HL7_IG_CDS_GUIDE_R1_D1_2013SEP

HL7 Implementation Guide: Decision Support Service, Release 1

September 2013

U.S. Realm DSTU Ballot

Sponsored by:

HL7 Clinical Decision Support and Services Oriented Architecture Work Groups in collaboration with the Standards and Interoperability Framework Health eDecisions Working Group

Identifying Information for Specification:
Specification Name and Release Number: HL7 Implementation Guide: Decision Support Service, Release 1
Realm: U.S.
Ballot Level: DSTU
Ballot Cycle: September 2013
Specification Date: September 2013
Version Number within Release 1: 1.0
Project Sponsor: HL7 Clinical Decision Support Work Group
Project Co-Sponsor: HL7 Services Oriented Architecture Work Group

Note Regarding Realm:
Per guidance from the HL7 Technical Steering Committee, this specification is being balloted as a U.S. Realm specification because its requirements were driven by the U.S. Standards and Interoperability Framework's Health eDecisions initiative (www.healthedecisions.org). However, the underlying HL7 Decision Support Service standard is a Universal Realm standard, and the business requirements addressed by this implementation guide are thought to be similar in countries other than the U.S. Thus, it is anticipated that this implementation guide would provide a viable and useful interoperability solution for Decision Support Services in non-U.S. Realms.

Moving forward, it is anticipated that future releases of the specification may be balloted in the Universal Realm following input from additional stakeholders outside of the U.S.

Acknowledgments

Listed below are the primary authors of this implementation guide.

Name	Organization
Bryn Rhodes	Veracity Solutions
Kensaku Kawamoto	University of Utah
David Shields	University of Utah
Aziz Boxwala	Meliorix

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

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

TABLE OF CONTENTS

TABLE OF CONTENTS 4
1.0 INTRODUCTION 6
1.1 S&I FRAMEWORK HED INITIATIVE 6
1.2 PURPOSE 7
1.3 APPROACH 9
1.3.1 HOW A CDS GUIDANCE SERVICE WORKS 9
1.3.2 OTHER RELEVANT SERVICES 11
1.3.3 TECHNICAL SOLUTION PLAN 12
1.4 INTENDED AUDIENCE 12
1.4.1 REQUISITE KNOWLEDGE 12
1.4.2 REFERENCED STANDARDS 13
1.5 CONVENTIONS AND ACRONYMS USED IN THIS GUIDE 14
1.5.1 CONFORMANCE VERBS (KEYWORDS) 14
1.5.2 CARDINALITY 14
1.6 ACRONYMS 15
2.0 IMPLEMENTATION APPROACH 16
2.1 PRE-CONDITIONS (REQUIRED ATTRIBUTES OF A SUITABLE ENVIRONMENT FOR IMPLEMENTING THIS GUIDE) 16
2.2 IMPLEMENTATION RESOURCES 17
2.3 SERVICE INTERACTION FRAMEWORK 18
2.3.1 DSS PROFILE 18
2.3.2 EXAMPLE SERVICE REQUEST 19
2.3.3 EVALUATE REQUEST 20
2.3.4 EVALUATE RESPONSE 24
2.3.5 EVALUATE RESPONSE 25
2.3.6 DSS EXCEPTIONS 28
2.3.7 SERVICE PROTOCOLS 29
2.4 CONTENT PAYLOAD DEFINITION 30
2.4.1 SEMANTIC SIGNIFIER 30
2.4.2 CDSINPUT 32
2.4.3 CDS OUTPUT 33
2.4.4 EXECUTION MESSAGE CONTAINER 37
2.5 CLINICAL DATA REPRESENTATION 39
2.6 SPECIFICATION OF KNOWLEDGE MODULE META-DATA 40

2.6.1 EXAMPLE SPECIFICATION OF KNOWLEDGE MODULE META-DATA: PNEUMOCOCCAL VACCINATION FOR INDIvIDUAL WITH DIABETES MELLITUS 41

2.7 EXAMPLE IMPLEMENTATION 46

2.7.1 SAMPLE CLIENT REQUEST 46

2.7.2 RECEIVING THE REQUEST 47

2.7.3 RETURNING THE RESPONSE 48

2.7.4 PROCESSING THE RESPONSE 49

2.7.5 EXCEPTION HANDLING 49

3.0 APPENDICES 51

APPENDIX A: ACRONYMS 51

APPENDIX B: GLOSSARY OF TERMS 53

1.0 INTRODUCTION

Clinical Decision Support (**CDS**) provides clinicians, staff, patients, or other individuals with knowledge and person-specific information, intelligently filtered or presented at appropriate times, to enhance health and health care. Effective CDS interventions require availability of computable biomedical knowledge, person-specific data, and a reasoning or inference mechanism that combines these elements to generate and present helpful and actionable information to clinicians, individuals, or caregivers in the right way and at the right time.

As with other types of applications, a CDS system can be more easily implemented and maintained if software services are available to provide functionality required by the application. In particular, a Decision Support Service (**DSS**) can facilitate the implementation of a CDS system by performing a key task required for providing CDS (the analysis of patient data to generate patient-specific assessments and recommendations).

In recognition of the importance of a DSS, HL7 has collaborated with the Object Management Group (**OMG**) to define a standard DSS interface known as the HL7 DSS standard. The purpose of this implementation guide is to define a DSS implementation approach that combines the HL7 DSS standard with other relevant standards, in particular the HL7 Virtual Medical Record for Clinical Decision Support (**vMR-CDS)** information model standard.

The requirements for this U.S. Realm implementation guide were derived primarily from the U.S. Standards and Interoperability (**S&I**) Framework's Health eDecisions (**HeD**) initiative (www.healthedecisions.org). However, the business requirements addressed by this implementation guide are thought to be similar in countries other than the U.S. Thus, it is anticipated that this implementation guide would provide a viable and useful interoperability solution in non-U.S. Realms.

1.1 S&I Framework HeD Initiative

Because the requirements for this implementation guide were developed through the HeD initiative, information is provided here regarding the initiative.

The goal of the S&I Framework's HeD initiative is to identify, define and harmonize standards that facilitate the emergence of systems and services whereby shareable CDS interventions can be implemented via:

- Standards to structure medical knowledge in a shareable and executable format for use in CDS (Use Case 1: "CDS Artifact Sharing"), and
- Standards that define how a system can interact with and utilize an electronic service that provides helpful, actionable clinical guidance (Use Case 2: "CDS Guidance Service")

This implementation guide focuses on Use Case 2: CDS Guidance Service. In this guide, the terms **CDS Guidance Service** and DSS will be used synonymously.

In this implementation guide, interoperability relates to the exchange of information that allows the delivery of the results derived from the execution of a CDS Web service. This implementation guide addresses a scenario in which a system sends information on a specific patient to a CDS Guidance Supplier that provides conclusions and recommendations to facilitate decision-making by the clinical user.

Table 1 includes links to obtain more information about the S&I Framework, CDS, and consensus-approved documents for the HeD Initiative.

Table 1: Introduction References

Title	Link
S&I Framework	http://siframework.org/
HeD Project Charter	http://wiki.siframework.org/Health+eDecisions+Project+Charter+and+Members
HeD Use Cases	http://wiki.siframework.org/Health+eDecisions+Use+Case
HL7 CDS Knowledge Artifact Implementation Guide (HeD Use Case 1 Implementation Guide) (Note that the notion of action group from HeD Use Case 1 is leveraged in Use Case 2 as a part of CDS outputs; see Section 0)	http://wiki.hl7.org/index.php?title=HL7_CDS_Standards

1.2 Purpose

The purpose of this implementation guide is to enable the use of standardized CDS Guidance Services through the leveraging of base standards. These leveraged standards include:

- The HL7 DSS standard as the service standard
- The HL7 Virtual Medical Record (**vMR**) standard for the primary content standard.
    - Of note, the vMR has been harmonized with the semantics, terminologies, and templates of the Consolidated Clinical Document Architecture (**C-CDA**) and Quality Reporting Document Architecture (**QRDA**) to the greatest extent possible.
- The **SOAP** and Representational State Transfer (**REST**) transport standards for Web services

The primary standards leveraged by this implementation guide are summarized in **Table 2**.

Table 2: Standards Leveraged by Implementation Guide

Transaction	Transport	Service Standard	Primary Content Standard
CDS Request (including *patient data and potentially context*)	SOAP/REST	DSS	vMR
CDS Response (including CDS *guidance and/or other response elements*)	SOAP/REST	DSS	vMR

Several assumptions and pre-conditions that were documented within the HeD Use Case specification for the CDS Guidance Service (see Section 1.1) apply to this implementation guide. The HeD community identified several transaction pairs leading to an ideal CDS Guidance Service, but narrowed the scope to just one transaction pair for the Guidance Service:

1) the CDS Guidance Requestor sends a request for CDS to the CDS Guidance Supplier, and
2) the CDS Guidance Supplier sends CDS to the CDS Guidance Requestor.

The other transaction pairs considered to be important for CDS, but *out of scope* for this implementation guide, include the query and response for a list of knowledge modules available from the CDS Guidance Supplier and the query and response for the metadata of available knowledge modules, including a specification of the data required and the valuation result to be returned. DSS operations corresponding to these interactions MAY be used to provide this information. However, as discussed further in Section 2.6, a DSS provider may use alternate methods such as documentation to provide this information to a DSS client.

The scope of this implementation guide is graphically depicted in **Figure 1** below. Additional information on assumptions and pre-conditions can be found in Section 2.1.

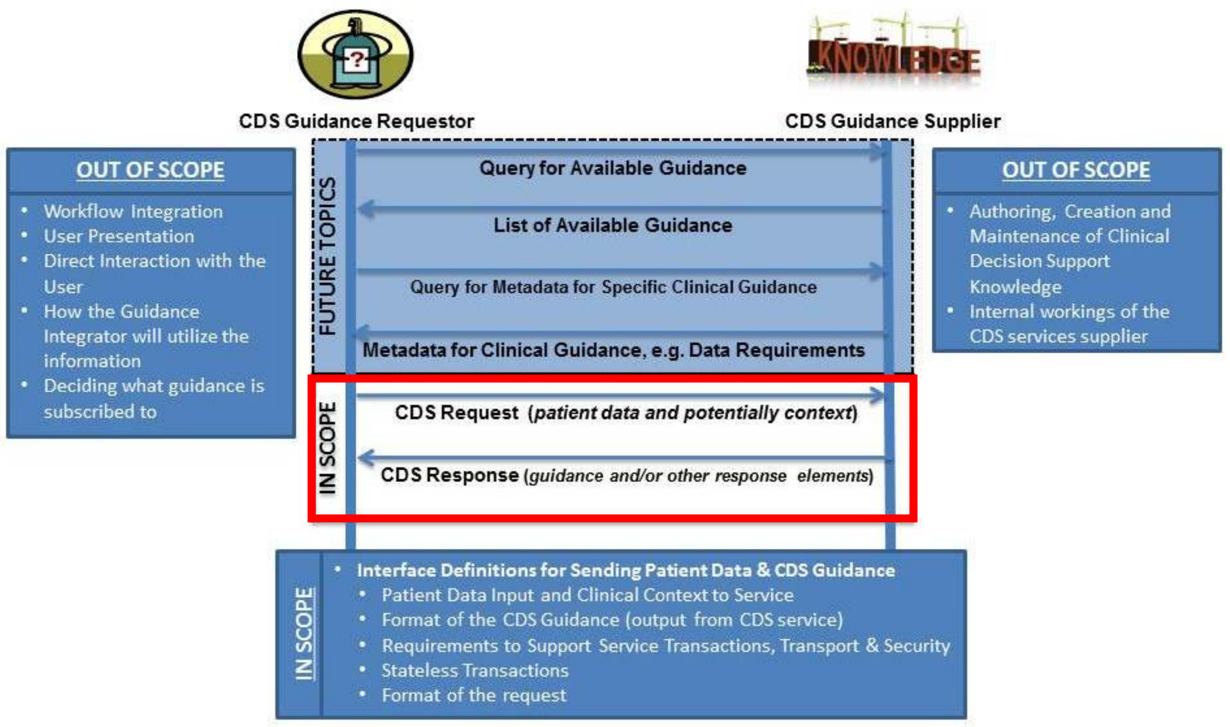
CDS Guidance Requestor
KNOWLEDGE
CDS Guidance Supplier
OUT OF SCOPE
Workflow Integration
User Presentation
Direct Interaction with the User
How the Guidance Integrator will utilize the information
Deciding what guidance is subscribed to
FUTURE TOPICS
Query for Available Guidance
List of Available Guidance
Query for Metadata for Specific Clinical Guidance
Metadata for Clinical Guidance, e.g. Data Requirements
OUT OF SCOPE
Authoring, Creation and Maintenance of Clinical Decision Support Knowledge
Internal workings of the CDS services supplier
IN SCOPE
CDS Request (patient data and potentially context)
CDS Response (guidance and/or other response elements)
IN SCOPE
Interface Definitions for Sending Patient Data & CDS Guidance
Patient Data Input and Clinical Context to Service
Format of the CDS Guidance (output from CDS service)
Requirements to Support Service Transactions, Transport & Security
Stateless Transactions
Format of the request

Figure 1: Scope of Implementation Guide

1.3 Approach

This guide is designed to support the following implementation objectives:

1. To provide an overview of the standards and specifications used in this implementation guide.
2. To specify how the full stack of base standards can be leveraged to standardize the use of CDS Guidance Services.

This implementation guide focuses not only on the structure of the in-scope transactions, but also on the message semantics through the use of standard terminologies, value sets and taxonomies such as SNOMED CT.

1.3.1 HOW A CDS GUIDANCE SERVICE WORKS

The user of a clinical information system (CDS Guidance Requestor) would like to receive clinical guidance for a particular patient. Alternatively, the clinical information system automatically determines the need for clinical guidance based on clinical observations. In order to obtain this guidance, the CDS Guidance Requestor generates one or more guidance requests, which are sent to the CDS Service Provider (CDS Guidance Supplier). The request includes patient clinical data as well as potentially contextual information such as the user role and type, the encounter setting, and the type and/or scope of CDS guidance being requested. This request may or may not include a specific time point to use for the evaluation.

The CDS Guidance Supplier will evaluate temporal logic against the specified evaluation time point. If an evaluation time point is not furnished, then the CDS Guidance Supplier will use the current time in its system. This evaluation time point may be in the past (for instance, as of the end of an evaluation period or calendar year), or it may be in the future (such as the time of the patient's next appointment). For example, if a CDS Guidance Requestor wishes to request guidance a day prior to outpatient visits for performance reasons, it may request an evaluation to be done using the date and time of the appointment as the evaluation time point. Similarly, a CDS Guidance Requestor wishing to evaluate patients for a quality measure may request an evaluation to be done using a date required by the quality measure, such as December 31st of the previous year. Finally, to conduct regression batch testing, a CDS Requestor may wish to use static test cases along with a static evaluation time point, so that the CDS evaluation results do not unintentionally change over time as the date changes.

The CDS Guidance Supplier receives and processes the request. The CDS Guidance Supplier returns the clinical guidance and/or other response elements (e.g., error messages) for each request to the CDS Guidance Requestor. This information is then made available to the end user. **Figure 2** provides a summary of this interaction.

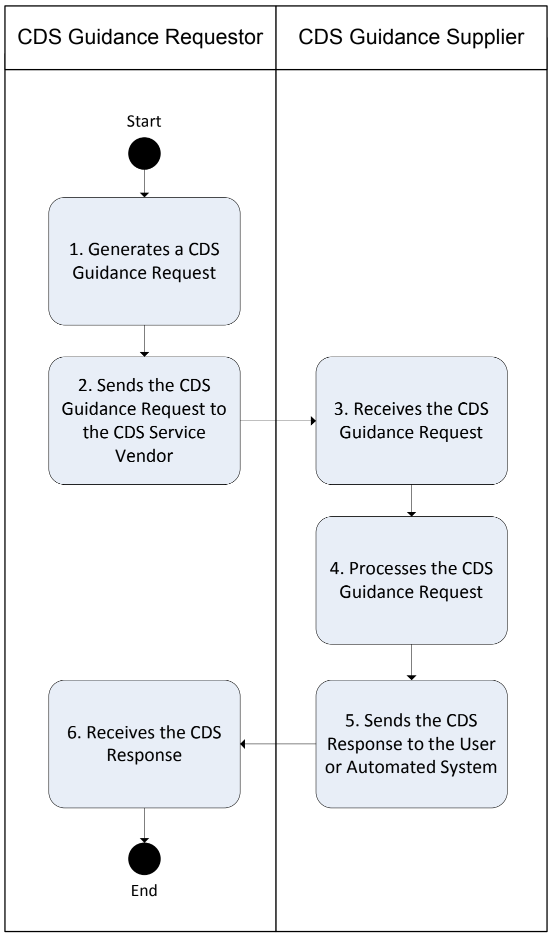
CDS Guidance Requestor
CDS Guidance Supplier
Start
1. Generates a CDS Guidance Request
2. Sends the CDS Guidance Request to the CDS Service Vendor
3. Receives the CDS Guidance Request
4. Processes the CDS Guidance Request
5. Sends the CDS Response to the User or Automated System
6. Receives the CDS Response
End

Figure 2: Activity Diagram for Interaction Between DSS Client and Provider

1.3.2 OTHER RELEVANT SERVICES

In implementing a CDS system, services other than a DSS may be useful. **Table 3**, which is adapted from a table in the HL7 DSS Release 2 specification, provides a sample of such services. Of note, standards for many of these services have already been developed, or under current development, by HL7 and OMG through the HL7-OMG Healthcare Services Specification Program (**HSSP**). Of note, the DSS standard is a product of the HSSP effort. The reader is directed to http://hssp.wikispaces.com/ for the latest information on the HSSP family of service standards.

Table 3. Sample of Other Services that may be Useful for Implementing a CDS System

Service	**Description**	**Example of Service Use by a CDS System**
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 CDS 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/IXS)	Allows the service client to identify entities (e.g., patients) across systems.	When determining whether a patient is in need of an influenza vaccine, a CDS system associated with Health System A uses 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 RLUSs of the health department and of Clinic B, and the CDS system requests that the RLUSs 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.
Retrieve, Locate, Update Service (RLUS)	Allows the service client to locate, retrieve, and update records for a patient across systems.	See example above for EIS.

1.3.3 TECHNICAL SOLUTION PLAN

The HL7 DSS standard is used as the interface for requesting and receiving a CDS evaluation result. The primary standard used for the CDS input and output content payloads is the HL7 vMR standard.

1.4 Intended Audience

The intended audience of this implementation guide is the developers of CDS Guidance Services and the developers of clinical information systems that use such services.

1.4.1 REQUISITE KNOWLEDGE

This section identifies pre-requisites for users of this implementation guide.

Table 4. Prerequisite Knowledge for Implementers

Required to Know	**Should Know**
1) Implementers must have a strong understanding of XML and related technologies, such as XML Schema and SOAP or REST. 2) Implementers should have a strong knowledge of the HL7 standards underlying this implementation guide, specifically the HL7 DSS standard, the vMR XML Implementation Guide, HL7 vMR Templates, and the HL7 CDS Knowledge Artifact Implementation Guide (see Section 1.4.2).	1) Implementers should have a basic understanding of the following vocabularies and value sets: • CPT • CVX • NDF-RT • LOINC • SNOMED-CT • ICD-9 • ICD-10 • HITSP C80 • RxNorm • MeSH

1.4.2 REFERENCED STANDARDS

Table 5 lists the standards referenced in this implementation guide.

Table 5. Referenced Standards

Standard	Description
HL7 DSS, Release 2	A DSS facilitates the delivery of CDS by receiving patient data as the input and returning patient-specific conclusions as the output. The HL7 DSS specification provides a standard interface for the provision and consumption of such services.
Object Management Group (OMG) Clinical Decision Support Service (CDSS) Standard, Version 1.0	The DSS standard is a part of the HL7-OMG Healthcare Services Specification Program (HSSP), which is a collaboration between HL7 and OMG to develop standard interface specifications for software services important to health care.[1,2] The OMG CDSS standard serves as the foundation for the HL7 DSS standard. It is anticipated that the OMG CDSS standard will be updated to be synchronized with the latest version of the HL7 DSS standard.
HL7 vMR XML Implementation Guide, Release 1, Version 2.0	A vMR is a data model for representing the data that are analyzed and/or produced by CDS engines. The XML Implementation Guide describes how to implement the vMR using XML.
HL7 vMR Templates, Release 1	This specification defines constraints on the vMR to facilitate semantic interoperability. These templates are aligned with the semantics of the C-CDA and QRDA wherever possible.
HL7 CDS Knowledge Artifact Implementation Guide, Release 1	This implementation guide was developed to support the HeD Artifact Sharing Use Case. The Action Groups construct within the CDS Knowledge Artifact Implementation Guide is leveraged in this specification.

[1] Healthcare Services Specification Program (HSSP) Home Page. Available at: http://hssp.wikispaces.com/.

[2] Kawamoto K, Honey A, and Rubin K. The HL7-OMG Healthcare Services Specification Project: motivation, methodology, and deliverables for enabling a semantically interoperable service-oriented architecture for healthcare. *J Am Med Inform Assoc.* 2009;16:874-81.

1.5 Conventions and Acronyms Used in this Guide

1.5.1 CONFORMANCE VERBS (KEYWORDS)

Conformance Verb (aka keywords) is defined throughout this implementation guide using **BOLD** and CAPS to denote the conformance criteria to be applied.

The keywords **SHALL, SHOULD, MAY, NEED NOT, SHOULD NOT,** and **SHALL NOT** in this document are to be interpreted as described in the *HL7 Version 3 Publishing Facilitator's Guide*[3]::

- **SHALL**: an absolute requirement
- **SHALL NOT**: an absolute prohibition against inclusion
- **SHOULD/SHOULD NOT**: best practice or recommendation. There may be valid reasons to ignore an item, but the full implications must be understood and carefully weighed before choosing a different course
- **MAY/NEED NOT**: truly optional; can be included or omitted as the author decides with no implications

Much of the conformance requirements are specified in the underlying standards. The **SHALL** and **SHALL NOT** conformance verbs relating to requirements that are only defined in this implementation guide are underlined as well for distinction.

1.5.2 CARDINALITY

The following table represents the *Cardinality* of elements within this guide. *Cardinality* is defined by the minimum and maximum number of times that the data element may appear.

The cardinality indicators are interpreted with the following format "m…n" where m represents the least and n the most:

Table 6. Cardinality

Cardinality	Description
0..0	The element is never present
0..1	The element MAY be omitted and has at most one occurrence
1..1	The element is present once and only once
0..n	The element MAY be omitted or may repeat up to *n* times
1..n	The element MUST appear at least once, and MAY repeat up to n times

[3] http://www.hl7.org/v3ballot/html/help/pfg/pfg.htm

Cardinality	**Description**
0..*	The element MAY be omitted, or it MAY repeat an unlimited number of times
1..*	The element MUST appear at least once, and MAY repeat an unlimited number of times
m..n	The element MUST appear at least *m* times, and at most, *n* times
2..2	The element MUST appear two and only two times
3..3	The element MUST appear three and only three times

1.6 Acronyms

Table 7 below lists the acronyms used most frequently within this implementation guide. A full listing of acronyms can be found in the Appendix (Section 3.0).

Table 7. Acronyms

Term	Definition
DRI	Data Requirement Item
DSS	Decision Support Service
KM	Knowledge Module
SS	Semantic Signifier
vMR	Virtual Medical Record

2.0 IMPLEMENTATION APPROACH

2.1 Pre-Conditions (Required Attributes of a Suitable Environment for Implementing this Guide)

- The CDS Guidance Requestor's system is pre-configured to identify triggers to request clinical guidance.
- The CDS Guidance Requestor's system is pre-configured to receive the clinical guidance data and integrate it into the system.
- The CDS Service Integrator is able to identify the type of CDS guidance required for specific scenarios (e.g., immunization reminders).
- The end-point address for making the request for CDS Guidance can be identified.
- The CDS Guidance Requestor's system is aware of and able to supply the input parameter values (the clinical information and context).
- The CDS Guidance Requestor's system is able to map and transform to and from the terminology and data model standards specified in this guide, either natively or by way of a third party (which might be the CDS Guidance Supplier itself).
- The CDS Guidance Supplier is able to formulate non-guidance components of the response (such as error messages).
- The CDS Guidance Requestor is able to properly process the defined non-guidance components of the response (such as errors in submitted data, additional information required, etc.). This would include things such as notifying technical staff of operational / data format errors, and notifying the end user, if appropriate, of the need for additional data.

2.2 Implementation Resources

The HL7 DSS Standard, Release 2 includes the following machine-consumable files, which are also included in this implementation guide as resources.

File(s)	Status	Description
Normative Content\PSM\dss.wsdl (not required)	Normative	WSDL file of SOAP implementation of DSS PSM for SOAP XML Web services. Supports the complete DSS functional profile, which includes additional service operations that are not required in this implementation guide, but which MAY be used to provide the service meta-data outlined in Section 2.6.
Normative Content\PSM\ dssEvaluate.wsdl	Normative	WSDL file of DSS PSM for SOAP XML Web services. Supports the simple evaluation functional profile, which includes only those operations required in this implementation guide.
Normative Content\PSM\dssEvaluateRest.wsdl	Normative	WSDL file of DSS PSM for RESTful XML Web services. Supports the simple evaluation functional profile. The service payloads are the same as for the SOAP implementation.
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 used by DSS WSDLs
Files in Normative Content\Schemas\ hl7v3schemas (not required)	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 optional OmgDssTraitSchema below.
Normative Content\Schemas\ hsspschemas\OmgDssTraitSchema.xsd (not required)	Normative	XSD file used the by HSSP Minimum DSS Trait Set Requirement, Version 2.0. that are not required in this implementation guide, but which MAY be used to provide the service meta-data outlined in Section 2.6.

In addition, this implementation guide includes the following supplemental files as implementation resources:

- The XML schemas from the HL7 vMR XML Implementation Guide Release 1 Version 2.0
- The vMR templates from the HL7 vMR Templates Release 1 Version 1.0 specification

2.3 Service Interaction Framework

This section of the implementation guide discusses the service interaction framework used by the HL7 DSS standard to provide standards-based CDS functionality. Within the HL7 DSS standard, this interaction framework is agnostic of the content payloads used. For this implementation, there is a specific binding to the use of the content payloads that leverage the HL7 vMR standard and are defined in the next section (Section 2.4).

This layering allows for flexible service implementation by separating the framework used by a DSS from the content payloads used to communicate the required input data and the returned evaluation results.

2.3.1 DSS PROFILE

DSS implementations conformant with this implementation guide **SHALL** conform to the HSSP Simple Evaluation DSS Functional Profile, defined in the HL7 DSS standard, Release 2. This profile includes only the *evaluate* and *evaluateAtSpecifiedTime* interfaces and is the minimum level of support required for a service to claim conformance with the DSS standard.

Note that use of *evaluateAtSpecifiedTime* with a specified time of now is equivalent to use of the *evaluate* operation. Implementers may find it convenient to implement the *evaluate* operation simply by calling *evaluateAtSpecifiedTime* internally in this manner. The rationale for supporting *evaluateAtSpecifiedTime* is described in Section 1.3.1.

2.3.2 EXAMPLE SERVICE REQUEST

Figure 3 provides a sample DSS request (implementable either via SOAP or REST) for the *evaluateAtSpecfiedTime* request. Note that the *evaluate* request is identical, except that the top-level element is “evaluate” rather than “evaluateAtSpecifiedTime” and there is no “specifiedTime” element. A brief explanation of each component of this request is provided as in-line comments in **Figure 3**, and a more detailed description is provided in the sections that immediately follow. Note that this example corresponds with the KM metadata described in Section 2.6.1.

```
<dss:evaluateAtSpecifiedTime>
        <!-- unique identifier of interaction provided by service requestor -->
    <interactionId scopingEntityId="edu.utah.bmi" interactionId="123456"
submissionTime="2012-01-20T07:59:35.123"/>
        <!-- the evaluation time to be used by the DSS provider -->
    <specifiedTime>2012-01-01T00:00:00.000</specifiedTime>
        <!-- wrapper for the evaluation request -->
    <evaluationRequest clientLanguage="en-US" clientTimeZoneOffset="-05:00">
            <!-- identification of knowledge module(s) to use for the patient evaluation -->
        <kmEvaluationRequest>
            <kmId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
        </kmEvaluationRequest>
            <!-- wrapper for the patient data payload -->
        <dataRequirementItemData>
                <!-- identification of KM data requirement item being fulfilled by the data payload -->
            <driId itemId="SoleDataRequirementItemForKM">
                <containingEntityId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
            </driId>
            <data>
                    <!-- identification of format used for providing patient clinical data -->
                <informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsinput:r2:CDSInput" version="2.0"/>
                    <!-- actual patient data is in the specified format, and base64 encoded (described in Section
2.4) -->
                <base64EncodedPayload>[base 64 encoded content from Section
2.4]</base64EncodedPayload>
            </data>
        </dataRequirementItemData>
    </evaluationRequest>
</dss:evaluateAtSpecifiedTime>
```

Figure 3. Sample DSS Request

2.3.3 EVALUATE REQUEST

The primary components of the evaluate request are outlined in **Table 8**. Each of these components is covered in more detail in the following sections.

Table 8. Components of Evaluate Request

Component	Description
scopingEntityId	Pre-coordinated identifier used to provide a unique scope for identifiers generated in different contexts, such as an organization or a knowledge module (**KM**)
interactionId	Client-generated identifier for the request
specifiedTime	Client-specified evaluation time
evaluationRequest	Container for the request information
kmEvaluationRequest	Indicates which KMs should be evaluated for the request
dataRequirementItemData	Contains the data required for the request

2.3.3.1 SCOPING ENTITY IDENTIFIER

Examples from **Figure 3**:

edu.utah.bmi
com.cdsvendor
org.hl7.cds

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" **SHALL** 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 (http://www.iso.org/iso/country_names_and_code_elements). Subsequently, the "id" **SHALL** start by defining the domain name that is associated with the scoping entity (e.g., "com.zynxhealth," "com.dbmotion," "edu.utah," "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.

2.3.3.2 INTERACTION IDENTIFIER

Example from **Figure 3**:

```
<interactionId scopingEntityId="edu.utah.bmi" interactionId="123456"
submissionTime="2012-01-20T07:59:35.123"/>
```

The request contains an Interaction Identifier which specifies the client-generated unique identifier for the request. This identifier **SHALL** be included as the requestId element of the response that is returned by the service. The client supplies values for the *scopingEntityId*, *interactionId*, and *submissionTime* attributes of the element.

- ***scopingEntityId*** (required): **SHALL** be set to the client's assigned scoping entity identifier as a string.
- ***interactionId*** (required): The client **SHALL** supply an interaction identifier as a string.
- ***submissionTime*** (required): The client **SHALL** supply a submission time using the xs:datetime data type.

2.3.3.3 EVALUATION REQUEST

Example from **Figure 3**:

```
<evaluationRequest clientLanguage="en-US" clientTimeZoneOffset="-05:00">
   ...
</evaluationRequest>
```

The evaluation request **SHALL** specify the client language and client time zone offset of the request.

Language **SHALL** be specified as either a 2-character ISO 639-1 language code or a combination of a 2-character ISO 639-1 language code and a 2-character ISO 3166-1 geographical code, concatenated with a hyphen. Example valid language specifications include: "en," "en-US," "en-GB," and "fr." ISO 639-1 codes are available at http://www.loc.gov/standards/iso639-2/php/English_list.php, and ISO 3166-1 codes are available at
http://www.iso.org/iso/home/standards/country_codes/country_names_and_code_elements.htm.

The client's time zone is offset from Universal Coordinated Time (UTC). The time zone SHALL be in the form "**±[hh]:[mm]"**, e.g., "-05:00". 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 **SHALL** be assumed to have this time zone offset.

2.3.3.4 KNOWLEDGE MODULE EVALUATION REQUEST

Example from **Figure 3**:

```
      <kmEvaluationRequest>
           <kmId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
       </kmEvaluationRequest>
```

Each request **SHALL** specify at least one KM identifier, indicating which KMs will be evaluated. The KMs available within any given service implementation will vary. DSS discovery interfaces can be implemented to provide programmatic access to this information, but at a minimum, the CDS provider **SHALL** provide clients with a catalog of available knowledge modules. Further information on recommended metadata to provide to DSS clients is in Section 2.6.

Note that the scopingEntityId in this case identifies the scoping entity for the KM, not the client.

2.3.3.5 DATA REQUIREMENT ITEM DATA

Example from **Figure 3**:

```
      <dataRequirementItemData>
           <!-- identification of KM data requirement item being fulfilled by the data payload -->
           <driId itemId="SoleDataRequirementItemForKM">
               <containingEntityId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
           </driId>
           <data>
               <!-- identification of format used for providing patient clinical data -->
               <informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsinput:r2:CDSInput" version="2.0"/>
               <!-- actual patient data is in the specified format, and base64 encoded -->
               <base64EncodedPayload>[base 64 encoded content from Section
2.4]</base64EncodedPayload>
           </data>
        </dataRequirementItemData>
```

As defined in the DSS standard, each KM specifies a set of data requirements. Required data are specified in terms of data requirement items (DRIs). The DSS provider identifies each DRI with a DRI identifier. An example from **Figure 3** is shown below:

```
           <driId itemId="SoleDataRequirementItemForKM">
               <containingEntityId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
           </driId>
```

To fulfill a DRI, DRI Data are provided with the DRI identifier specifying which DRI is being fulfilled, and a “data” section (example below) providing the required data.

```
            <data>
                 <!-- identification of format used for providing patient clinical data -->
                <informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsinput:r2:CDSInput" version="2.0"/>
         <!-- actual patient data is in the specified format, and base64 encoded -->
                <base64EncodedPayload>[base 64 encoded content from Section
2.4]</base64EncodedPayload>
             </data>
```

See Section 2.6 later in this document for how this payload information is specified.

2.3.4 EVALUATE RESPONSE

Figure 4 provides a sample DSS response (returned either via SOAP or REST) for the *evaluateAtSpecfiedTime* request. Note that the response to the *evaluate* request is identical, except that the top-level element is “evaluateResponse” rather than “evaluateAtSpecifiedTimeResponse.” A brief explanation of each component of this request is provided as in-line comments in the example below, and a more detailed description is provided in the sections immediately below.

```
<dss:evaluateAtSpecifiedTimeResponse>
    <!-- unique identifier of interaction provided by service requestor -->
    <requestId scopingEntityId="edu.utah.bmi" interactionId="123456"
submissionTime="2012-01-20T07:59:35.123"/>
    <!-- unique identifier of response provided by service provider -->
    <responseId scopingEntityId="com.cdsvendor" interactionId="987654321"
submissionTime="2012-01-20T07:59:35.567"/>
    <!-- wrapper for the response from the service provider -->
    <evaluationResponse>
        <finalKMEvaluationResponse>
            <!-- identification of KM(s) generating the patient evaluation -->
            <kmId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
            <!-- any warnings generated by the KM during the patient evaluation.  If provided, uses same
approach as the “data” section below.  -->
            <warning/>
            <kmResponseId/>
            <!-- wrapper for the patient output data payload -->
            <kmEvaluationResultData>
                <!-- identification of KM evaluation result being provided -->
                <evaluationResultId itemId="testPayload.EvaluationResult">
                    <containingEntityId scopingEntityId="com.cdsvendor"
businessId=”OID for response template" version="1.0.0"/>
                </evaluationResultId>
                <data>
                    <!-- identification of specific format used for providing evaluation results -->
                    <informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsoutput:r2:CDSOutputAsVMR"
                        version="2.0"/>
                    <!-- actual patient response data is in the specified format, and base64 encoded (described
in Section 2.4) -->
                    <base64EncodedPayload>[base 64 encoded content from Section 2.4]
</base64EncodedPayload>
                </data>
            </kmEvaluationResultData>
        </finalKMEvaluationResponse>
    </evaluationResponse>
</dss:evaluateAtSpecifiedTimeResponse>
```

Figure 4. Sample DSS Response

2.3.5 EVALUATE RESPONSE

The DSS standard allows for single and iterative evaluations, and the same response model is used to support both scenarios. For the purposes of this guide, the iterative interaction model is out of scope. For more information, see the DSS standard.

The evaluate response contains the following primary components, each of which is considered in more detail in the following sections.

Table 9. Components of Evaluate Response

Component	Description
FinalKMEvaluationResponse	Describes the results of evaluation for each requested KM, one per KM.
Warning	Optional component which may contain any number of warnings generated by evaluation of the KM.
KMEvaluationResultData	Contains any number of results generated by evaluation of the KM.

2.3.5.1 EVALUATE RESPONSE

Example from **Figure 4**:

```
<dss:evaluateAtSpecifiedTimeResponse>
    <!-- unique identifier of interaction provided by service requestor -->
    <requestId scopingEntityId="edu.utah.bmi" interactionId="123456"
submissionTime="2012-01-20T07:59:35.123"/>
    <!-- unique identifier of response provided by service provider -->
    <responseId scopingEntityId="com.cdsvendor" interactionId="987654321"
submissionTime="2012-01-20T07:59:35.567"/>
    <!-- wrapper for the response from the service provider -->
    <evaluationResponse>
        <finalKMEvaluationResponse>
             ...
         </finalKMEvaluationResponse>
    </evaluationResponse>
</dss:evaluateAtSpecifiedTimeResponse>
```

The request ID **SHALL** be the ID provided by the CDS Guidance Requestor in the request.

The response ID **SHALL** be provided by the CDS Guidance Provider to the client. It **SHALL** be a unique identifier within the scope of the scoping entity.

2.3.5.2 FINAL KM EVALUATION RESPONSE

Example from **Figure 4**:

```
 <finalKMEvaluationResponse>
                <!-- identification of KM(s) generating the patient evaluation -->
                <kmId scopingEntityId="com.cdsvendor"
businessId="DiabetesMellitusNeedForPneumococcalVaccination" version="1.0.0"/>
                 <!-- any warnings generated by the KM during the patient evaluation.  If provided, uses same
approach as the "data" section below.  -->
                <warning/>
               <kmResponseId/>
               <!-- wrapper for the patient output data payload -->
                <kmEvaluationResultData>
                    ...
               </kmEvaluationResultData>
</finalKMEvaluationResponse>
```

This element contains the identifier of the knowledge module that was evaluated.

This element **MAY** also contain a warning (see section immediately below), as well as an optional KM Response ID to uniquely identify the evaluation results from the specific KM. The Final KM Evaluation Response also contains KM Evaluation Result Data, as described below.

2.3.5.3 WARNING

The response **MAY** contain any number of warnings that were generated during the evaluation of the KM. Each warning has a single semantic payload which contains the base-64 encoded content of the warning (see Section 2.4).

2.3.5.4 KM EVALUATION RESULT DATA

Example from Figure 4:

```
 <kmEvaluationResultData>
                    <!-- identification of KM evaluation result being provided -->
                    <evaluationResultId itemId="testPayload.EvaluationResult">
                        <containingEntityId scopingEntityId="com.cdsvendor"
businessId="OID for response template" version="1.0.0"/>
                    </evaluationResultId>
                    <data>
                        <!-- identification of specific format used for providing evaluation results -->
                        <informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsoutput:r2:CDSOutputAsVMR"
                            version="2.0"/>
                        <!-- actual patient response data is in the specified format, and base64 encoded (described
in Section 2.4) -->
                        <base64EncodedPayload>[base 64 encoded content from Section 2.4]
</base64EncodedPayload>
                    </data>
</kmEvaluationResultData>
```

As defined in the DSS standard, each KM specifies a set of evaluation results that will be returned. The DSS provider identifies each evaluation result to be returned with an Evaluation

Result identifier (see Section 2.6 for how a DSS provider **MAY** provide such meta-data to its clients).

In the DSS evaluation response, the KM Evaluation Result Data will be identified in terms of its Evaluation Result identifier. The KM Evaluation Result Data will also specify the information model used to deliver the evaluation results, as well as the actual evaluation results using the specified information model and encoded in base 64. See Section 2.4 for how the payload content is represented.

2.3.6 DSS EXCEPTIONS

The DSS standard defines several exceptions that can be thrown at the DSS level. For implementations using SOAP, these exceptions **SHALL** be returned as SOAP faults. For REST implementations, the exception message **SHALL** be returned as the body of the response, with an appropriate HTTP Status Code as specified in the DSS Standard.

The exceptions defined as part of the DSS Standard and which **SHALL** be used by implementations conformant with this guide are listed in the table below.

Table 10. DSS Exceptions

SOAP Exception	Condition	REST 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

2.3.7 SERVICE PROTOCOLS

The DSS standard defines both SOAP and REST-based platform-specific profiles for the Simple Evaluation functional profile. Selecting which protocol to use is an architectural decision based on the best fit for the implementation environment. As such, this guide makes no recommendation, and allows for either protocol to be used.

Similarly, the security standards and protocols used depend on a number of factors including the implementation environment, licensing and business arrangements, sensitivity of information in the calls, and many others. Security is therefore considered out of scope for the purpose of this document. However, implementers **SHALL** ensure appropriate secure operation of all components as needed, especially if the possibility exists for requests and responses to contain identifiable patient health information.

2.3.7.1 SOAP PROTOCOL

For the SOAP protocol, the DSS standard provides a 1.1 WSDL document that describes a conforming service. This WSDL is defined in the dssEvaluate.wsdl file, located in the NormativeContent\PSM folder of the DSS standard. This WSDL is provided as a supplement to this implementation guide (Section 2.2).

2.3.7.2 REST PROTOCOL

For RESTful implementations, the DSS standard provides a 2.0 WSDL document that describes a conforming service. This WSDL is defined in the dssEvaluateRest.wsdl file, located in the NormativeContent\PSM folder of the DSS standard. This WSDL is provided as a supplement to this implementation guide (Section 2.2).

2.4 Content Payload Definition

This section describes the specification for the payloads for the evaluation request, evaluation response, and warning (if provided). SOAP exceptions may also contain messages using the warning payload definition.

The payload for the request is the clinical and contextual data that is evaluated by the DSS. The payload for the response is the guidance, such as for clinical interventions, provided by the DSS. The data model used in request and in response is the vMR Implementation Guide Release 1 Veersion 2.0. This is further constrained by the vMR templates defined in the vMR Templates Release 1 (**Table 5**).

2.4.1 SEMANTIC SIGNIFIER

The evaluation request **SHALL** use the following semantic signifier for the information model. This semantic signifier corresponds to the CDSInput element defined in the cdsinput.xsd XML schema defined in the HL7 vMR XML Implementation Guide Release 1 Version 2.0 (this and other schemas referenced in this guide are provided as supplemental files in this specification).

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId=" cdsinput:r2:CDSInput" version="2.0"/>
```

The evaluation response **SHALL** use one of the following semantic signifiers for the information model.

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId=" cdsoutput:r2:CDSOutputAsVMR"
version="2.0"/>

<informationModelSSId scopingEntityId="org.hl7.cds" businessId=" cdsoutput:r2:CDSOutputAsDataType"
version="2.0"/>

<informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsoutput:r2:CDSOutputAsStringNameValuePairs" version="2.0"/>

<informationModelSSId scopingEntityId="org.hl7.cds" businessId="kaoutput:r1:CDSActionGroupResponse"
version="2.0"/>
```

The CDSOutput element is defined in the cdsoutput.xsd XML schema included in the HL7 vMR XML Implementation Guide Release 1 Version 2.0. The CDSOutput is abstract, and the vMR XML Implementation Guide defines three concrete implementations (CDSOutputAsVMR, CDSOutputAsDataType, and CDSOutputAsStringNameValuePairs). This implementation guide specifies an additional concrete implementation based on the Action Groups used in the HL7 CDS Knowledge Artifact Implementation Guide (CDSActionGroupResponse). KMs conformant to this implementation guide **SHALL** use one of these four concrete CDSOutput implementations.

For CDS inputs, specifications on what to include, if provided by a DSS, **SHALL** use the CDSInputSpecification information model defined in the HL7 vMR XML Implementation Guide Release 1 Version 2.0. The semantic signifier denoting this information model **SHALL** be:

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsinputspecification:r2:CDSInputSpecification" version="2.0"/>
```

For CDS outputs, specifications on what to include, if provided by a DSS, **SHALL** use the appropriate CDSOutputSpecification information model defined in the HL7 vMR XML Implementation Guide Release 1 Version 2.0.

For CDSOutputAsVMR and CDSOutputAsActionGroup, if providing CDS output specifications, CDSOutputAsVMRSpecification **SHALL** be used.

For CDSOutputAsDataType, if providing CDS output specifications, CDSOutputAsDataTypeSpecification **SHALL** be used.

For CDSOutputAsStringNameValuePairs, if providing CDS output specifications, CDSOutputAsStringNameValuePairSpecification **SHALL** be used.

The semantic signifier denoting these information models **SHALL** be:

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsoutputspecification:r2:CDSOutputAsVMRSpecification" version="2.0"/>
```

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsoutputspecification:r2:CDSOutputAsDataTypeSpecification" version="2.0"/>
```

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId="
cdsoutputspecification:r2:CDSOutputAsStringNameValuePairSpecification" version="2.0"/>
```

2.4.2 CDSINPUT

2.4.2.1 SAMPLE CDS INPUT

In the sample below, the CDS Input is shown unencoded. However, it will be encoded within the service request in base 64. The content to be provided in a CDS Input is defined by a CDS Input Specification (see Section 2.6).

Also of note, for the sake of simplicity, namespaces are omitted from the following examples.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CDSInput>
  <vmrInput>
    <patient>
      <birthTime value="19630525"/>
      <gender codeSystem="2.16.840.1.113883.1.11.1" codeSystemName="HL7" code="M"/>
      <clinicalStatement xsi:type="Problem">
        <templateId root="2.16.840.1.113883.3.1829.11.7.2.4" identifierName="ActiveProblemListEntryCodeOnly"/>
        <problemCode codeSystem="OID for ICD9CM" codeSystemName="ICD9CM" code="250.00">
          <displayName value="Diabetes mellitus"/>
        </problemCode>
      </clinicalStatement>
    </patient>
  </vmrInput>
</CDSInput>
```

Figure 5. Sample CDS Input

2.4.3 CDS OUTPUT

There are four potential outputs that may be used: CDSOutputAsVMR, CDSOutputAsDataType, CDSOutputAsStringNameValuePairs, and CDSOutputAsActionGroup. This section describes these allowed outputs.

2.4.3.1 CDS OUTPUT AS VMR

As the name suggest, this output uses the vMR as the output. Please refer to the HL7 vMR XML Implementation Guide for details (Section 1.4.2). An example is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CDSOutputAsVMR>
  <vmrOutput>
    <patient>
      <clinicalStatement xsi:type="ImagingProposal">
        <id root="9a6d1bac-17d3-4195-89c4-1121bc809b5a"/>
        <procedureCode code="168731009" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT">
          <displayName value="Chest X-Ray"/>
        </procedureCode>
        <proposedProcedureTime><low value="20120826"/><high value="20120826"/></proposedProcedureTime>
      </clinicalStatement>
    </patient>
  </vmrOutput>
</CDSOutputAsVMR>
```

Figure 6. Sample CDSOutputAsVMR

2.4.3.2 CDS OUTPUT AS DATA TYPE

As the name suggest, this output uses a data type as defined in the vMR as the output. Please refer to the HL7 vMR XML Implementation Guide for details (Section 1.4.2). An example is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CDSOutputAsDataType>
  <cdsoutput:value><value xsi:type="BL" value="false"/></cdsoutput:value>
</CDSOutputAsDataType>
```

Figure 7. Sample CDSOutputAsDataType

2.4.3.3 CDS OUTPUT AS STRING NAME VALUE PAIRS

As the name suggest, this output uses string name-value pairs as defined in the vMR as the output. Please refer to the HL7 vMR XML Implementation Guide for details (Section 1.4.2). An example is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CDSOutputAsStringNameValuePairs>
  <stringNameValuePair>
    <name value="Denominator criteria met"/>
    <value value="true"/>
  </stringNameValuePair>
  <stringNameValuePair>
    <name value="Numerator criteria met"/>
    <value value="true"/>
  </stringNameValuePair>
</CDSOutputAsStringNameValuePairs>
```

Figure 8. Sample CDSOutputAsStringNameValuePairs

2.4.3.4 CDS OUTPUT AS ACTION GROUP

This output uses the notion of an Action Group defined in the HL7 CDS Knowledge Artifact Implementation Guide, Release 1 (Section 1.4.2). Further details on this output is provided below.

The response payload for this type of output **SHALL** be CDSOutputAsActionGroup. This type, whose schema is included in the supplemental files, contains these elements:

Component	Description
PatientId	The identifier of the patient for whom this guidance is provided
ActionGroup	Guidance provided by the server

The patient identifier, if specified, **SHALL** be the same as that specified in the vMR element of the request. If the request element did not include a patient identifier, then the response element also will not include a patient identifier. The patient identifier **MAY** be any identifier used by the requesting organization. The identifier also may be a temporary identifier created for the purpose of the CDS transaction.

The ActionGroup element **SHALL** be of the type ActionGroup that is specified in the HL7 CDS Knowledge Artifact Implementation Guide, Release 1. Further constraints on the use of ActionGroup are specified below.

2.4.3.5 CONSTRAINTS ON HED ACTIONS WHEN USED WITHIN CDS OUTPUT AS ACTION GROUP

These constraints are specified largely because of the requirement that the CDS requestor (client of the DSS) should not have to evaluate expressions from the knowledge artifact schema except literal expressions. Thus, in that sense the actions specified in the CDS response payload are simpler. The constraints below are grouped by the base data type (of the knowledge artifact) in which they occur.

ActionBase

Constraint AB-1: No conditions

The conditions element of ActionBase **SHALL NOT** be used. Since ActionBase is the ancestor of all actions and groups, this constraint implies that actions and groups cannot specify conditions.

ActionGroup

Constraint AG-1: No group references

The subElements element of ActionGroup **SHALL NOT** allow any actionGroupReference elements within it.

Constraint AG-2: At least one simple action

The subElements of ActionGroup or its contained groups **SHOULD** contain at least one simple action. That is, a CDS response **SHOULD** always contain one action even if there is no specific guidance being returned. The action element **MAY** indicate that no specific action is needed at the current time.

AtomicAction

Constraint AA-1: No DeclareResponseAction

Actions of type DeclareResponseAction **SHALL NOT** be allowed.

Constraint AA-3: No FireEventAction

Actions of type FireEventAction **SHALL NOT** be allowed.

CollectInformationAction

Constraint CIA-1: No ResponseBinding

The element responseBinding **SHALL NOT** be used in an element of type CollectionInformationAction.

CreateAction

Constraint CA-1: Only ComplexLiteral actionSentence

The actionSentence element in CreateAction **SHALL** only allow expressions of type ComplexLiteral.

UpdateAction

Constraint UA-1: Only ObjectDescriptor expression for source

The element source of the element actionSentence (which is of type ObjectRedefine) **SHALL** be of type ObjectDescriptor.

Constraint UA-2: Only literal properties

The element value of the element property of the element actionSentence **SHALL** only allow expressions of literal types.

RemoveAction

Constraint RA-1: Only ObjectDescriptor expression for actionSentence

The element actionSentence **SHALL** be of type ObjectDescriptor.

When the response needs to reference an existing data object (in UpdateAction and RemoveAction) known to the CDS requestor, it **SHALL** use an ObjectDescriptor expression. This expression specifies an identifier of the object. In order to determine the identifier property or properties of a given type, the service references a configuration file. This information is provided in the vMRDefinitions.xml document included in the supplemental files in Example Implementation\HeD.DSS\Models.

In the sample below, the payload is shown unencoded. However, it will be encoded in base 64 format within the service response.

```
<cdsOutput xsi:type=”CDSActionGroupResponse”>
        <patientId identifierName="mrn" root="999112AR" />
        <ka:actionGroup>
                <ka:actionId identifierName="CDSSGenId"
                        root="9547370d-b0b0-4820-b3ce-5eadec7d37d4" />
                <ka:behaviors>
                        <ka:behavior xsi:type="ka:GroupSelectionBehavior" value="AtMostOne" />
                </ka:behaviors>
                <ka:title value="CDS Response Action Examples" />
                <ka:subElements>
                        <ka:simpleAction xsi:type="ka:CreateAction">
                                <!-- Create a proposed procedure to be presented to the physician at
                                        CPOE time -->
                                <ka:textEquivalent value="Ambulate" />
                                <ka:actionSentence xsi:type="ka:ComplexLiteral">
                                        <ka:value xsi:type="vmr:ProcedureProposal">
                                                <vmr:templateId root="2.16.840.1.113883.3.1829.11.8.3.2"
                                                        identifierName="ProcedureProposal"/>
                                                <vmr:procedureCode codeSystem="2.16.840.1.113883.6.96"
                                                        code="62013009" codeSystemName="SNOMED-
CT"><dt:displayName value="Ambulating Patient"/></vmr:procedureCode>
                                        </ka:value>
                                </ka:actionSentence>
                        </ka:simpleAction>
                        <ka:simpleAction xsi:type="ka:CreateAction">
                                <ka:textEquivalent value="Bed rest" />
                                <ka:actionSentence xsi:type="ka:ComplexLiteral">
                                        <ka:value xsi:type="vmr:ProcedureProposal">
                                                <vmr:templateId root="2.16.840.1.113883.3.1829.11.8.3.2"
                                                        identifierName="ProcedureProposal"/>
                                                <vmr:procedureCode codeSystem="2.16.840.1.113883.6.96"
                                                        code="183074009" codeSystemName="SNOMED-CT"
<dt:displayName value="Recommendation to rest in bed"/></vmr:procedureCode>
                                        </ka:value>
                                </ka:actionSentence>
                        </ka:simpleAction>

                </ka:subElements>
        </ka:actionGroup>
</cdsOutput>
```

Figure 9. Sample CDSActionGroupResponse

2.4.4 EXECUTION MESSAGE CONTAINER

This section describes the payload structure for warning, informational, and exception messages from the server to the requestor. An information message is returned when the DSS execution proceeded as expected and it successfully evaluated the patient data. A warning is issued by the DSS if it encountered unexpected issues but was able to evaluate the patient data against the specified knowledge module. In comparison, the DSS throws an exception when it is unable to successfully initiate or complete the evaluation of patient data against the specified knowledge module. Within the SOAP implementation, a message can be provided within the exception using this message container. The same schema models these messages with varying severity. The semantic signifier for the information model for the warning and exception message payload **SHALL** be:

```
<informationModelSSId scopingEntityId="org.hl7.cds" businessId="kaoutput:r1:CDSExecutionMessage"
version="1.0"/>
```

The XML schema for this model, called CDSExecutionMessage.xsd, is provided as a supplemental file to this specification.

The message payload **SHALL** be of type CDSExecutionMessage. This type, an example of which is provided in Section 2.4.4.1, contains the following elements:

Component	Description
Id	A DSS generated identifier for the message.
Reason	A code indicating the specific reason for the message.
Level	A value indicating the relative severity of the message.
Message	Descriptive text explaining why the message was issued and potential ways to resolve any issues.
SourceComponentType	The type of the component within the DSS "ecosystem" that caused the message to be issued. A value for this element must be provided if a value for the sourceComponentId is provided.
SourceComponentId	The specific source component instance that caused the message. This can be a knowledge module id, a clinical statement id for patient data, a URL for an external resource.

The level of the message **SHALL** be from one of the following values:

Item	Description
Information	The message may provide helpful information to the requestor when the evaluation proceeded as expected. The message is issued as payload for a DSS Response.
Warning	The message is issued when issues were encountered but did not prevent the evaluation of the data. This can occur when the DSS can correct a problem, for example. The message is issued as payload for a DSS Response.
Exception	This message is issued when problems were encountered that prevented a successful evaluation. Typically, these problems can be corrected by the requestor for future requests. The message is issued as payload for a DSS Exception.
Fatal	This message is issued when there were severe problems within the DSS that prevent the evaluation. The problem must be corrected by the DSS administrator. The message is issued as payload for a DSS Exception.

The SourceComponentType **SHALL** be one of the following items:

Item	Description
Data	The warning is related to the data being sent as part of the request.
Connectivity	The warning is related to connectivity with one or more components within the system.
KnowledgeBase	The warning is related to an inappropriate use of a knowledge base.
Security	The warning is related to authentication or authorization of the request.
System	The warning is an internal warning from the CDS engine.

The Reason **SHALL** be selected from one of the following items:

Item	Description
DataFormatNotRecognized	The format in which the patient data was sent was not recognized. This may include non-conformance to a vMR template.
DataValueIsOutOfRange	Value for data item is out of expected range.
DataIsMissing	Data item or value for a data item property is missing.
KnowledgeModuleIsNotCurrent	A knowledge module is not current and has been superceded by a new version.
KnowledgeModuleIsNotApplicable	The requested knowledge module is not applicable to the given context - i.e., patient, care location, encounter, user, or recipient.
ResourceIssuedMessage	A message was issued by another resource (e.g., terminology server) that is being passed through by the DSS.
AuthenticationCredentialsNearingExpiry	The credentials used for establishing connectivity

will expire in the near future.

2.4.4.1 SAMPLE WARNING PAYLOAD

In the sample below, the payload is shown unencoded. However, it will be encoded in base 64 within the service response.

```
<cdsExecutionMessage>
        <!-- This warning shows that the problem was coded in a code system that does not conform to requirements  -->
        <id root="54ff5a98-b61f-4ef8-871f-2a2bb2feeb4d"/>
        <reason value="DataFormatNotRecognized"/>
        <level value="Warning"/>
        <message value="Invalid codeSystem ICPC used for problemCode in Problem"/>
        <sourceComponentType value="Data"/>
        <sourceComponentId objectType="vmr:Problem" >
                <ka:property name="id">
                        <ka:value xsi:type="ka:ComplexLiteral">
                                <ka:value xsi:type="dt:II"
                                        root="8a625979-500d-4c49-b5a4-a39cc72061bd"/>
                        </ka:value>
                </ka:property>
</cdsExecutionMessage>
```

Figure 10. Sample CDSExecutionMessage

2.5 Clinical Data Representation

DSS implementations compliant with this implementation guide **SHALL** use the vMR as defined in the HL7 vMR XML Implementation Guide Release 1 Version 2.0 for clinical data representation. Moreover, compliant DSS implementations **SHALL** use HL7 vMR templates as defined in the HL7 vMR Templates Release 1 Version 1.0 specification when an appropriate template is available. If a vMR template is used, extraneous data elements or attributes that are outside the scope of the template **SHALL** be ignored by the DSS provider. For example, consider the ActiveProblemListEntryCodeOnly template. This template requires that only active problem list entries be provided, and that only the problemCode is populated for each problem. If the DSS client uses this template and sends in additional information (e.g., problemStatus or problemEffectiveTime), such additional data **SHALL** be ignored by the DSS provider.

2.6 Specification of Knowledge Module Meta-Data

A DSS provider will need to make certain meta-data available to its clients. At a minimum, DSS implementations compliant with this implementation guide **SHALL** make the following meta-data available to its clients with regard to its knowledge modules (**KM**):

- The KM identifier (including version) and status (note that the DSS specification provides detailed guidance on conventions for these items that **SHALL** or **SHOULD** be followed)
- The KM traits specified in the HL7 DSS standard, Release 2 (Section 1.4.2)
  - Purpose
  - Explanation
  - StewardOrganization
  - CreationDate
  - LastReviewDate
  - AuthorList
  - FreeTextKeywordList
  - CodedValueKeywordList
- The data requirements for the KM
- How KM evaluation results will be returned

This implementation guide does NOT require that a specific approach be used to make these meta-data available to clients. For example, this information could be provided via documentation, a Web page, or the use of DSS operations for retrieving meta-data. Examples are provided below for illustration purposes. Note that in some cases, more information than may be necessary for a document-based delivery of these meta-data is provided, as the DSS operations require these data elements.

2.6.1 EXAMPLE SPECIFICATION OF KNOWLEDGE MODULE META-DATA: PNEUMOCOCCAL VACCINATION FOR INDIVIDUAL WITH DIABETES MELLITUS

KM Identifier	Element	Example Value
	Scoping Entity ID	com.cdsvendor
	Business ID	DiabetesMellitusNeedForPneumococcalVaccination
	Version	1.0.0
Status	PROMOTED	
Purpose	The purpose of this knowledge module is to identify whether a patient has diabetes mellitus and is in need of a pneumococcal vaccination.	
Explanation	The recommendations are based on the American Diabetes Association's 2013 Standard of Medical Care for Diabetes Mellitus (http://care.diabetesjournals.org/content/36/Supplement_1/S11.full). A pneumococcal vaccine is recommended once for all patients with diabetes mellitus age greater than or equal to 2 years. A repeat vaccine is recommended for individual greater than or equal to 65 years, as long as the previous vaccine was administered greater than or equal to 5 years ago when the patient was less than 65 years of age. The guideline defines other conditions for repeat vaccination, including "nephrotic syndrome, chronic renal disease, and other immunocompromised states, such as after transplantation." This knowledge module does not implement these additional conditions for repeat vaccination.	
StewardOrganization	CDS Vendor X	
CreationDate	February 1, 2013	
LastReviewDate	August 1, 2013	
AuthorList	Kensaku Kawamoto, MD, PhD David Shields	
FreeTextKeywordList	Diabetes, pneumococcal vaccination, pneumococcal immunization	

CodedValueKeywordList	SNOMED CT 73211009 (diabetes mellitus), SNOMED CT 333598008 (pneumococcal vaccine)
Data Requirement Group	

Element	Example Value
Name	Default data requirement group
Description	The default data requirement group. Data requirement groups are organizational grouping of data requirement items that can be used to designate data required initially for iterative DSS interactions, which are not currently utilized in this implementation guide.
Scoping Entity ID	com.cdsvendor
Business ID	DSSRequirements
Version	1.0
Item ID	DefaultDataRequirementGroup

Data Requirement Item Identifying Information	

Element	Example Value
Name	Sole data requirement item for diabetes pneumococcal vaccine KM
Description	The sole data requirement item required for the diabetes pneumococcal vaccine KM.
Scoping Entity ID	com.cdsvendor
Business ID	DiabetesMellitusNeedForPneumococcalVaccination
Version	1.0.0
Item ID	SoleDataRequirementItemForKM

Data Requirement Item	

Information Model

Element	Example Value
Scoping Entity ID	org.hl7.cds
Business ID	cdsinput:r2:CDSInput
Version	2.0

Input Specification Model (Query Model)

Element	Required Value
Scoping Entity ID	org.hl7.cds
Business ID	cdsinputspecification:r2:CDSInputSpecification
Version	2.0

Input Specification Contents (Query Contents) for Patients

Requirement for Birth Date

Element	Example Value
Required Evaluated Person Template	PatientDOBAndGenderOnly OID: 2.16.840.1.113883.3.1829.11.2.1.2

Requirement for Simple Problem List

Element	Example Value
Required General Clinical Statement Class	Problem
Required Clinical Statement Template ID	ActiveProblemListEntryCodeOnly OID: 2.16.840.1.113883.3.1829.11.7.2.4

	Requirement for Pneumococcal Vaccine Administration Events

Element	Example Value
Required Clinical Statement Template ID	SimpleImmunizationHistoryListEntry CodeAndDateOnly 2.16.840.1.113883.3.1829.11.9.1.11
Target Coded Attribute	substance.substanceCode
Target Codes	CVX 133, 100, 152, 33, 109

Evaluation result identifying information	

Element	Example Value
Name	Sole Evaluation Result for Diabetes Pneumococcal Vaccination KM
Description	The sole evaluation result for a diabetes pneumococcal vaccination KM. It will be simple a BL data type that returns true if the patient requires a pneumococcal vaccination, and false if not.
Scoping Entity ID	com.cdsvendor
Business ID	DiabetesMellitusNeedForPneumococcalVaccination
Version	1.0.0
Item ID	Sole Evaluation Result

Evaluation result data model	

Element	Example Value
Scoping Entity	org.hl7.cds
Business Name	cdsoutput:r2:CDSOutputAsDataType
Version	2.0

2.7 Example Implementation

To provide a concrete illustration of the implementation of a DSS using the standards and practices recommended in this implementation guide, an example service and client implementation is provided. The example is written using the Microsoft .NET Framework, and is based on a REST implementation using ASP.NET MVC4 as the web scaffolding.

The example is available in the supplemental folder of the publication materials. The example also makes use of some of the definitional assemblies from the HeD Schema Framework, released as part of the HL7 CDS Knowledge Artifact Implementation Guide.

The example is built as a Microsoft Visual Studio solution file containing the following projects:

- HeD.Model – Contains class definitions generated from the HeD Action Schema.
- HeD.DSS.Models – Contains class definitions generated from the OmgDssSchema.xsd, published as part of HL7 DSS Release 2.
- HeD.CDSS.Model – Contains class definitions generated from the CDS Request and Response schemas defined as part of this implementation guide.
- HeD.DSS – Contains the Decision Support Service implementation.
- HeD.DSS.SampleClient – Contains a sample DSS client implementation.

2.7.1 SAMPLE CLIENT REQUEST

The SampleClient executable is a simple console application that creates an HTTP-POST request for the *evaluate* service method. The payload is a simple vMR patient data package containing a single SubstanceAdministrationEvent. The following code generates the sample request:

```
// Get the body of the request
var vMRRequest = GetSampleRequest();
```

And then packages it in the DSS evaluate construct:

```
// Construct the DSS-level request, base-64 encoding the vMR request within it.
var evaluate =
  new evaluate
  {
    interactionId =
      new InteractionIdentifier
      {
        interactionId = Guid.NewGuid().ToString("N"),
        scopingEntityId = "SAMPLE-CLIENT",
        submissionTime = DateTime.Now.ToUniversalTime()
      },
    evaluationRequest =
      new EvaluationRequest
      {
        clientLanguage = "XXX",
        clientTimeZoneOffset = "XXX",
        dataRequirementItemData = new List<DataRequirementItemData>
        {
          new DataRequirementItemData
          {
            driId = new ItemIdentifier { itemId = "RequiredDataId" },
            data =
              new SemanticPayload
              {
```

```
                        informationModelSSId = SemanticSignifiers.CDSInputId,
                        base64EncodedPayload = Packager.EncodeRequestPayload(vMRRequest)
                    }
                }
            },
            kmEvaluationRequest = new List<KMEvaluationRequest>
            {
                new KMEvaluationRequest { kmId = new EntityIdentifier { scopingEntityId = "org.hl7.cds", businessId =
"NQF-0068", version = "1.0" } }
            }
        }
    };
```

This request is then passed to the service using a stock HTTP request method:

```
// Post the request and retrieve the response
var response = client.PostAsXmlAsync("api/evaluation", evaluate).Result;
```

2.7.2 RECEIVING THE REQUEST

The DSS Web Service application receives the HTTP request discussed in the previous section as a DSS evaluate construct, and after some validity checks, uses the following code in the HeD.DSS.Models.Evaluator class, Evaluate method to decode the payload:

```
var payload = Packager.DecodeRequestPayload(dri.data.base64EncodedPayload);
```

This results in the decoded payload, represented as a CDSInput object. This is the point where an actual CDS engine would be invoked to process the request. For the purposes of the example, a sample response containing a single SubstanceAdministrationProposal is generated:

```
try
{
    // Pass payload to rule engine for evaluation here...
    // NOTE: This could be done a per KM basis as well, depends on whether the engine
    // is aggregating results, or keeping results separate.
    engineResponse = GetSampleEngineResponse(payload);
}
catch (CDSExecutionMessageWrapper m)
{
    // Catch any execution exceptions and convert them to DSS exceptions
    throw new DSSExceptionWrapper(m.ExecutionMessage.ToDSSException());
}
```

Note that the request processing is wrapped with a try..catch statement that will catch all CDSExecutionMessageWrapper exceptions. This class is a simple wrapper that contains the CDSExecutionMessage instance and exposes it as a .NET exception. The wrapper here will catch the CDSExecutionMessage and convert it to a DSSExceptionWrapper so that it can be returned as an exception by the service. Exception management is discussed in a separate section below.

2.7.3 RETURNING THE RESPONSE

Once the sample response is generated, the Evaluate method packages the response in an *evaluateResponse* construct as defined by the DSS specification:

```
// Package the engine's response in a DSS evaluate response.
return
  new evaluateResponse
  {
    requestId = input.interactionId,
    responseId = new InteractionIdentifier { scopingEntityId = "org.hl7.cds", interactionId =
Guid.NewGuid().ToString("N"), submissionTime = DateTime.Now },
    evaluationResponse =
      new EvaluationResponse
      {
        finalKMEvaluationResponse =
          new List<FinalKMEvaluationResponse>
          {
            new FinalKMEvaluationResponse
            {
              kmEvaluationResultData =
                new List<KMEvaluationResultData>
                {
                  new KMEvaluationResultData
                  {
                    evaluationResultId = new ItemIdentifier { itemId = Guid.NewGuid().ToString("N") },
                    data =
                      new SemanticPayload
                      {
                        informationModelSSId = SemanticSignifiers.CDSActionGroupResponseId,
                        base64EncodedPayload =
Packager.EncodeActionGroupResponsePayload((CDSActionGroupResponse)engineResponse.Response)
                      }
                  }
                },
              kmId = new EntityIdentifier { businessId = Guid.NewGuid().ToString("N"), scopingEntityId =
inputScopingEntityId, version = "1" },
              warning =
                (
                  from m in engineResponse.Messages
                  select
                    new Warning
                    {
                      value =
                        new SemanticPayload
                        {
                          informationModelSSId = SemanticSignifiers.CDSExecutionMessageId,
                          base64EncodedPayload = Packager.EncodeExecutionMessagePayload(m)
                        }
                    }
                ).ToList()
            }
          }
      }
  };
}
```

Note that the response and any execution messages are encoded separately. The execution messages in this response are limited to informational and warning messages. Exception level messages are handled differently using the ExceptionWrapper classes as discussed in the Exception Handling section below.

The evaluateResponse instance is returned as the result of the EvaluationController Evaluate method, and the ASP.NET MVC plumbing manages serialization of that message into an HTTP response.

2.7.4 PROCESSING THE RESPONSE

Once the service returns the response, the sample client decodes the result using the following code:

```
var actionGroupResponse = Packager.DecodeActionGroupResponsePayload(evaluateResult.data.base64EncodedPayload);
```

This results in an action group response, in this case containing the single SubstanceAdministrationProposal produced by the sample service engine implementation. The processing logic assumes that if the response contains a CreateAction, the action sentence will be a ComplexLiteral. The constraints on the Action schema ensure that the actionSentence element of the CreateAction will contain a ComplexLiteral, so this assumption is valid. In this case, the ComplexLiteral contains a SubstanceAdministrationProposal, and the sample prints out the contents of the proposal.

2.7.5 EXCEPTION HANDLING

The above sections discussed the normal processing route through the service, with a request-response interaction being handling normally. For exceptional cases, this guide defines two separate levels of handling. The first level is based on the exception classes defined within the DSS specification itself, and is meant for dealing with exceptions at the DSS level. The second level is based on the exception classes defined within this document, and is intended to deal with exceptions occurring at the semantic payload and engine processing levels. These exception classes are then mapped to the DSS Exception level as either EvaluationException instances, or DSSRuntimeException instances.

The CDSExecutionMessage type represents execution messages that may be raised by a CDS engine. The Level element of this type indicates whether the message should be returned as part of normal processing (as a Warning in the DSS response), or as an exception.
If the message is handled as an exception, the CDSExecutionMessageWrapper is used to wrap the CDSExecuteMessage and handle it as an Exception in the .NET runtime.
The sample Evaluator implementation traps for exceptions of this type, and converts them to the DSSExceptionWrapper, which is used to marshal exception information between the service and client at the DSS level.

The following code illustrates converting a CDSExecutionMessageWrapper to a DSSExceptionWrapper:

```
try
{
   // Call engine to process request
}
catch (CDSExecutionMessageWrapper m)
{
   // Catch any execution exceptions and convert them to DSS exceptions
   throw new DSSExceptionWrapper(m.ExecutionMessage.ToDSSException());
}
```

In the EvaluationController, DSSExceptionWrapper exceptions are caught and converted to HttpResponseExceptions, with the body of the response set to the Xml representation of the DSSException instance:

```
catch (DSSExceptionWrapper de)
{
   throw new HttpResponseException(de.Exception.ToHttpResponseMessage());
}
```

On the client side, if the response does not indicate success, the following code is used to print the exception message:

```
if (!response.IsSuccessStatusCode)
{
   Console.WriteLine("{0} ({1})", (int)response.StatusCode, response.ReasonPhrase);

   var content = response.Content.ReadAsStringAsync().Result;
   // NOTE: Deserialization here is assuming EvaluationException, need to peek into the Xml stream to determine the
actual type.
   var dssException = DSSExceptionExtensions.DeserializeFromString<EvaluationException>(content);
   Console.WriteLine(dssException.GetType().Name);
   Console.WriteLine(String.Join("\r\n", dssException.errorMessage));
   if (dssException.value != null)
   {
      var cdsMessage = Packager.DecodeExecutionMessagePayload(dssException.value.base64EncodedPayload);
      Console.WriteLine(cdsMessage.GetType().Name);
      Console.WriteLine(cdsMessage.message.value);
   }
}
```

As noted in the comments, the code here is assuming an EvaluationException. A production implementation would need to peek into the Xml stream to determine the actual type of the exception to deserialize.

This code also checks to see whether the exception has a payload, and if it does, the Packager is used to decode the CDSExecutionMessage contents.

3.0 APPENDICES

Appendix A: Acronyms

The following acronyms are referenced in this implementation guide:

Acronym	Definition/Description
C-CDA	Consolidated Clinical Document Architecture
CDS	Clinical Decision Support
CPT	Current Procedural Terminology
CVX	Codes for Vaccine Administered
DAM	Domain Analysis Model
DRI	Data Requirement Item
DSS	Decision Support Service
DSTU	Draft Standard for Trial Use
ECA	Event Condition Action
EHR	Electronic Health Record
EMR	Electronic Medical Record
HeD	Health e-Decisions
HIT	Health Information Technology
HITSP C80	Health Information Technology Standards Panel Clinical Document and Message Terminology Component C80
HL7	Health Level 7
HTTP	Hypertext Transfer Protocol
ICD-10	International Classification of Diseases, Ninth Revision
ICD-9	International Classification of Diseases, Tenth Revision
IHE	Integrating the Healthcare Enterprise
KM	Knowledge Module

LOINC	Logical Observation Identifiers Names and Codes
MeSH	Medical Subject Headings
NDF-RT	National Drug File – Reference Terminology
ONC	Office of the National Coordinator
PHR	Personal Health Record
QDM	Quality Data Model
QRDA	Quality Reporting Document Architecture
REST	Representational State Transfer
SNOMED-CT	Systematized Nomenclature of Medicine – Clinical Terms
SOAP	Simple Object Access Protocol
SS	Semantic Signifier
UML	Unified Modeling Language
vMR	Virtual Medical Record
WSDL	Web Services
XML	Extensible Markup Language

Appendix B: Glossary of Terms

Key terms used in this implementation guide are listed in the table below:

Reference	Description and/or Link
CDS Artifact Sharing	Standards to structure medical knowledge in a shareable and executable format for use in CDS
CDS Guidance Requestor	The user of the CDS Guidance System
CDS Guidance Service	Standards that define how a system can interact with and utilize an electronic service that provides helpful, actionable clinical guidance
CDS Guidance Supplier	The CDS Service Vendor
CDS Knowledge Artifact	Medical knowledge represented in a structured and encoded form to enable computer based clinical decision support.
CDS Service Vendor	Also known as the CDS Guidance Supplier
CDS System	This term is used to describe any system that provides clinical decision support. Often, though not exclusively, the CDS is provided by an EHR system.
Clinical Decision Support (CDS)	Clinical Decision Support is a process for enhancing health-related decisions and actions with pertinent, organized clinical knowledge and patient information to improve health and healthcare delivery. Information recipients can include patients, clinicians and others involved in patient care delivery; information delivered can include general clinical knowledge and guidance, intelligently processed patient data, or a mixture of both; and information delivery formats can be drawn from a rich palette of options that includes data and order entry facilitators, filtered data displays, reference information, alerts and others. (Jerome A. Osheroff, Jonathan M. Teich, Donald Levick, Luis Saldana, Ferdinand T. Velasco, & Dean F. Sittig, 2012)
Document Templates	A structured form for recording information on a patient into a set of pre-defined data slots.

Event Condition Action (ECA) Rule	An ECA rule has the general syntax "on event if condition is true do action." • The event part specifies the signal that triggers the invocation of the rule • The condition part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out • The action part consists of execution of operations. These actions may in turn cause further events to occur, which may in turn cause more ECA rules to fire (Wikipedia), (Pissinou, et al., 1994), (Papamarkos, Poulovassilis, & Wood, 2006)
Import	To incorporate a CDS Knowledge Artifact into a CDS System.
Mapping	Mapping of data elements and actions within a CDS Knowledge Artifact to local data elements and actions within the CDS System.
Metadata	Information about the design and specification of data structures
Obtain	To acquire a CDS Knowledge Artifact
Order Set	A pre-defined and approved group of orders related to a particular clinical condition (e.g., hypertension treatment and monitoring) or stage of care (e.g., hospital admission to Coronary Care Unit). Often the order set consists of both diagnostic and therapeutic orders. The goals in creating order sets are to standardize care, increase compliance with best clinical practices, and facilitate the order entry process.
Payload Standards	Standards regarding the cargo of a data transmission
Standards & Interoperability (S&I) Framework	The S&I Framework is a collaborative community of participants from the public and private sectors who are focused on providing the tools, services and guidance to facilitate the functional exchange of health information. The S&I Framework uses a set of integrated functions, processes, and tools that enable execution of specific value-creating initiatives. Each S&I Initiative tackles a critical

	interoperability challenge through a rigorous process that typically includes: • Development of clinically-oriented user stories and robust use cases • Harmonization of interoperability specifications and implementation guidance • Provision of real-world experience and implementer support through new initiatives, workgroups and pilot projects • Mechanisms for feedback and testing of implementations, often in conjunction with ONC partners such as NIST