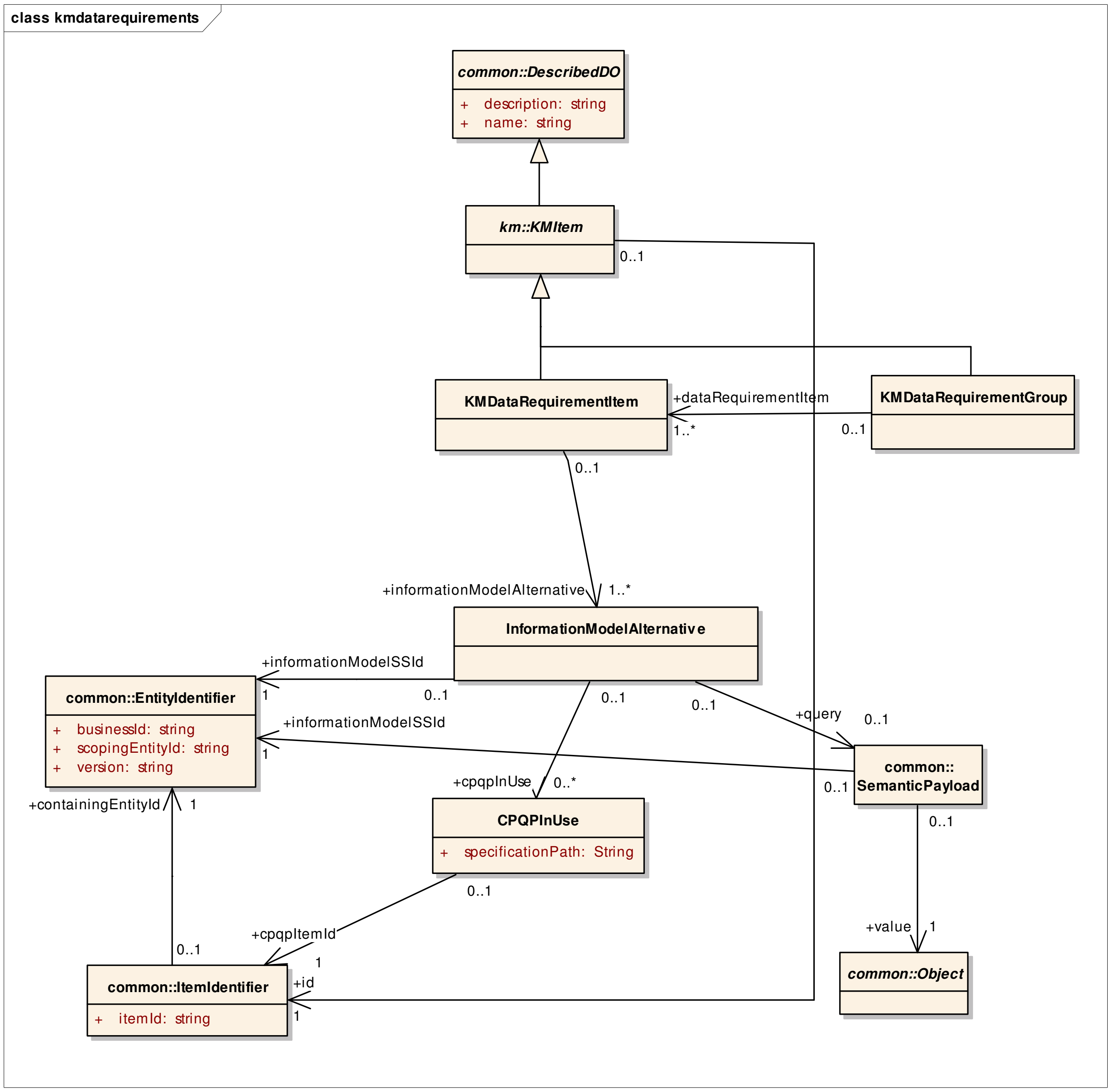


Figure 6.15. Model for KM Data Requirement Group.

6.3.4.4 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.

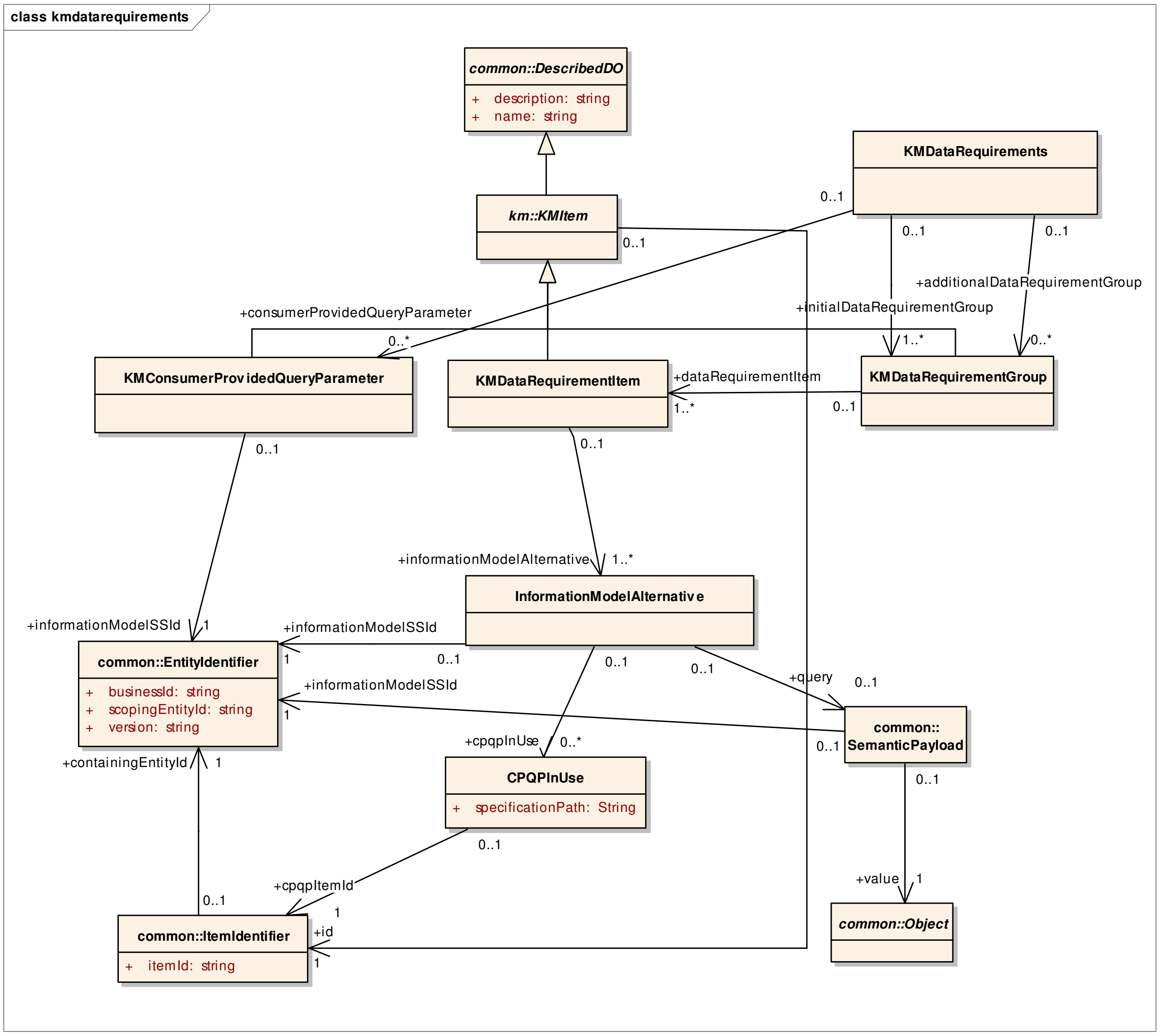


Figure 6.16. Model for KM Data Requirements.

6.3.5 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. The optional cdsOutputSpecification enables the specification of additional information on the evaluation results to be returned.

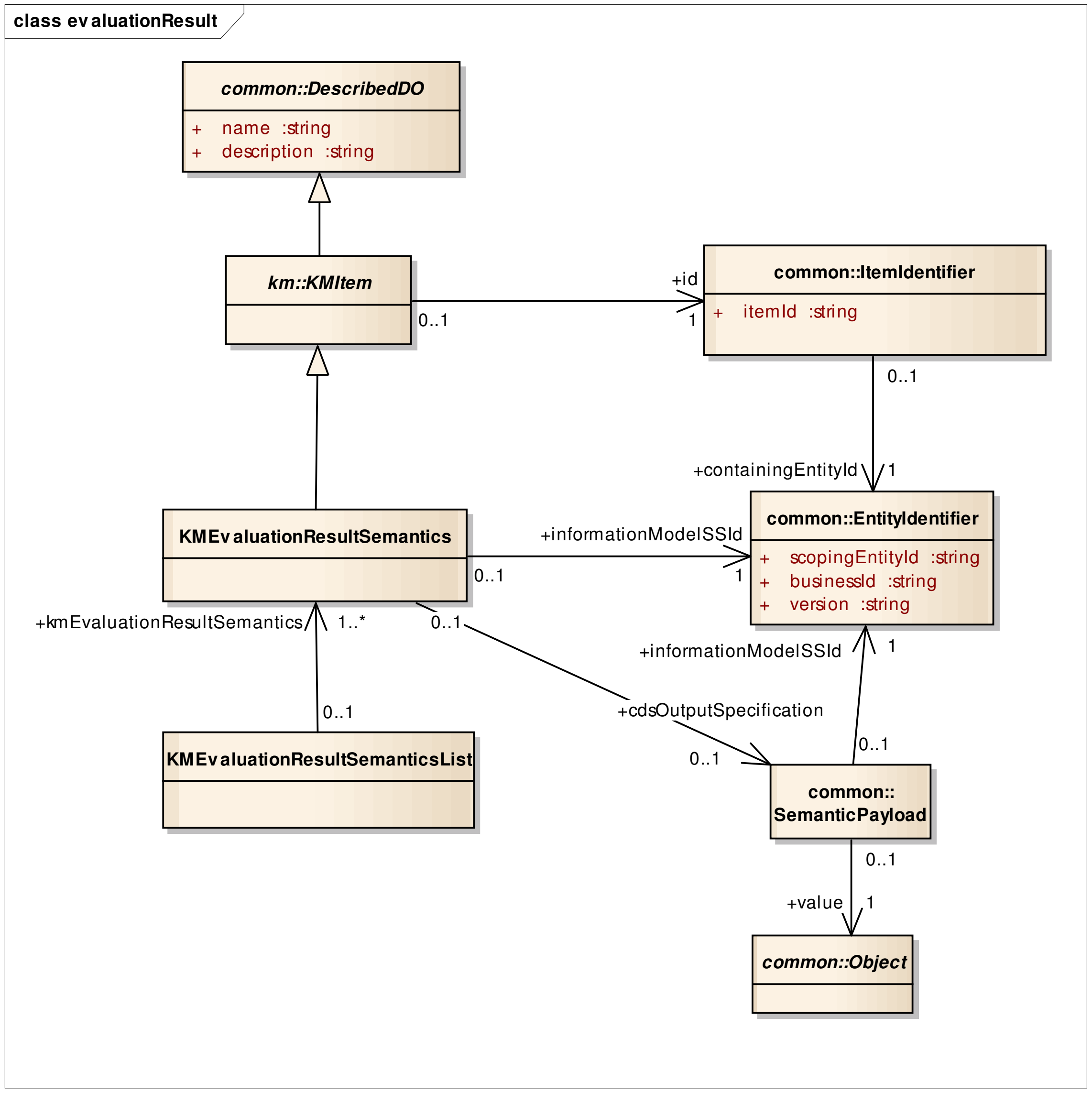


Figure 6.17. Model for KM Evaluation Result Semantics.

6.4 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 Section 6.7, Metadata Discovery Interface, Section 6.8, Query Interface, and Section 6.9, Evaluation Interface. The three models are provided here for reference.

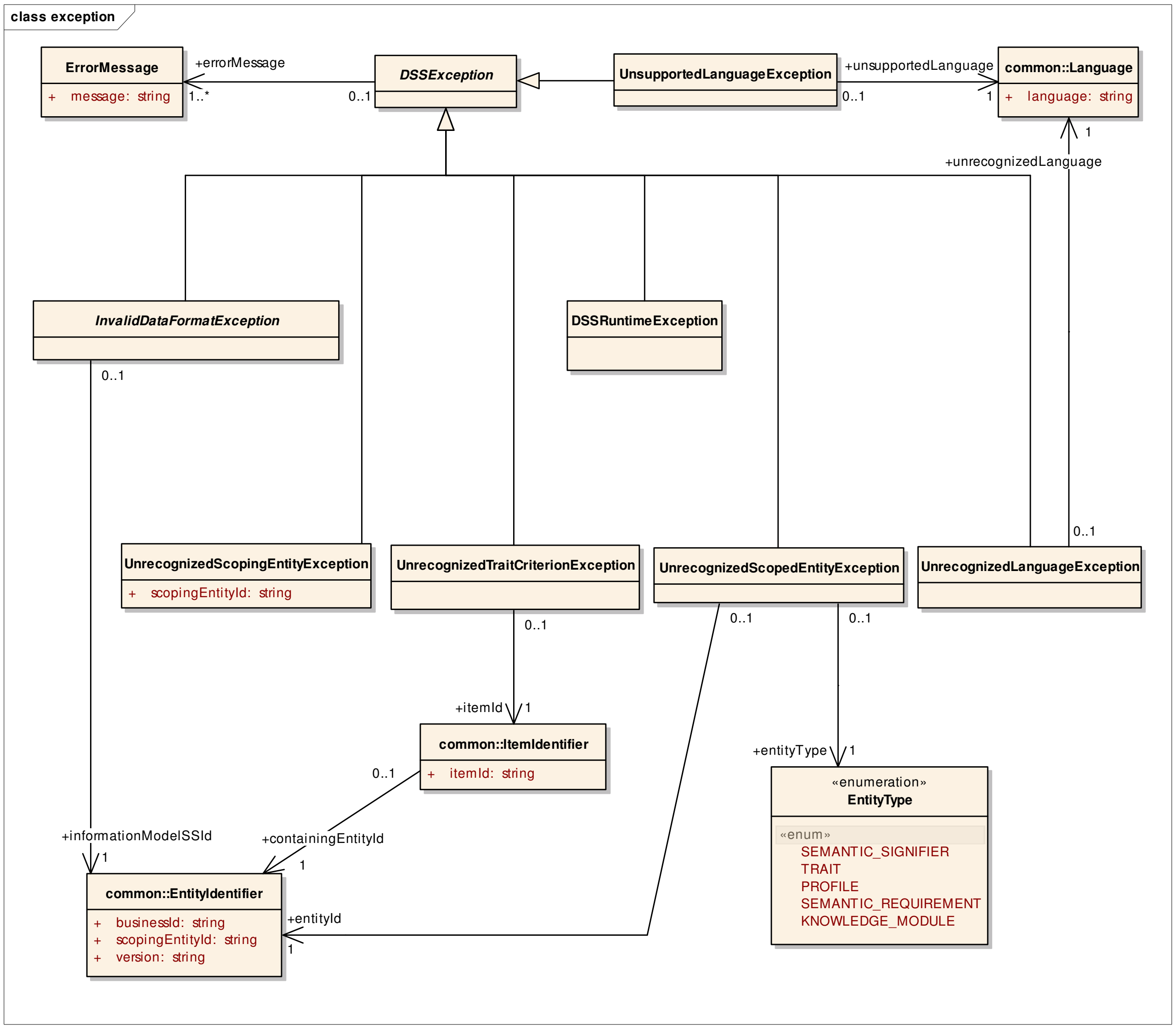


Figure 6.18. Model for Exceptions in common.exception package.

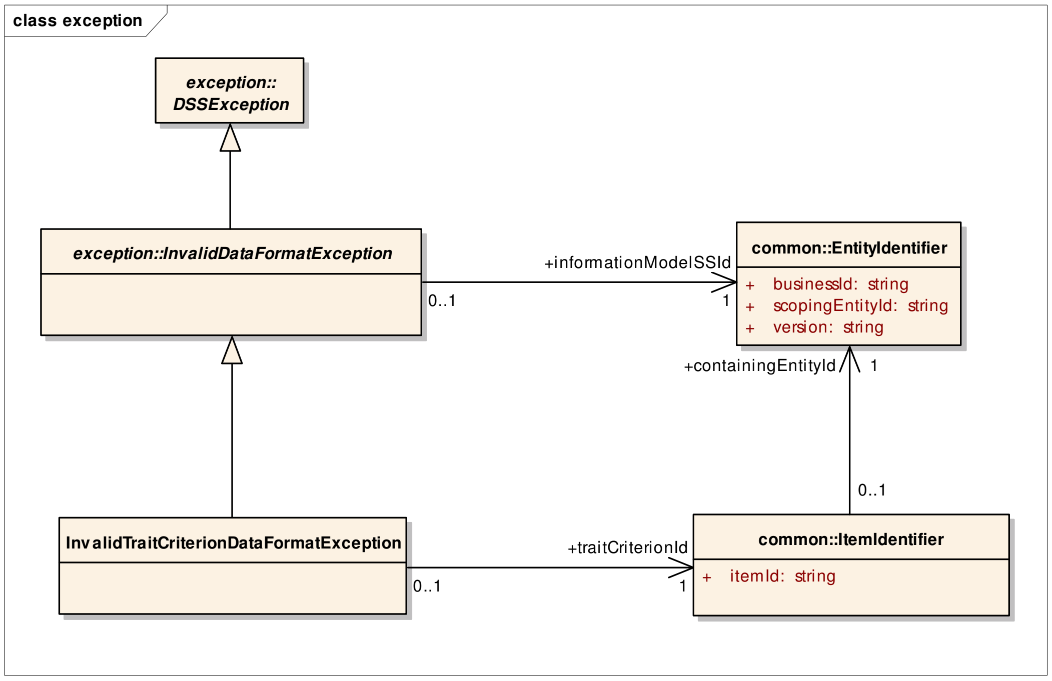


Figure 6.19. Model for Exceptions in query.exception package.

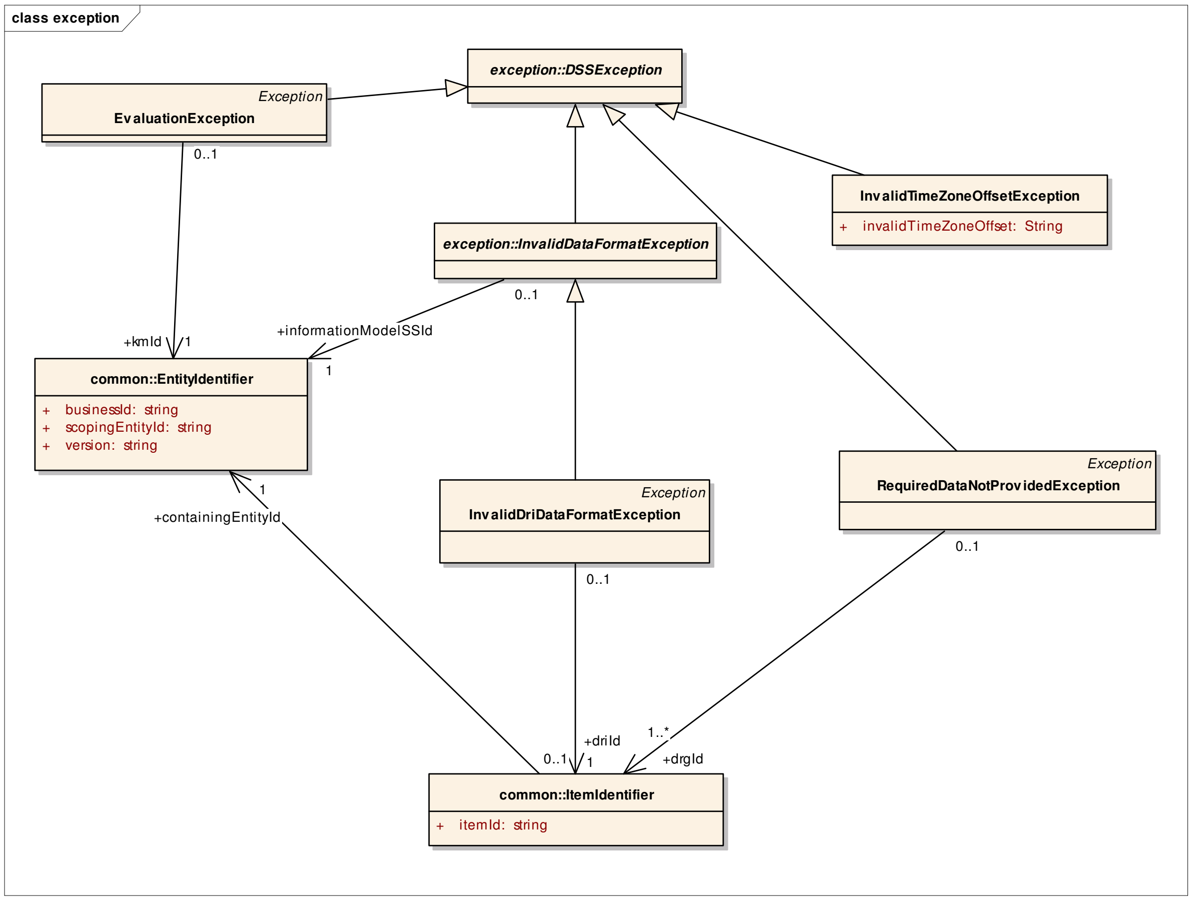


Figure 6.20. Model for Exceptions in evaluation.exception package.

6.5 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.

6.5.1 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 Section 6.3.1.1, 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

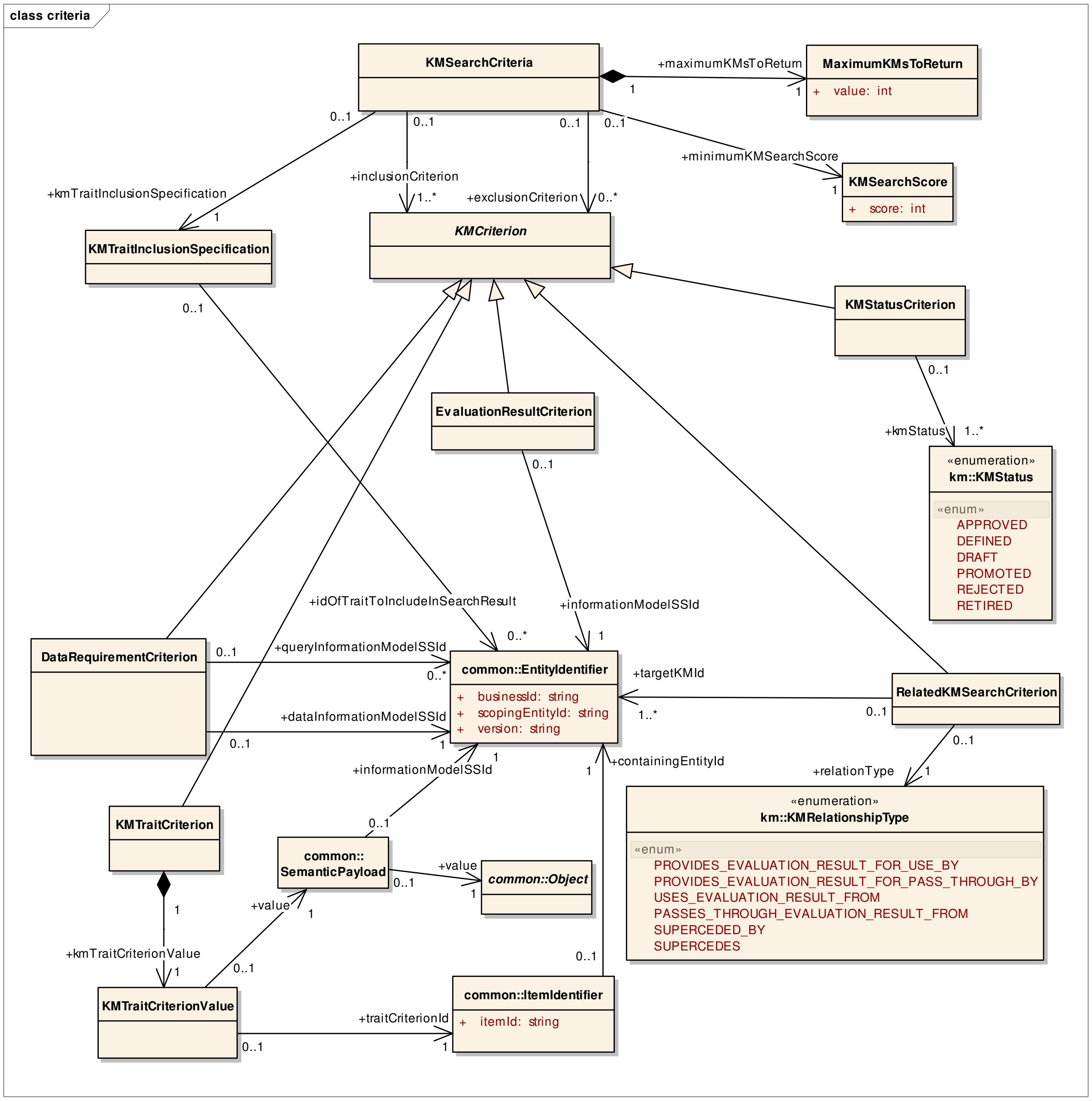


Figure 6.21. Model for KM Search Criteria.

6.6 *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.

6.6.1 Evaluation Request Model

The request model for DSS evaluations is shown below.

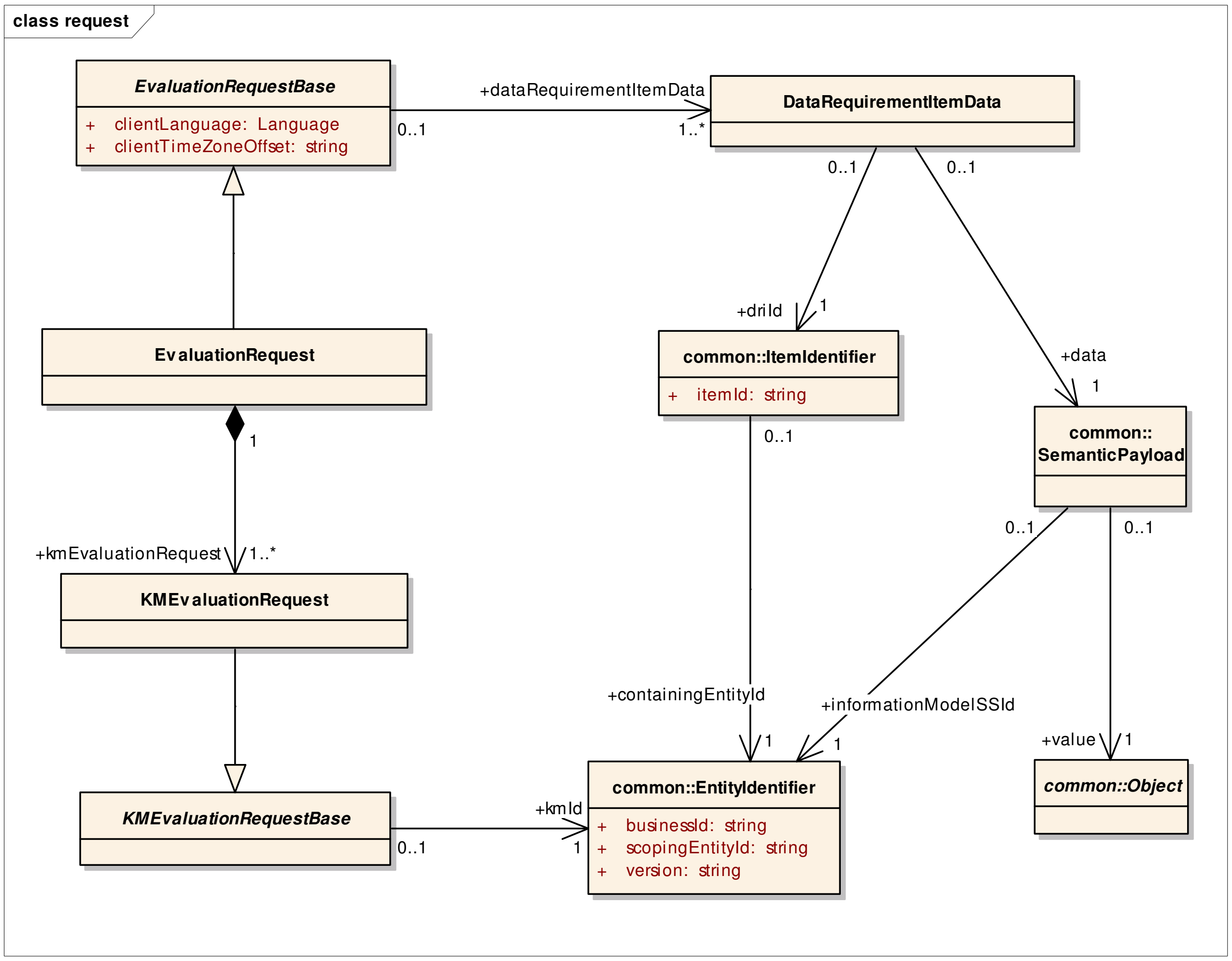


Figure 6.22. Model for Evaluation Request.

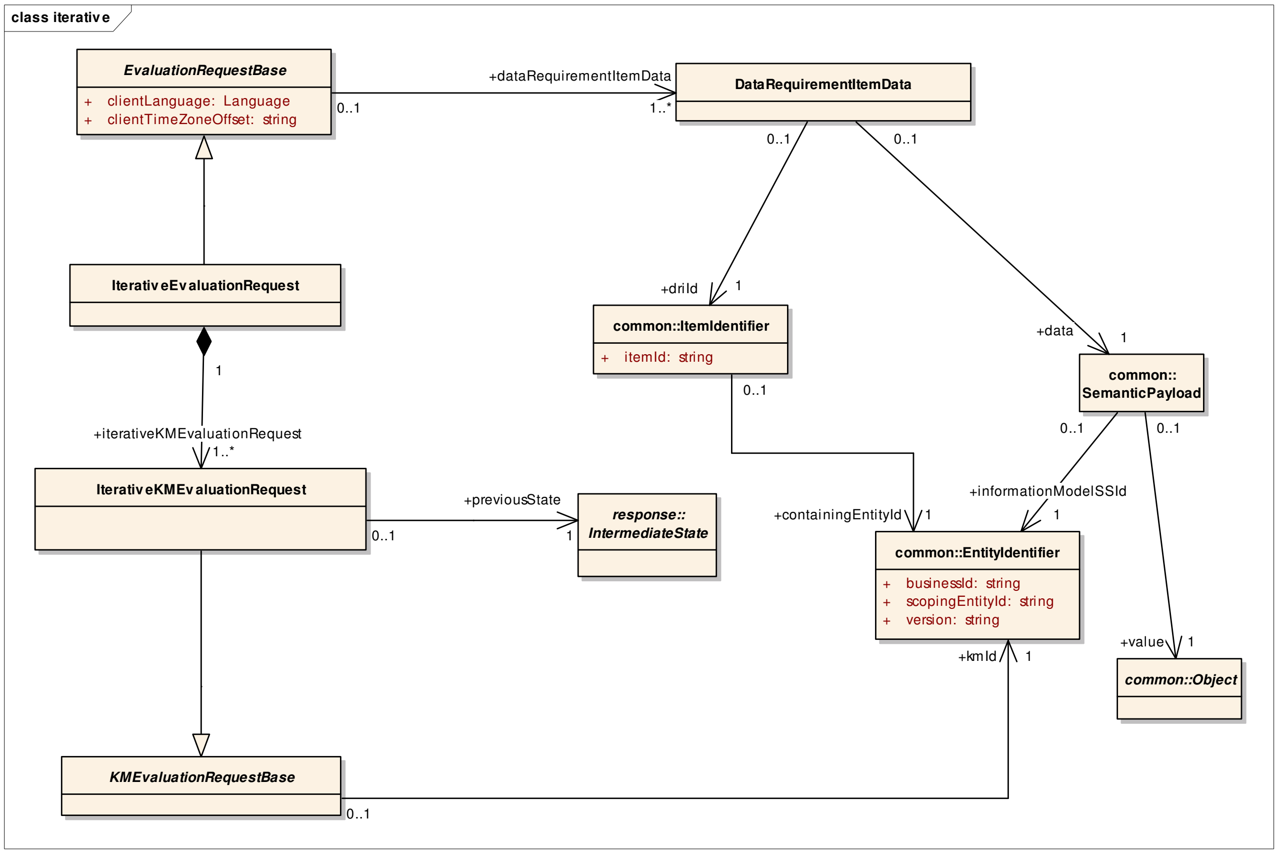


Figure 6.23. Model for Iterative Evaluation Request.

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).

6.6.1.1 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.

6.6.1.2 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.

6.6.2 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.

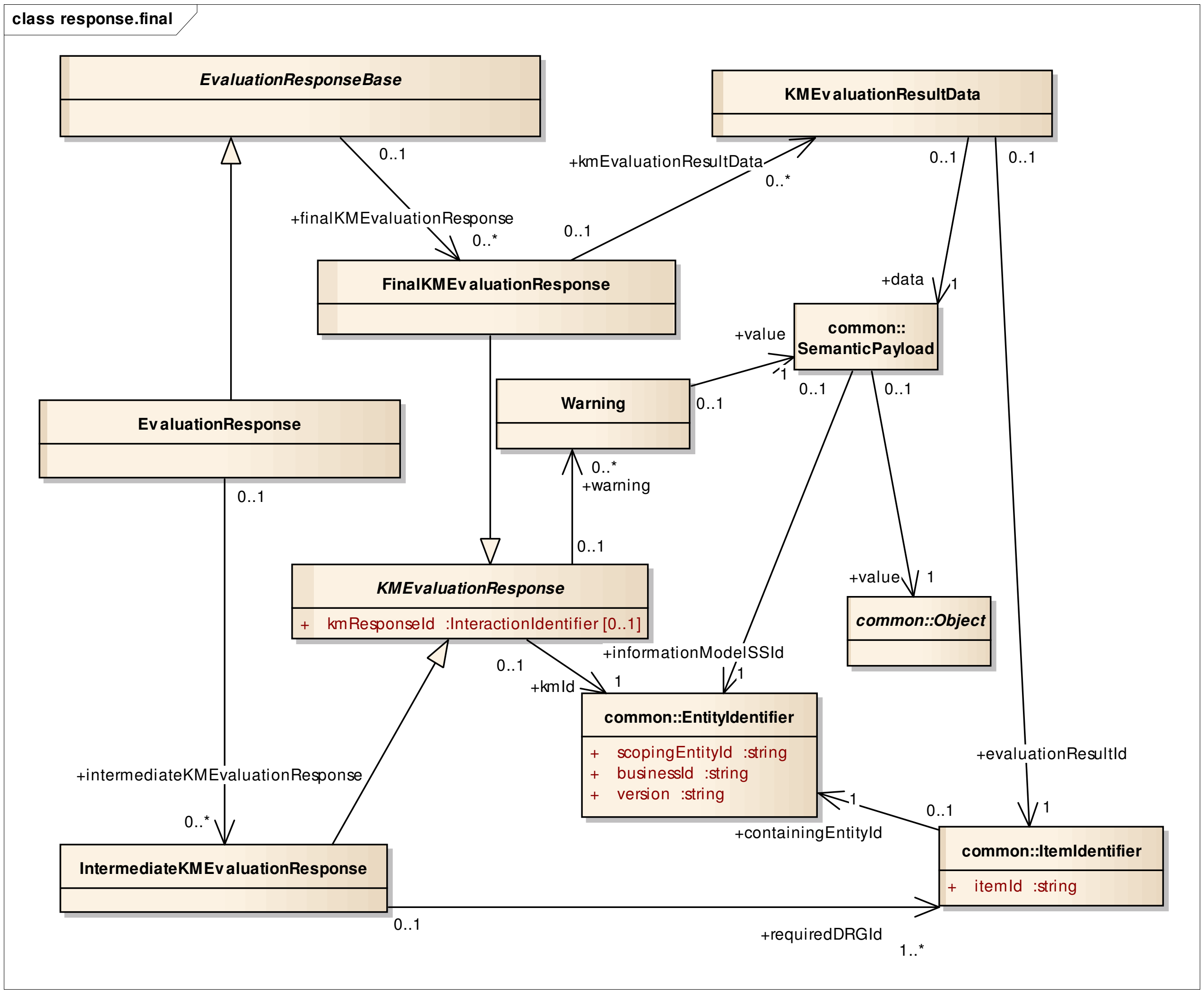


Figure 6.24. Model for Evaluation Response.

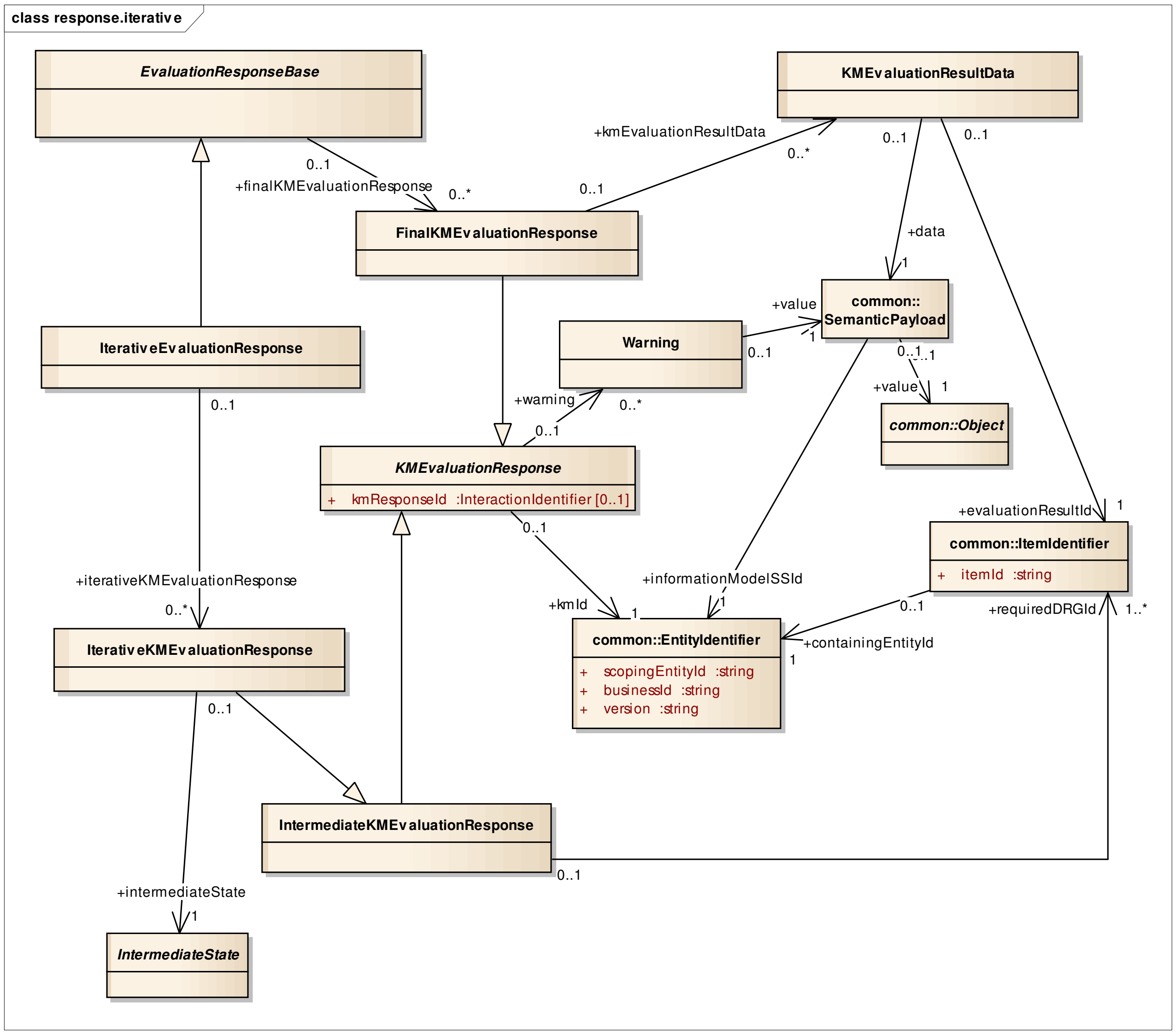
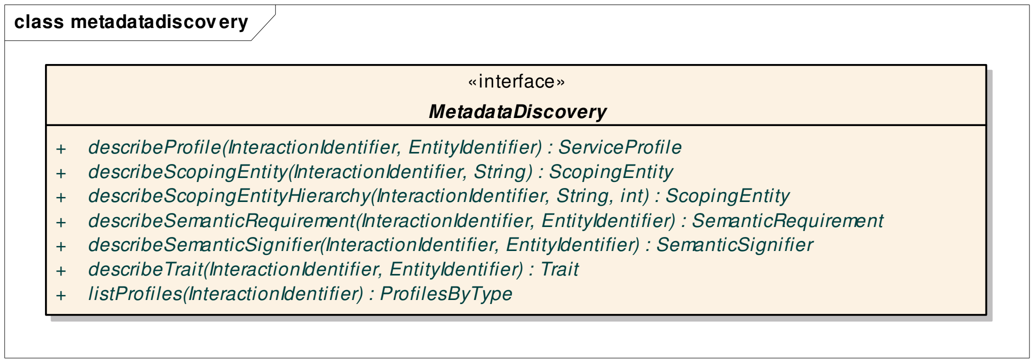


Figure 6.25. Model for Iterative Evaluation Response.

6.7 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.


class metadatadiscovery
«interface»
MetadataDiscovery
+ describeProfile(InteractionIdentifier, EntityIdentifier) : ServiceProfile
+ describeScopingEntity(InteractionIdentifier, String) : ScopingEntity
+ describeScopingEntityHierarchy(InteractionIdentifier, String, int) : ScopingEntity
+ describeSemanticRequirement(InteractionIdentifier, EntityIdentifier) : SemanticRequirement
+ describeSemanticSignifier(InteractionIdentifier, EntityIdentifier) : SemanticSignifier
+ describeTrait(InteractionIdentifier, EntityIdentifier) : Trait
+ listProfiles(InteractionIdentifier) : ProfilesByType

While not specified individually in the definition of DSS operations that follow in Section 6.7, Metadata Discovery Interface, Section 6.8, Query Interface, and Section 6.9, Evaluation Interface, **note that the following hold true for all operations across all interfaces:**

- All operations may throw a DSSRuntimeException if there is a runtime exception that is not otherwise specifically named.
- 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.
- All operations include the following in the response (not shown in the interface models): a requestId representing the InteractionIdentifier provided as the input, and a responseId representing the InteractionIdentifier created by the DSS to identify the response.

6.7.1 listProfiles

Description	Returns a list of all of the profiles supported by the service as a ProfilesByType object.
Inputs	None.
Outputs	ProfilesByType (Section 6.2.1.3, 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.

6.7.2 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 (Section 6.1.3, Entity Identifier).
Outputs	ServiceProfile (Section 6.2.1.3, Service Profile Model).
Exception conditions	The profile is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.7.3 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) (Section 6.1.2, Scoping Entity).
Outputs	ScopingEntity (Section 6.1.2, 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	

6.7.4 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) (Section 6.1.2, 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 (Section 6.1.2, 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	

6.7.5 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 (Section 6.1.3, Entity Identifier).
Outputs	SemanticRequirement (Section 6.3.2.3, Semantic Requirement Model).
Exception conditions	The semantic requirement is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.7.6 describeSemanticSignifier

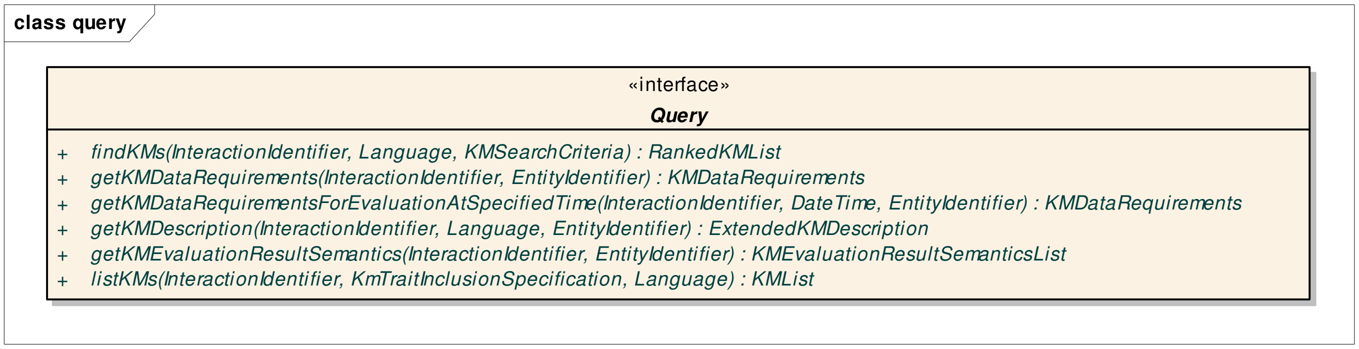
Description	Throws UnrecognizedScopedEntityException if the specified semantic signifier EntityIdentifier is not recognized by the service. If specified EntityIdentifier is recognized by the service, returns a description of the semantic signifier as a SemanticSignifier object.
Inputs	EntityIdentifier of semantic signifier (Section 6.1.3, Entity Identifier).
Outputs	SemanticSignifier (Section 6.2.2.3, Semantic Signifier Model).
Exception conditions	The semantic signifier is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.7.7 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 (Section 6.1.3, Entity Identifier).
Outputs	Trait (Section 6.3.1.4, Knowledge Module Traits).
Exception conditions	The knowledge module trait is not recognized by the service (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.8 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.



```
+ findKMs(InteractionIdentifier, Language, KMSearchCriteria) : RankedKMList
+ getKMDataRequirements(InteractionIdentifier, EntityIdentifier) : KMDataRequirements
+ getKMDataRequirementsForEvaluationAtSpecifiedTime(InteractionIdentifier, DateTime, EntityIdentifier) : KMDataRequirements
+ getKMDescription(InteractionIdentifier, Language, EntityIdentifier) : ExtendedKMDescription
+ getKMEvaluationResultSemantics(InteractionIdentifier, EntityIdentifier) : KMEvaluationResultSemanticsList
+ listKMs(InteractionIdentifier, KmTraitInclusionSpecification, Language) : KMList
```

6.8.1 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 (Section 6.3.2.2, Language specification). • KMTraitInclusionSpecification: specification of which KM traits to include in the KM descriptions returned (Section 6.5.1, Search Criteria).
Outputs	List of KMs (KMList; Section 6.3.1, 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.

6.8.2 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 (Section 6.3.2.2, Language specification). • KMSearchCriteria (Section 6.5.1, Search Criteria).
Outputs	List of knowledge modules fulfilling the search criteria (RankedKMList; Section 6.3.1, 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	

6.8.3 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 (Section 6.1.3, Entity Identifier). • Client's Language (Section 6.3.2.2, Language specification).
Outputs	ExtendedKMDescription (Section 6.3.1, 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	

6.8.4 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 (Section 6.1.3, Entity Identifier).
Outputs	KMEvaluationResultSemanticsList (Section 6.3.5, KM Evaluation Result Semantics).
Exception conditions	The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.8.5 getKMDataRequirements

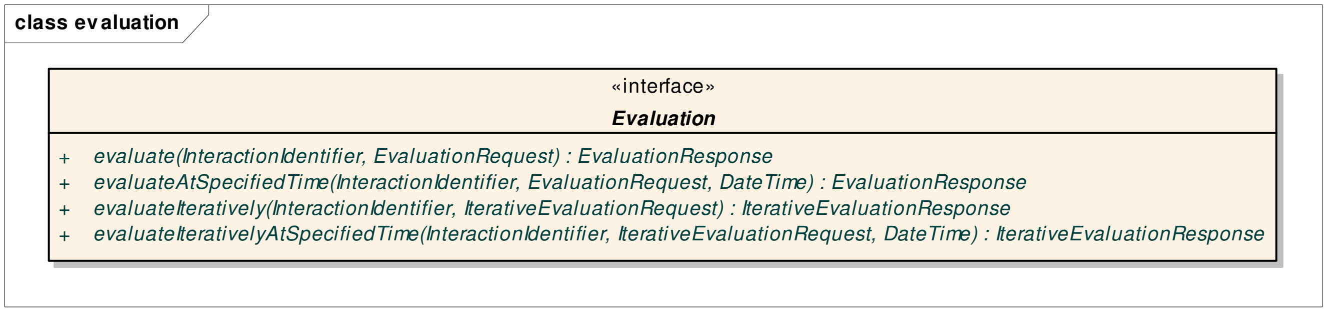
Description	Throws UnrecognizedScopedEntityException if the specified knowledge module EntityIdentifier is not recognized by the service. If specified EntityIdentifier is recognized by the service, returns a specification of the data required by the knowledge module for conducting an evaluation as a KMDataRequirements object.
Inputs	EntityIdentifier of knowledge module (Section 6.1.3, Entity Identifier).
Outputs	KMDataRequirements (Section 6.3.4.4, KM Data Requirements).
Exception conditions	The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.8.6 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 (Section 6.1.3, Entity Identifier).
Outputs	KMDataRequirements (Section 6.3.4.4, KM Data Requirements).
Exception conditions	The requested knowledge module does not exist (UnrecognizedScopedEntityException).
Aspects left to implementers	

6.9 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.


class evaluation
«interface»
Evaluation
+ evaluate(InteractionIdentifier, EvaluationRequest) : EvaluationResponse
+ evaluateAtSpecifiedTime(InteractionIdentifier, EvaluationRequest, DateTime) : EvaluationResponse
+ evaluateIteratively(InteractionIdentifier, IterativeEvaluationRequest) : IterativeEvaluationResponse
+ evaluateIterativelyAtSpecifiedTime(InteractionIdentifier, IterativeEvaluationRequest, DateTime) : IterativeEvaluationResponse

6.9.1 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 (Section 6.6.1, Evaluation Request Model)
Outputs	EvaluationResponse object (Section 6.6.2, 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	

6.9.2 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 (Section 6.6.1, Evaluation Request Model). • DateTime of the evaluation.
Outputs	EvaluationResponse object (Section 6.6.2, 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	

6.9.3 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 (Section 6.6.1, Evaluation Request Model).
Outputs	IterativeEvaluationResponse object (Section 6.6.2, 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	

6.9.4 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 (Section 6.6.1, Evaluation Request Model). • Date and time of the evaluation.
Outputs	IterativeEvaluationResponse object (Section 6.6.2, 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 *Profiles and Semantic Requirements Specified as a Part of this Specification*

6.10.1 Overview

This specification specifies several profiles and semantic requirements to ensure a minimum level of interoperability among DSSs.

This section defines these normative specifications, which consist of two functional profiles (the HSSP Simple Evaluation DSS Functional Profile and the HSSP Complete DSS Functional Profile), one semantic profile (the HSSP Minimum DSS Semantic Profile), one semantic requirement (the HSSP Minimum DSS Trait Set Requirement), and two conformance profiles (the HSSP Simple Evaluation Conformance Profile and the HSSP Complete 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 Simple Evaluation DSS Functional Profile	Supported by HSSP Complete DSS Functional Profile
describeProfile		X
describeScopingEntity		X
describeScopingEntityHierarchy		X
describeSemanticRequirement		X
describeSemanticSignifier		X
describeTrait		X
listProfiles		X
findKMs		X
getKMDataRequirements		X
getKMDataRequirementsForEvaluationAtSpecifiedTime		X
getKMDescription		X
getKMEvaluationResultSemantics		X
listKMs		X
evaluate	X	X
evaluateAtSpecifiedTime	X	X
evaluateIteratively		X
evaluateIterativelyAtSpecifiedTime		X

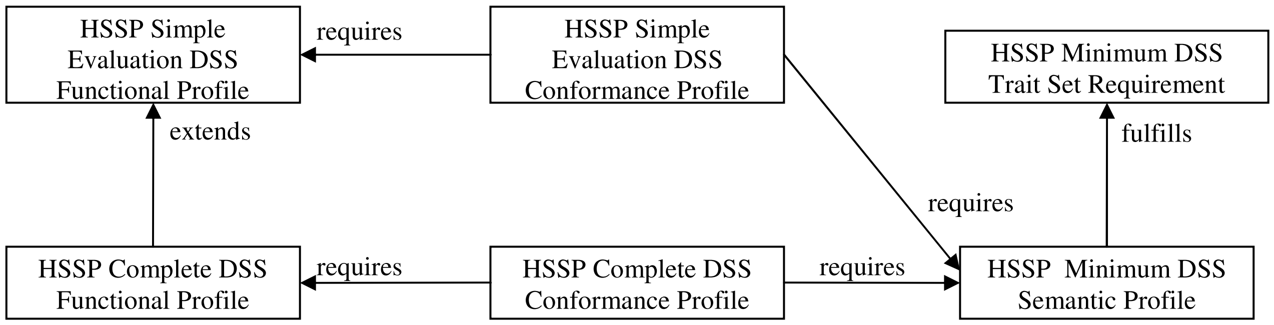

HSSP Simple Evaluation DSS Functional Profile
requires
HSSP Simple Evaluation DSS Conformance Profile
HSSP Minimum DSS Trait Set Requirement
extends
fulfills
requires
HSSP Complete DSS Functional Profile
requires
HSSP Complete DSS Conformance Profile
requires
HSSP Minimum DSS Semantic Profile

Figure 6.26. Relationships between profiles.

6.10.2 HSSP Simple Evaluation DSS Functional Profile, Version 2.0

To claim conformance to the HSSP Simple Evaluation DSS Functional Profile, version 2.0, a DSS must implement and support the following service operations defined in this specification:

From the Evaluation interface:

evaluate

evaluateAtSpecifiedTime (this has been added for Version 2.0)

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

scopingEntityId: org.hssp.dss

businessId: HSSP_Simple_Evaluation_DSS_Functional_Profile

version: 2.0

type: FUNCTIONAL_PROFILE

6.10.3 HSSP Complete DSS Functional Profile, Version 2.0

To claim conformance to the HSSP Complete DSS Functional Profile, version 2.0, 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: 2.0

type: FUNCTIONAL_PROFILE

6.10.4 HSSP Minimum DSS Semantic Profile, Version 2.0

To claim conformance to the HSSP Minimum DSS Semantic Profile version 2.0, the service must fulfill the HSSP Minimum DSS Trait Set Requirement, which is specified in Section 6.10.5 below. The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Minimum_Meta_Data_DSS_Semantic_Profile

version: 2.0

type: SEMANTIC_PROFILE

6.10.5 HSSP Minimum DSS Trait Set Requirement, Version 2.0

To claim conformance to this trait set requirement, all knowledge modules in the DSS must support the traits and trait criteria specified in Section 6.10.5.1 and Section 6.10.5.2 below. The relevant identifying parameters for this semantic requirement shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Minimum_DSS_Trait_Set_Requirement

version: 2.0

type: TRAIT_SET_REQUIREMENT

6.10.5.1 Knowledge Module Traits Required by Trait Set Requirement

Please note that all schemas referenced in this section are included as supplemental files with this specification.

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 Section 6.2.2.3, Semantic Signifier Model).

6.10.5.1.1 StewardOrganization

Description:

The organization acting as the steward of the KM

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: StewardOrganization

Version: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.StewardOrganization

Version: 2.0

Root global element name: StewardOrganization

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.10.5.1.2 CreationDate

Description:

Date KM was first created

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: CreationDate

Version: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.CreationDate

Version: 2.0

Root global element name: CreationDate

Trait attributes:

Is mandatory: true

Trait value is localized: false

6.10.5.1.3 LastReviewDate

Description:

Date when KM was last reviewed for accuracy

Trait identifier:

Scoping entity identifier: org.hssp.dss.traits

Business identifier: LastReviewDate

Version: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.LastReviewDate

Version: 2.0

Root global element name: LastReviewDate

Trait attributes:

Is mandatory: true

Trait value is localized: false

6.10.5.1.4 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: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.AuthorList

Version: 2.0

Root global element name: AuthorList

Trait attributes:

Is mandatory: true

Trait value is localized: false

6.10.5.1.5 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: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.FreeTextKeywordList

Version: 2.0

Root global element name: FreeTextKeywordList

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.10.5.1.6 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: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.CodedValueKeywordList

Version: 2.0

Root global element name: CodedValueKeywordList

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.10.5.1.7 ***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: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.Purpose

Version: 2.0

Root global element name: Purpose

Trait attributes:

Is mandatory: true

Trait value is localized: true

6.10.5.1.8 ***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: 2.0

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.Explanation

Version: 2.0

Root global element name: Explanation

Trait attributes:

Is mandatory: true

Trait value is localized: true

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

6.10.5.2.1 ReviewedOnOrAfter

Parent trait: LastReviewDate (Section 6.10.5.1.3, 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: 2.0

Item identifier: ReviewedOnOrAfter

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.ReviewedOnOrAfter

Version: 2.0

Root global element name: ReviewedOnOrAfter

6.10.5.2.2 ReviewedWithinLastXDays

Parent trait: LastReviewDate (Section 6.10.5.1.3, LastReviewDate)

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: 2.0

Item identifier: ReviewedWithinLastXDays

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.ReviewedWithinLastXDays

Version: 2.0

Root global element name: ReviewedWithinLastXDays

6.10.5.2.3 FreeTextKeywordContainsString

Parent trait: FreeTextKeywordList (Section 6.10.5.1.5, FreeTextKeywordList)

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: 2.0

Item identifier: FreeTextKeywordContainsString

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.FreeTextKeywordContainsString

Version: 2.0

Root global element name: FreeTextKeywordContainsString

6.10.5.2.4 CodedValueKeywordExists

Parent trait: CodedValueKeywordList (Section 6.10.5.1.6, CodedValueKeywordList)

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: 2.0

Item identifier: CodedValueKeywordExists

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.CodedValueKeywordExists

Version: 2.0

Root global element name: CodedValueKeywordExists

6.10.5.2.5 CodedValueKeywordOrKeywordDescendantExists

Parent trait: CodedValueKeywordList (Section 6.10.5.1.6, CodedValueKeywordList)

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 (Section 6.10.5.2.4, CodedValueKeywordExists) 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: 2.0

Item identifier: CodedValueKeywordOrKeywordDescendantExists

Semantic signifier for information model:

Scoping entity identifier: org.hssp.dss

Business identifier: HsspDssTraitSchema.CodedValueKeywordOrKeywordDescendantExists

Version: 2.0

Root global element name: CodedValueKeywordOrKeywordDescendantExists

6.10.6 HSSP Simple Evaluation DSS Conformance Profile, Version 2.0

To claim conformance to this profile, a DSS must be conformant with the HSSP Simple Evaluation DSS Functional Profile, version 2.0 (Section 6.10.2, HSSP Simple Evaluation DSS Functional Profile, Version 2.0) and the HSSP Minimum DSS Semantic Profile, version 2.0 (Section 6.10.4, HSSP Minimum DSS Semantic Profile, Version 2.0). The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Simple_Evaluation_DSS_Conformance_Profile

version: 2.0

type: CONFORMANCE_PROFILE

6.10.7 HSSP Complete DSS Conformance Profile, Version 2.0

To claim conformance to this profile, a DSS must be conformant with the HSSP Complete DSS Functional Profile, version 2.0 (Section 6.10.3, HSSP Complete DSS Functional Profile, Version 2.0) and the HSSP Minimum DSS Semantic Profile, version 2.0 (Section 6.10.4, HSSP Minimum DSS Semantic Profile, Version 2.0). The relevant identifying parameters for this profile shall be as follows:

scopingEntityId: org.hssp.dss

businessId: HSSP_Complete_DSS_Conformance_Profile

version: 2.0

type: CONFORMANCE_PROFILE

6.11 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 2.0 (Section 6.10.6, HSSP Simple Evaluation DSS Conformance Profile, Version 2.0).

6.12 Future Specifications of Profiles and Semantic Requirements

It is anticipated that 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 SOAP XML Web Services

The Platform Specific Model (PSM) for SOAP XML Web services is derived from the platform independent model specified in Section 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.

7.1 PSM-Specific Conformance Criteria

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

- HSSP Simple Evaluation DSS Functional Profile, Version 2.0: dssEvaluate.wsdl
- HSSP Complete DSS Functional Profile, Version 2.0: dss.wsdl

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

Also note that while there are minor updates to the schema (xsd file) for this release, the version 2.0 dss.wsdl mentioned above is unchanged from Release 1. The Simple Evaluation DSS Functional Profile, Version 2.0 dssEvaluate.wsdl has been extended to include the evaluateAtSpecifiedTime operation.

8 DSS Platform Specific Model for RESTful XML Web Services

The Platform Specific Model (PSM) for RESTful XML Web services is derived from the platform independent model specified in Section 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). The section on security handling provides some sample guidance on a possible security implementation using HMAC tokens. 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.

8.1 PSM-Specific Conformance Criteria

The PSM conformance criteria correspond to the conformance criteria defined for the PIM in Section 6. One version 2.0 WSDL is provided to correspond with the first functional profile defined in this specification, as follows:

- HSSP Simple Evaluation DSS Functional Profile, Version 2.0: dssEvaluateRest.wsdl

Of note, this wsdl references the OmgDssSchema.xsd to incorporate the DSS schema elements. This is the same schema used by the SOAP profile.

For the purposes of this profile, the only functionality exposed are the *evaluate* and *evaluateAtSpecifiedTime* methods, which support a single evaluation request and response. These methods are modeled as an HTTP POST with the evaluate request as the body of the POST, and the evaluate response as the resulting document. Even though the evaluation is not expected to change state, the call is modeled as an HTTP POST to ensure that the results of the evaluation are not considered a cacheable resource (as they would be with an HTTP GET).

Note also that the service is described using a WSDL 2.0 document. WSDL 2.0 was chosen over WSDL 1.1 for its ability to describe RESTful services. In addition, the RESTful profile differs from the SOAP WSDL in that the types for the operations use the schema types declared in OmgDssSchema directly, rather than indirectly through the *message* construct.

Although the SOAP functional profiles include an HSSP Complete DSS Functional Profile, that functionality is not supported for REST at this time. We expect that it will be at some point in the future.

8.2 Example Security Implementation Using HMAC

As mentioned above, security in general is considered out of scope for the normative aspects of this profile. However, as an informative implementation example, the following scenario describes a potential mechanism for providing security of a RESTful implementation.

The method used is a variation of a symmetric key method designed to minimize chatter while offering robust security. Encryption is not part of the method but is assumed to be happening at the transport layer.
The advantages of this approach are that it is simple to implement, yet still offers robust, low latency authentication using an authentication token that is unique to the time, service URL, request, and user.

The following sections describe the general protocol:

8.2.1 Assumptions

- The service host will not allow anonymous access to Web services.
- The service host implements a user directory and some form of signup/licensing scheme.
- The token creation specification is well documented and unambiguous
- The client will be licensed to use the host's services

8.2.2 Prelude

When a new Web service user account is created, a unique identifier (e.g., a GUID) is generated for the user, and a secret is generated that consists of a random hash value. This information is provided by the service host to the new user one time on account creation.

8.2.3 Transaction

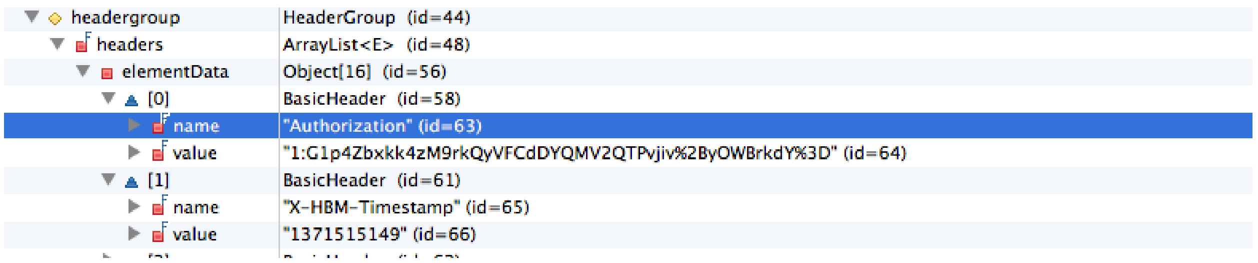
The following steps are involved in securing any transaction from the user:

- The Web service client captures the date-time as seconds since the epoch UTC
- The Web service client concatenates (HttpMethod + URI + RequestBody + Time + UserGuid + UserSecret) into a long string
- The Web server client invokes an HMAC 256 algorithm on the above concatenated string with UserSecret as the Cryptographic key, resulting in a long sequence of bytes, called the token.
- The Web server client converts the token to Base64 and then URLencodes it.
- The Web server client creates two headers in the Web service request stream:
    - X-Time: The same value as was included in creation of the hash
    - Authorization: user-guid : B64Token
- The Web server client sends the request

The following steps are performed by the host to process the authentication and authorization:

- The Web service host receives the request
- The Web service host checks that the X-Time header is recent to within a minute or two (allow for time skew between clocks). If not, returns 403 Forbidden. (Remember that Time has also been part of creating the token)
- The Web service host unpacks the Authorization header and uses the GUID to look up the corresponding secret. The host then checks the user information to determine whether the requesting user is licensed. If the user is not licensed or unknown, the host returns 403 Forbidden.
- The Web service host concatenates HttpMethod + URI + RequestBody + Time + UserGuid + UserSecret.
- The Web service host invokes an HMAC 256 algorithm on the above concatenated string with UserSecret as the Cryptographic key, resulting in a sequence of bytes that should be equivalent to the client token.
- The Web service host converts the token to Base64
- The Web service host then compares the locally created b64 token with the one unpacked from the client's Authorization header. If they match, the vendor can process the request and return the response, otherwise, the vendor returns 401 Unauthorized.

The following snapshot shows an example of the headers of such a request with a sample userId of ‘1’:



```
▼ headergroup          HeaderGroup  (id=44)
  ▼ headers            ArrayList<E>  (id=48)
    ▼ elementData      Object[16]  (id=56)
      ▼ [0]            BasicHeader  (id=58)
        ▶ name         "Authorization" (id=63)
        ▶ value        "1:G1p4Zbxkk4zM9rkQyVFCdDYQMV2QTPvjiv%2ByOWBrkdY%3D" (id=64)
      ▼ [1]            BasicHeader  (id=61)
        ▶ name         "X-HBM-Timestamp" (id=65)
        ▶ value        "1371515149" (id=66)
```

8.3 Exception Handling

Unlike SOAP, REST does not have a standard prescribed methodology for dealing with exceptions. Industry best practices generally align around returning any exceptions as the body of the response document, and selecting an appropriate HTTP Status Code for the response.

The REST profile for DSS therefore prescribes that when an exception occurs, the response document will contain the appropriate exception content as defined in the OmgDssSchema.xsd, and the HTTP Status Code will be set according to the following table:

Exception	Condition	HTTP Status Code
InvalidTimeZoneOffsetException	The specified time zone offset is invalid.	400: Bad Request
UnsupportedLanguageException	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.	400: Bad Request
UnrecognizedLanguageException	The client's specified Language is not recognized.	400: Bad Request
UnrecognizedScopedEntityException	A requested knowledge module does not exist.	400: Bad Request
RequiredDataNotProvidedException	Required data not provided. This exception specifies the data requirement group(s) for which data were required but not provided.	400: Bad Request
InvalidDriDataFormatException	Required data were not provided in the correct format.	400: Bad Request
EvaluationException	An exception occurred during the evaluation process.	500: Internal Server Error
DSSRuntimeException	A DSSRuntimeException is thrown when the DSS service encounters an error at runtime. This exception is used when the error in question is not covered by the other DSS exception types.	500: Internal Server Error

9 Appendices

9.1 Appendix I: 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\dssEvaluateRest.wsdl	Normative	WSDL file of DSS PSM for RESTful 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 WSDLs
Files in Normative Content\Schemas\hl7v3schemas	Normative	XSD files for normative Health Level 7 version 3 information models obtained from the HL7 2012 Version 3 Normative Edition and used by the HSSP Minimum DSS Trait Set Requirement, Version 2.0 (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 2.0 (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

9.2 Appendix II: Relationship to OMG Specifications

OMG Specification	OMG Document Number and/or URL	Relationship to Current Specification
OMG CDSS Specification, Version 1.0	dtc/10-12-07, dtc/10-12-08	Basis of current specification. See Sections 1 and 2 for details.
Unified Modeling Language (UML)	http://www.omg.org/technology/documents/formal/uml.htm formal/07-02-05, formal/07-02-06, ptc/06-10- 06	Used to define PIM.
XML Metadata Interchange Specification (XMI)	http://www.omg.org/technology/documents/formal/xmi.htm formal/2005-09-01	Used to exchange the UML models that define the PIM.
OMG RLUS Technical Specification	health/08-12-03	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.
OMG Entity Identification Service (EIS) Technical Specification	health/08-09-02	EIS is an HSSP service for identifying entities (e.g., patients) across systems. An EIS *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 EIS simply as an Identification Service (IS).

9.3 Appendix III: Related Activites, Documents, and Standards

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).[13]
Relevant standard – HL7 DSS implementation guide	Context-Aware Knowledge Retrieval (Infobutton) Service-Oriented Architecture Implementation Guide Release 1	Implementation guide on the use of a DSS to support context-aware knowledge retrieval.
Relevant standards development activity – HL7	HL7 Virtual Medical Record (vMR) project	The vMR project is defining standard information models for CDS that could be specified as service input or output parameters through the use of semantic

~~13~~

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.

Category	Standard, Activity, or Reference Content	Relationship to the DSS Standard
		signifiers.
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.
Relevant standard – HL7	Arden Syntax	The Arden Syntax is an 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.
Relevant standard – HL7	CDS Work Group GELLO standard	Medical knowledge encoded in GELLO could potentially be exposed to clients using a DSS interface.
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.