



*Unified Medical Language System  
Knowledge Source Server (UMLS)*

**Developer's Guide**

**Version 4.2**

**July, 2004**

Revision 1.0



# Table of Contents

About This Guide	viii
Audience	viii
Release Notes	viii
How to Use This Guide	viii
If You Need Help...	viii
1 Introduction	1-1
1.1 Requirements Summary	1-1
1.1.1 Key Requirements and Design Goals	1-1
1.2 External Interfaces	1-1
1.3 Performance, Sizing and Loading	1-2
1.4 Logical Architecture	1-2
1.5 Physical Architecture	1-2
2 Installing the UMLSKS	2-1
3 Building UMLSKS Software Applications	3-1
3.1 Building and Running Your Program	3-1
3.1.1 Building the Example . java Files	3-2
3.1.2 Running the Client	3-2
3.1.3 Running the ExpertClient	3-3
3.1.4 Running the SocketClient	3-3
3.1.5 Running the StandardQueryClient	3-4
3.2 API Package Structure	3-4
3.2.1 Package gov.nih.nlm.kss.api	3-5
3.2.2 Package gov.nih.nlm.kss.util	3-6
3.3 Program Initialization	3-6
3.4 UMLSKS API Functions	3-7
3.5 Using the UMLSKS Object Model	3-23
3.5.1 Properties Required by a UMLSKS Client	3-23
3.5.2 Package gov.nih.nlm.kss.models	3-24
3.5.3 Package gov.nih.nlm.kss.models.meta.concept	3-25
3.5.4 Package gov.nih.nlm.kss.models.meta.context	3-26
3.5.5 Package gov.nih.nlm.kss.models.meta.assocExp	3-27
3.5.6 Package gov.nih.nlm.kss.models.meta.attribute	3-27
3.5.7 Package gov.nih.nlm.kss.models.meta.cooccurrence	3-28
3.5.8 Package gov.nih.nlm.kss.models.meta.locator	3-29
3.5.9 Package gov.nih.nlm.kss.models.meta.meshentry	3-29
3.5.10 Package gov.nih.nlm.kss.models.meta.relation	3-29
3.5.11 Package gov.nih.nlm.kss.models.meta.source	3-30
3.5.12 Package gov.nih.nlm.kss.models.meta.deltas	3-30
3.5.13 Package gov.nih.nlm.kss.models.meta	3-30
3.5.14 Package gov.nih.nlm.kss.models.sem.units	3-31
3.5.15 Package gov.nih.nlm.kss.models.sem.rels	3-32
3.5.16 Package gov.nih.nlm.kss.models.lex	3-32

4	Using the XML Query Facility	4-1
4.1	Retrieving Metathesaurus and Semantic Network Data using an XML Query	4-1
4.2	Structure of an XML Query	4-1
4.3	Metathesaurus and Semantic Network Table Contents	4-2
4.3.1	Table 'mrcon'	4-2
4.3.2	Table 'mrso'	4-3
4.3.3	Table 'mrdef'	4-3
4.3.4	Table 'mrsty'	4-3
4.3.5	Table 'mrrel'	4-3
4.3.6	Table 'mrmtx'	4-4
4.3.7	Table 'mrcxt'	4-4
4.3.8	Table 'mrcoc'	4-5
4.3.9	Table 'mrsat'	4-6
4.3.10	Table 'mrlo'	4-7
4.3.11	Table 'mrxts_eng'	4-7
4.3.12	Table 'mrxtw_eng'	4-8
4.3.13	Table 'mrxtw_eng'	4-8
4.3.14	Table 'mrcui'	4-8
4.3.15	Table 'mrsab'	4-9
4.3.16	Table 'mshqual'	4-9
4.3.17	Table 'strattr'	4-9
4.3.18	Table 'ttys'	4-10
4.3.19	Table 'cots'	4-10
4.3.20	Table 'stts'	4-10
4.3.21	Table 'rels'	4-10
4.3.22	Table 'srdef'	4-10
4.3.23	Table 'srgrp'	4-11
4.3.24	Table 'srstr'	4-11
4.3.25	Table 'srstre1'	4-11
4.3.26	Table 'srstre2'	4-11
4.3.27	Table 'mrconso'	4-12
4.3.28	Table 'mrdoc'	4-12
4.3.29	Table 'mrhier'	4-12
4.3.30	Table 'mrhist'	4-13
4.3.31	Table 'mrmap'	4-13
4.4	XML Query Example	4-13
5	Using the UMLS/SKS Socket Server	5-1
5.1	Connecting to the UMLS/SKS Socket Server	5-1
5.2	General Queries	5-1
5.2.1	XML Query getCurrentUMLSVersion	5-1
5.2.2	XML Query getUMLSVersions	5-1
5.2.3	XML Query getSWVersion	5-1
5.2.4	XML Query query	5-1
5.2.5	XML Query describeCurrentUMLSVersion	5-1
5.2.6	XML Query describeUMLSVersions	5-1
5.2.7	XML Query listDocEntryTypes	5-2

5.3	Metathesaurus Queries	5-2
5.3.1	XML Query listDictionaries	5-2
5.3.2	XML Query suggestSpelling	5-2
5.3.3	XML Query getConceptName	5-2
5.3.4	XML Query findCUI	5-3
5.3.5	XML Query getConceptProperties	5-4
5.3.6	XML Query findConcept	5-4
5.3.7	XML Query getBasicConceptProperties	5-5
5.3.8	XML Query findBasicConcept	5-6
5.3.9	XML Query getTerminology	5-7
5.3.10	XML Query getTerms	5-7
5.3.11	XML Query getDefinition	5-8
5.3.12	XML Query getSemanticType	5-9
5.3.13	XML Query getContext	5-9
5.3.14	XML Query getAssocExprs	5-9
5.3.15	XML Query getCooccurrences	5-10
5.3.16	XML Query getRelations	5-10
5.3.17	XML Query getStringAttributes	5-11
5.3.18	XML Query getLocator –or- getLocators	5-12
5.3.19	XML Query getMeSHEntries	5-12
5.3.20	XML Query getMeSHInfo	5-13
5.3.21	XML Query getRawRecords	5-13
5.3.22	XML Query describeSource	5-13
5.3.23	XML Query listSources	5-14
5.3.24	XML Query findLUI	5-14
5.3.25	XML Query getTermsForLUI	5-15
5.3.26	XML Query findSUI	5-15
5.3.27	XML Query getStringForSUI	5-16
5.3.28	XML Query listStrAttrs	5-16
5.3.29	XML Query listMeSHQuals	5-16
5.3.30	XML Query describeUMLSChanges	5-17
5.3.31	XML Query listTermTypes	5-17
5.3.32	XML Query listCooccurrenceTypes	5-18
5.3.33	XML Query listStringTypes	5-18
5.3.34	XML Query listRelationTypes	5-18
5.3.35	XML Query listMetaTableNames	5-19
5.3.36	XML Query listRelationTypes	5-19
5.4	Semantic Network Queries	5-19
5.4.1	XML Query findSemType	5-19
5.4.2	XML Query getSemTypeProperties	5-20
5.4.3	XML Query getSemTypeAncestors	5-20
5.4.4	XML Query getSemTypeSiblings	5-20
5.4.5	XML Query listSemTypeIds	5-21
5.4.6	XML Query findSemRelation	5-21
5.4.7	XML Query getSemRelationProperties	5-22
5.4.8	XML Query getSemRelationAncestors	5-22
5.4.9	XML Query listSemRelationIds	5-22

5.4.10	XML Query existsAssociativeRelation	5-23
5.4.11	XML Query getAssociativeRelations	5-23
5.4.12	XML Query existsHierRelRelRelation	5-24
5.4.13	XML Query listSemGroups	5-24
5.4.14	XML Query listSemTypes	5-25
5.4.15	XML Query getSemGroup	5-25
5.4.16	XML Query findBasicSemType	5-25
5.4.17	XML Query findBasicSemRelation	5-26
5.4.18	XML Query listSemNetTableNames	5-26
5.5	SPECIALIST Lexicon Queries	5-26
5.5.1	XML Query getLexicalRecords	5-26

## Acronym List

API	Application Programmer Interface
CUI	Concept Unique Identifier
FTP	File Transfer Protocol
HHS	Health and Human Services
HTTP	Hypertext Transfer Protocol
UMLSKS	Knowledge Source Server
NIH	National Institutes of Health
NLM	National Library of Medicine
RAM	Random Access Memory
RMI	Remote Method Invocation
UMLS	Unified Medical Language System
XML	eXtensible Markup Language

## List of Figures

Figure 1 - Logical System Diagram .....	1-2
Figure 2 - Physical System Environment .....	1-3
Figure 3 – Client.java Menu .....	3-3
Figure 4 – UMLSKS API Packages .....	3-5
Figure 5 – UMLSKS Connection Class .....	3-6
Figure 6 – UMLSKS Utility Classes .....	3-6

## Revision History

March, 2004	Initial creation after separation from the User's Guide
July, 2004	Addition of the Release 4.2 API interface classes.

## About This Guide

This guide describes the installation, usage, and programmatic interface for the UMLS Knowledge Source Server (UMLS<sub>KS</sub>).

### *Audience*

The audience for this guide is developers of UMLS<sub>KS</sub> applications using the UMLS<sub>KS</sub> API.

### *Release Notes*

Please refer to the Release Bulletin found in the [umlsks.nlm.nih.gov](http://umlsks.nlm.nih.gov) web site for detailed information about the changes found in each release of the UMLS<sub>KS</sub> software.

### *How to Use This Guide*

This manual contains the following chapters:

- *Chapter 1 - Introduction* describes the basic features and architecture of the UMLS<sub>KS</sub>.
- *Chapter 2 - Installing the UMLS<sub>KS</sub>* provides administrators instructions on installing and tailoring a UMLS<sub>KS</sub> installation.
- *Chapter 3 - Building UMLS<sub>KS</sub> Software Applications* describes the functions available to developers wanting to interface to the UMLS<sub>KS</sub> through another Java program.
- *Chapter 4 - Using the XML Query Facility* describes how to use the querying facility of the UMLS<sub>KS</sub> wherein users build XML queries to be executed.
- *Chapter 5 - Using the UMLS<sub>KS</sub> Socket Server* describes how to use the socket server to pass XML formatted commands or command-line type queries (e.g. `ks -meta -c aids`) that are to be executed by the server, with the results passed back to the client listening to the socket for a response.

### *If You Need Help...*

If you run into problems with the UMLS<sub>KS</sub>, please first consult the User's Guide for detailed instructions on usage and known problems with the system. Should you still encounter problems, please send a detailed message via email to the development team at the following email address: [umlsks@nlm.nih.gov](mailto:umlsks@nlm.nih.gov). Be sure to include the release version number for the UMLS<sub>KS</sub> system with your mailing.





## 1 Introduction

The Unified Medical Language System (UMLS) Knowledge Sources and related lexical programs, developed at the U.S. National Library of Medicine (NLM), provide access to the UMLS. The Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon are part of the UMLS and are designed primarily for use by system developers. They are meant to be consulted and used by application programs to interpret and refine user queries, to map the user's terms to appropriate controlled vocabularies and classification schemes, to interpret natural language, and to assist in structured data creation. They are also useful as reference tools for database builders, librarians and other information professionals.

The UMLS Knowledge Source Server (UMLS/SKS) is the set of machines, programs and Application Programmer Interfaces (APIs), written in Java, located and maintained by staff at the NLM that allow access to the UMLS/SKS services. This document describes the system architecture for this version of the UMLS Knowledge Source Server and supporting systems and tools. For the remainder of the document, UMLS/SKS will be used as shorthand for the UMLS Knowledge Source Server.

### 1.1 Requirements Summary

#### 1.1.1 Key Requirements and Design Goals

The driving requirements for the UMLS/SKS are as follows:

- Public access to UMLS Knowledge Sources via a web interface.
- Programmer access to the same functions and features of the UMLS/SKS available from the web interface.
- 24x7 availability, with minimal down time.

Additional design goals:

- Extensibility: the system is expected to have a long life and should improve with the technology over time.
- Scalability: the system is expected to handle an increasing number of user requests and larger UMLS vocabularies.
- Ease of administration: NLM and its contractors should be able to operate and maintain the system with a minimum of effort.
- System and software upgrades should generally not require any interruption of service, although moderate performance degradation is acceptable.

### 1.2 External Interfaces

The UMLS/SKS provides three mechanisms for external entities to interface with the UMLS/SKS. The first is through a web server running on a local machine at the NLM; the second is through an Application Programmer Interface (API) that connects user programs to the UMLS/SKS; and the third is a TCP/IP socket interface for non-Java programs to access the services of the UMLS/SKS. The web interface provides links to several web sites providing related information and services. The National Library of Medicine (NLM), National Institutes of Health (NIH), Health and Human Services (HHS) and other government organizations maintain these sites.

### 1.3 Performance, Sizing and Loading

The UMLS/SKS system is expected to occupy approximately 200 MB of disk space per UMLS release. The RAM required to efficiently run the UMLS/SKS will be determined during system testing but is expected to be on the order of 4 GB. Load balancing and changes to the machine configurations may be made after initial testing is complete.

### 1.4 Logical Architecture

The logical system architecture for the Knowledge Source Server is shown at a high level in Figure 1. The services of the UMLS/SKS can be accessed three different ways: through a web client using a standard browser (Netscape or Internet Explorer), through a program written to use the UMLS/SKS API, or through a TCP/IP socket-based interface. Data is returned to the user in XML form. A set of classes that provide a data-centric representation of the UMLS Metathesaurus, Semantic Network, and SPECIALIST Lexicon is provided with the API and is capable of reading the XML generated by the API methods (except **query**, see Section 3.4).

The web interface issues HTTP requests through the Internet to a web server. The web server then issues a request using Java’s Remote Method Invocation (RMI) methodology to execute a particular function on behalf of the user. The RMI Server receives the request, executes the request against the database, and formulates and returns that result to the web interface in XML.

The API issues RMI requests through the Internet directly to the RMI server using the RMI protocol. The RMI Server receives the request, executes the request against the database, and formulates and returns the result to the client API in one of two forms. The client program can request the information be returned in XML form and may subsequently use the Object Model classes to interpret the data as a set of data-centric objects. The object form is a set of classes that can be incorporated into the client application that provide functions for directly manipulating the database request results from within a Java program.

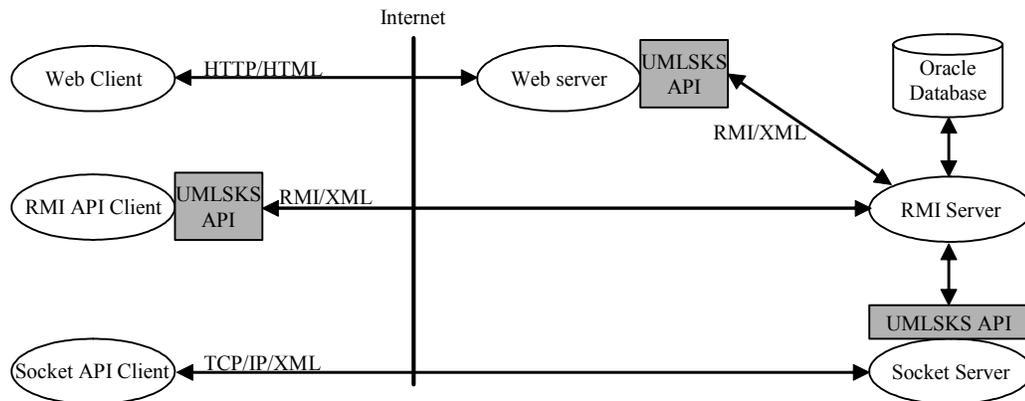
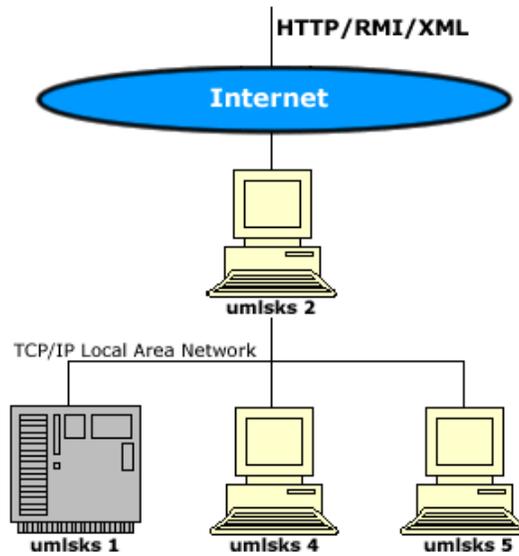


Figure 1 - Logical System Diagram

### 1.5 Physical Architecture

The four machines shown in Figure 2 will be used for the UMLS/SKS.



**Figure 2 - Physical System Environment**

Machine	Hardware Configuration	Functions
<b>umlks1</b>	Enterprise 4000, 4 CPU Sun Microsystems machine, 2 GB RAM	<ul style="list-style-type: none"> <li>- Resonate Back-up Server</li> <li>- Web Server</li> <li>- UMLSks Server/Oracle Database</li> </ul>
<b>umlks2</b>	Sun Sparc Ultra 2 workstation	<ul style="list-style-type: none"> <li>- Primary Resonate Server</li> <li>- MySQL User Accounts Database</li> </ul>
<b>umlks4</b>	Sun Sparc Ultra 2 workstation with 2 GB RAM	<ul style="list-style-type: none"> <li>- Web Server</li> <li>- UMLSks Server/Oracle Database</li> </ul>
<b>umlks5</b>	Sun Sparc Ultra 2 workstation with 2 GB RAM	<ul style="list-style-type: none"> <li>- Web Server</li> <li>- UMLSks Server/Oracle Database</li> </ul>

**Table 1 – Physical Machine/Processing Function Map**



## **2 Installing the UMLSKS**

The UMLSKS consists of software developed in the Cognitive Sciences Branch of the Lister Hill National Center for Biomedical Communications, a division of the National Library of Medicine, and a number of third party applications/toolkits. At this time, the UMLSKS is not available for user download and local installation. Plans are in place to develop a downloadable version. When this version becomes available, this chapter will detail instructions for downloading, installing, and running the UMLSKS on a user's local machine.



### 3 Building UMLSks Software Applications

The UMLSks API provides Java developers with convenience functions for retrieving Metathesaurus, Semantic Network, and SPECIALIST Lexicon data from the UMLSks and a set of classes that place an object model representation upon that data for convenience. The API is based upon Java's Remote Method Invocation (RMI) communications protocols.

#### 3.1 Building and Running Your Program

The complete set of files for the UMLSks API may be downloaded from the UMLSks website. The UMLSks API download includes sample Java programs that exercise the various functions of the API. The following instructions describe how to build and run these sample clients. Building and running your program will use similar steps. Upon downloading of the `kss-api-X.jar` file, where *X* is the version indicator for the software, you will need to "unjar" the contents into a local directory, hereafter referred to as `<APIINSTALLDIR>`. For example, to install version 4.0 of the API, you would execute the following:

```
cd <APIINSTALLDIR>
jar xvf ./kss-api-4.2.jar
```

After 'unjar'ring the contents, you will see at a minimum the following files and directories in `<APIINSTALLDIR>`:

```
data/    example/  html/    lib/    META-INF/  README.html
```

File/Directory	Description
data/	Contains required data files for use with the object model, including the client properties file.
data/sampleCommands/	The directory holds a set of sample commands that may be used as input to the <code>StandardQueryExecutor.request</code> method.
data/sampleQueries/	The directory holds a set of sample queries that may be used with the <code>KSSRetriever.query</code> method.
data/sampleXMLOutput/	Contains the output in XML for the various API methods.
example/	Contains example <code>.java</code> files that exercise the various API functionalities of the UMLSks.
html/	Documentation describing the object model in detail and the UMLSks RMI interface classes.
lib/	Contains the <code>.jar</code> files that must reside in the CLASSPATH for client programs to execute properly.
lib/kss-api-X.jar	Java <code>.jar</code> file containing the object model, UMLSks interface classes, and necessary third party classes with which a Java client program "links". <i>X</i> is the version of the API.
META-INF/	Directory containing the manifest of the <code>.jar</code> file contents.
README.html	Describes the installation of the client API.

### 3.1.1 Building the Example .java Files

The `.java` files, located in the example directory, are all Java programs and must be compiled with a Java 1.2 or later compiler. Assuming that Java is installed in `/usr/java` and your current working directory is set to `<APIINSTALLDIR>`, the command to build the program is

```
/usr/java/javac -classpath ../../lib/java/kss-api-4.2.jar *.java
```

Once all files are compiled, a number of class files will appear in the directory. These class files are used to run the sample applications that connect to the UMLS<sub>KS</sub>.

### 3.1.2 Running the Client

Each `Client` class provides a command-line entry points to execute all API functions for a given software release except the function to query the UMLS<sub>KS</sub> using an XML formatted query. Assuming that the version of Java you are using is installed in `/usr/java` and your current working directory is set to `<APIINSTALLDIR>`, the command to run the each program is

```
/usr/java/java -classpath ../../lib/java/kss-api-4.2.jar <CLIENT> \  
<SVRHOST> <SVRNAME>
```

where,

- `<CLIENT>` is the full package class name of the client class to be run;
- `<SVRHOST>` is the hostname of the server running the UMLS<sub>KS</sub> which should be `"umlsks.nlm.nih.gov"`; and
- `<SVRNAME>` is the name of the UMLS<sub>KS</sub> Service, `"KSSRetriever"`.

**Note:** If upon running the client you immediately receive a `Java RemoteException` indicating the connection could not be established, either one of two things has occurred. The first thing could be that the UMLS<sub>KS</sub> was installed on a different host or the service name is not `KSSRetriever`. In either case, see your UMLS<sub>KS</sub> administrator for the actual names to be used. The second thing could be that the UMLS<sub>KS</sub> is down for maintenance. Your UMLS<sub>KS</sub> administrator should be notified immediately of any unplanned outages of service.

This command assumes that class files listed previously and the `.jar` file are located in the current working directory. If the files are located elsewhere, then the `CLASSPATH` should be set to appropriately locate the `.class` and `.jar` files.

Once running, you will be presenting with a number of options in menu form similar to the one shown in Figure 3.



Figure 3 – Client.java Menu

For each of the menu options, you will be requested to enter necessary details required to complete the function's execution. For example, by selecting the option to find a concept using an input term when executing the `Client.class`, you will be prompted to enter the CUI or term name for the concept, any source abbreviation filtering, and any language filtering, and an output form (either to the screen or to a file). The data returned to the user and subsequently displayed or written to the output file are an XML formatted set of data. See some example output for details on the returned XML documents for each of the functions.

### 3.1.3 Running the ExpertClient

The `ExpertClient` class provides a command-line entry point to execute the function to query the UMLS/SKS using an XML formatted query. Assuming that the version of Java you are using is installed in `/usr/java` and your current working directory is set to `<APIINSTALLDIR>`, the command to run the program is

```

/usr/java/java -classpath ../../lib/kss-api-4.2.jar \
  ExpertClient <SVRHOST> <SVRNAME> <XMLQueryFile>

```

where,

- `<SVRHOST>` is the hostname of the server running the UMLS/SKS which should be **umlsks.nlm.nih.gov**, and
- `<SVRNAME>` is the name of the UMLS/SKS Service, **KSSRetriever**, and
- `<XMLQueryFile>` is the full path to a file containing an XML query as described in Section 4.2.

This command assumes that class files listed previously and the `.jar` file are located in the current working directory. If the files are located elsewhere, then the `CLASSPATH` should be set to appropriately locate the `.class` and `.jar` files.

Once running, you will see the XML output of the execution of the query contained within the argument XML query file output to the screen.

### 3.1.4 Running the SocketClient

The `SocketClient` class provides a command-line entry point to execute the Socket Server interface portion of the UMLS/SKS. The Socket Server accepts XML formatted commands and returns the resulting XML data through the same socket interface and is subsequently printed to the screen. Assuming that the version of Java you are using is installed in `/usr/java` and your current working directory is set to `<APIINSTALLDIR>`, the command to run the program is

```
/usr/java/java -classpath ../../lib/kss-api-4.0.jar \
  SocketClient <SVRHOST> <SVRNAME> <XMLCmdFile>
```

where,

- `<SVRHOST>` is the hostname of the server running the UMLSKS which should be **umlsks.nlm.nih.gov**, and
- `<SVRNAME>` is the name of the UMLSKS Socket Server port number, **8042**, and
- `<XMLCmdFile>` is the full path to a file containing a standard query defined in XML. Examples of these queries are described in Chapter 4.

This command assumes that class files listed previously and the `.jar` file are located in the current working directory. If the files are located elsewhere, then the `CLASSPATH` should be set to appropriately locate the `.class` and `.jar` files.

Once running, you will see the XML output of the execution of the query contained within the argument XML query file output to the screen.

### 3.1.5 Running the StandardQueryClient

The `StandardQueryClient` class provides a command-line entry point to execute the functionality provided by the `StandardQueryExecutor` class. Assuming that the version of Java you are running is installed in `/usr/java` and your current working directory is set to `<APIINSTALLDIR>`, the command to run the program is

```
/usr/java/java -classpath ../../lib/kss-api-4.0.jar \
  StandardQueryClient <SVRHOST> <SVRNAME> \
  <XMLQueryFile>
```

where,

- `<SVRHOST>` is the hostname of the server running the UMLSKS which should be **umlsks.nlm.nih.gov**, and
- `<SVRNAME>` is the name of the UMLSKS Socket Server port number, **8042**, and
- `<XMLQueryFile>` is the full path to a file containing a standard query defined in XML. Examples of these queries are described in Chapter 5.

This command assumes that class files listed previously and the `.jar` file are located in the current working directory. If the files are located elsewhere, then the `CLASSPATH` should be set to appropriately locate the `.class` and `.jar` files.

Once running, you will see the XML output of the execution of the query contained within the argument XML query file output to the screen.

## 3.2 API Package Structure

The packages shown in Figure 4, are contained within the `kss-api-4.2.jar` file and constitute the required set of packages to enable API communication with the UMLSKS.

Package Name	Package Description
<code>gov.nih.nlm.kss.api</code>	<i>KSSRetriever</i> , <i>KSSRetrieverV2_1</i> (Version 2.1), <i>KSSRetrieverV3_0</i> (Version 3.0), <i>KSSRetrieverV4_0</i> (Version 4.0), and <i>KSSRetrieverV4_2</i> (Version 4.2) interface definitions.

Package Name	Package Description
<i>gov.nih.nlm.kss.example</i>	Examples that exercise the various API functionalities of the UMLS/SKS.
<i>gov.nih.nlm.kss.util</i>	Utilities classes used by the retrieval classes and the object model classes.
<i>gov.nih.nlm.kss.models</i>	Object model classes for UMLS/SKS related data including classes to support spelling suggestions (V2.1) and ingestion of UMLS releases (V2.1).
<i>gov.nih.nlm.kss.models.meta</i>	Metathesaurus object model classes
<i>gov.nih.nlm.kss.models.meta.assocExp</i>	Object model classes representing associated expressions of a concept.
<i>gov.nih.nlm.kss.models.meta.attribute</i>	Object model classes representing attributes of a term.
<i>gov.nih.nlm.kss.models.meta.concept</i>	Object model classes representing a Metathesaurus concept.
<i>gov.nih.nlm.kss.models.meta.context</i>	Object model classes representing a Metathesaurus concept's hierarchical context.
<i>gov.nih.nlm.kss.models.meta.cooccurrence</i>	Object model classes representing co-occurring concepts amongst different UMLS sources.
<i>gov.nih.nlm.kss.models.meta.deltas</i>	Object model classes found in Version 2.1 representing the changes in the Metathesaurus for the various UMLS releases.
<i>gov.nih.nlm.kss.models.meta.locator</i>	Object model classes representing concept locator information.
<i>gov.nih.nlm.kss.models.meta.meshentry</i>	Object model classes representing the MeSH entries for a specific term name.
<i>gov.nih.nlm.kss.models.meta.relation</i>	Object model classes representing a concept's relations.
<i>gov.nih.nlm.kss.models.meta.source</i>	Object model for a UMLS source.
<i>gov.nih.nlm.kss.models.sem</i>	Semantic Network object model classes
<i>gov.nih.nlm.kss.models.sem.units</i>	Object model classes representing the components within the Semantic Network
<i>gov.nih.nlm.kss.models.sem.rels</i>	Object model classes representing the associative and hierarchical relations within the Semantic Network
<i>gov.nih.nlm.kss.models.lex</i>	SPECIALIST Lexicon object model classes

Figure 4 – UMLS/SKS API Packages

Packages `gov.nih.nlm.kss.api` and `gov.nih.nlm.kss.util` are described in the following sections. The packages relating to the Object Model, `gov.nih.nlm.kss.models.*` are described in Section 0.

### 3.2.1 Package `gov.nih.nlm.kss.api`

Figure 5 lists the classes and their descriptions for the `gov.nih.nlm.kss.api` package.

Class	Description
-------	-------------

Class	Description
<i>KSSRetriever</i>	Java Interface file defining the available Version 2.0 and later API functions for Java programs.
<i>KSSRetrieverV2_1</i>	Java interface file defining the available Version 2.1 API functions for Java programs.
<i>KSSRetrieverV3_0</i>	Java interface file defining the available Version 3.0 API functions for Java programs.
<i>KSSRetrieverV4_0</i>	Java interface file defining the available Version 4.0 API functions for Java programs.
<i>KSSRetrieverV4_2</i>	Java interface file defining the available Version 4.2 API functions for Java programs.

**Figure 5 – UMLSKS Connection Class**

### 3.2.2 Package `gov.nih.nlm.kss.util`

Figure 6 lists the classes and their descriptions for the `gov.nih.nlm.kss.util` package.

Class	Description
<i>Util</i>	Set of utility functions that enable reading of properties files and retrieval of necessary files for use with the Object Model.
<i>DatabaseException</i>	Class encapsulating details about an exception that occurred when attempting to access the UMLSKS data.
<i>XMLInterpreter</i>	Class from which all object model instances extend, providing useful utility functions for ingesting XML generated by the UMLSKS.
<i>XMLException</i>	Class encapsulating details about an exception generated during the reading of an XML document by the Object Model.

**Figure 6 – UMLSKS Utility Classes**

### 3.3 Program Initialization

The UMLSKS is implemented using the Remote Method Invocation feature of the Java language. RMI is a mechanism that enables an object on one Java virtual machine to invoke methods on an object in another Java virtual machine. Any object that can be invoked this way implements the Remote Interface. When such an object is invoked, its arguments are "marshaled" and sent from the local virtual machine to the remote one, where the arguments are "unmarshalled." When the method terminates, the results are marshaled from the remote machine and sent to the caller's virtual machine. If the method invocation results in an exception being thrown, the exception is indicated to the caller. The RMI interfaces are implemented in such a manner. This allows client programs to make calls "directly" as if the server were running locally.

The RMI registry acts as the router to establish a connection to a remote object running on a remote Java virtual machine. To request a "handle" to a remote object, you request one from the RMI registry using the assigned object name and host machine on which the object is running. In the case of the UMLSKS, the `KSSRetrieverV4_2` (which extends each of the interfaces `KSSRetriever`, `KSSRetrieverV2_1`, `KSSRetrieverV3_0`, and `KSSRetrieverV4_0`) is registered by the UMLSKS at startup and made available to the user with the object name of "KSSRetriever". For example, the following segment of code will establish a connection to the UMLSKS running on the machine named "`umlsks.nlm.nih.gov`" with the name "`KSSRetriever`". This object is used to issue requests to search the Metathesaurus. The following code snippet also shows how to "create" a

KSSRetrieverV4\_0 instance and request the return of a concept name described in the KSSRetriever interface definition.

To begin using the new features of a new release of the API, users must modify their code to perform the cast to the appropriate RMI interface class.

```
try
{
    String name = "//umlsks.nlm.nih.gov/KSSRetriever";

    KSSRetrieverV4_2 retriever =
        (KSSRetrieverV4_2)Naming.lookup(name);

    char[] result = retriever.getConceptName("2001",
        "C0001175", null, "ENG");
    String conceptName = new String(result);

    System.out.println("Concept Name in XML: " + conceptName);

} catch (RemoteException ex) {
    // handle remote exception
} catch (NotBoundException ex) {
    // handle fact that the server is not bound to the given name
} catch (MalformedURLException ex) {
    // handle fact that the name was not a properly formatted URL
}
}
```

### 3.4 UMLS SKS API Functions

The following interfaces provide a number of API functions giving access to the resources of the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon. The table shown below lists the functions and a brief description of the function. More detailed descriptions are available from each of the interface class descriptions.

**KSSRetriever Interface**

<b>Function</b>	<b>Description</b>
getCurrentUMLSVersion	Returns the UMLS data versions considered to be the current release version.
GgetUMLSVersions	Returns the list of UMLS data versions that can be accessed by the UMLS SKS.
getSWVersion	Returns the version of the UMLS SKS Software Release.
getConceptName	Obtains the concept name from the Metathesaurus for a given argument concept identifier and returns the data as a character buffer containing the concept name in XML format. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class ConceptIdVector..

**KSSRetriever Interface**

<b>Function</b>	<b>Description</b>
<code>findCUI</code>	Obtains the concept unique identifiers from the Metathesaurus for that have the concept name as one of its terms and returns the list of CUIs as a character buffer containing those concept unique identifiers in XML format. This information is accessed by matching the concept name to the one found in the database. If the connection to the UMLS database cannot be established, a null is returned to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptIdVector</code> .
<code>getConceptProperties</code>	Obtains Metathesaurus information on the terms, definitions, semantic type, and context for a given argument concept identifier and returns the data as a buffer of characters that contains the concept in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .
<code>findConcept</code>	Obtains Metathesaurus information on the terms, definitions, semantic type, and context for a given argument term name and returns the data as a buffer of characters that contains the concept in XML form. This is equivalent to performing a <code>findCUI</code> function call and subsequently feeding those CUIs to a call to <code>getConceptProperties</code> . This information is accessed by locating concepts whose terms match the input term. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .
<code>getBasicConceptProperties</code>	Obtains Metathesaurus information on the terms, definitions, and semantic type(s) for a given argument concept identifier and returns the data as a buffer of characters that contains the concept in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .

**KSSRetriever Interface**

<b>Function</b>	<b>Description</b>
<code>findBasicConcept</code>	Obtains Metathesaurus information on the terms, definitions, and semantic type(s) for a given argument term name and returns the data as a buffer of characters that contains the concept in XML form. This is equivalent to performing a <code>findCUI</code> function call and subsequently feeding those CUIs to a call to <code>getBasicConceptProperties</code> . This information is accessed by matching the term name using the scheme specified to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .
<code>getTerminology</code>	Obtains Metathesaurus information on the terms found in a set of sources for a given argument concept identifier and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermVector</code> .
<code>getTerminology</code>	Obtains Metathesaurus information on the terms found in a particular source for a given argument concept identifier and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermVector</code> .
<code>getTerms</code>	Obtains Metathesaurus information on the terms found in a set of sources for a given argument term to be matched exactly and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermCollection</code> .

**KSSRetriever Interface**

<b>Function</b>	<b>Description</b>
<code>getDefinition</code>	Obtains Metathesaurus information on the definition for a given argument concept identifier and returns the data as a buffer of characters that contain the definition in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>DefinitionVector</code> .
<code>getSemanticType</code>	Obtains Metathesaurus information on the semantic types for a given concept identifier and returns the data as a buffer of characters that contain the semantic type information in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>SemTypeVector</code> .
<code>getContext</code>	Obtains Metathesaurus context information for a given argument concept unique identifier and returns the data as a buffer of characters that contain the context in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ContextVector</code> .
<code>getAssocExprs</code>	Obtains Metathesaurus associated expressions found in a specific source for a given argument concept identifier and returns the data as a character buffer containing the associated expressions in XML format. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>AssociatedExpVector</code> .

**KSSRetriever Interface**

<b>Function</b>	<b>Description</b>
<code>getCooccurrences</code>	Obtains Metathesaurus information on the co-occurrences found in a given source with the given co-occurrence type for a given argument concept identifier and returns the data as a character buffer containing the co-occurrences in XML format. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>CooccurrenceVector</code> .
<code>getRelations</code>	Obtains Metathesaurus information on the relationships for a given argument concept and returns a set of characters containing the XML representation for the relations. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>RelationVector</code> .
<code>getStringAttributes</code>	Obtains Metathesaurus information on the term attributes found in a particular source with a specific attribute name for a given argument concept and returns the data as a buffer of characters that contain the attributes in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, a null is returned to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermAttributeVector</code> .
<code>getLocator</code>	Obtains the locator information from the Metathesaurus and returns the data as a buffer of characters that contain the locators in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established a null is returned to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>LocatorVector</code> .

**KSSRetriever Interface**

<b>Function</b>	<b>Description</b>
getMeSHEntries	Obtains the MeSH entries for a particular concept unique identifier from the Metathesaurus. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <i>MeSHEntryVector</i> .
getMeSHInfo	Obtains the MeSH terms for a particular DUI from the Metathesaurus. This information is accessed by matching the DUI to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <i>MeSHInfoVector</i> .
getRawRecords	Obtains the raw records from the Metathesaurus from the specified table. This information is accessed by matching the DUI to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller.
query	Executes a query formatted in the UMLSKS XML language described in Chapter 4.
describeSource	Returns a triplet describing the argument source including its abbreviation, its short name, and its long name. The XML data returned from this function call can be interpreted using the Object Model class <i>SourceVector</i> .
listSources	Returns a list of triplets that describes each source including its abbreviation, its short name, and its long name. The XML data returned from this function call can be interpreted using the Object Model class <i>SourceVector</i> .

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
describeCurrentUMLSVersion	Returns the UMLS data versions, in XML format, considered to be the current release version. The XML data returned from this function call can be interpreted using the Object Model class <i>UMLSYearVector</i> .
describeUMLSVersions	Returns the list of UMLS data versions, in XML format, that can be accessed by the UMLSKS. The XML data returned from this function call can be interpreted using the Object Model class <i>UMLSYearVector</i> .

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
<code>findCUI</code>	Obtains the concept unique identifiers from the Metathesaurus for that have the concept name as one of its terms and returns the list of CUIs as a character buffer containing those concept unique identifiers in XML format. This information is accessed by matching the concept name to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, a null is returned to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptIdVector..</code>
<code>findCUI</code>	Obtains the concept unique identifiers from the Metathesaurus for concepts belonging to the specified semantic type and returns the list of CUIs as a character buffer containing those concept unique identifiers in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptIdVector..</code>
<code>findCUI</code>	Obtains the concept unique identifiers from the Metathesaurus for concepts with the specified source id in a specific source vocabulary and returns the list of CUIs as a character buffer containing those concept unique identifiers in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptIdVector..</code>
<code>getConceptProperties</code>	Obtains Metathesaurus information on the terms, definitions, semantic type, and context for a given argument concept identifier and returns the data as a buffer of characters that contains the concept in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector.</code>

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
<code>findConcept</code>	Obtains Metathesaurus information on the terms, definitions, semantic type, and context for a given argument term name and returns the data as a buffer of characters that contains the concept in XML form. This is equivalent to performing a <code>findCUI</code> function call and subsequently feeding those CUIs to a call to <code>getConceptProperties</code> . This information is accessed by locating concepts whose terms match the input term. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .
<code>getBasicConceptProperties</code>	Obtains Metathesaurus information on the terms, definitions, and semantic type(s) for a given argument concept identifier and returns the data as a buffer of characters that contains the concept in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .
<code>findBasicConcept</code>	Obtains Metathesaurus information on the terms, definitions, and semantic type(s) for a given argument term name and returns the data as a buffer of characters that contains the concept in XML form. This is equivalent to performing a <code>findCUI</code> function call and subsequently feeding those CUIs to a call to <code>getBasicConceptProperties</code> . This information is accessed by matching the term name using the scheme specified to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ConceptVector</code> .

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
getTerminology	Obtains Metathesaurus information on the terms found in a set of sources for a given argument concept identifier and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermVector</code> .
getTerminology	Obtains Metathesaurus information on the terms found in a particular source for a given argument concept identifier and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermVector</code> .
getTerms	Obtains Metathesaurus information on the terms found in a set of sources for a given argument term to be matched exactly and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermCollection</code> .
getAssocExprs	Obtains Metathesaurus associated expressions found in a set of sources for a given argument concept identifier and returns the data as a character buffer containing the associated expressions in XML format. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>AssociatedExpVector</code> .

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
getLocators	Obtains the locator information from the Metathesaurus and returns the data as a buffer of characters that contain the locators in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established a null is returned to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>LocatorVector</code> .
getStringAttributes	Obtains Metathesaurus information on the term attributes found in a set of sources with a specific attribute name for a given argument concept and returns the data as a buffer of characters that contain the attributes in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. If the connection to the UMLS database cannot be established, a null is returned to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermAttributeVector</code> .
findLUI	Obtains the LUI/term name pairs that match an input term name and returns the data as a buffer of characters that contain the pair in XML form. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermIdVector</code> .
getTermsForLUI	Obtains the LUI/term name pairs that match an input LUI and returns the data as a buffer of characters that contains the pairs in XML form. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermIdVector</code> .
findSUI	Obtains the SUI/string pairs that match an input term name and returns the data as a buffer of characters that contain the pair in XML form. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>StringIdVector</code> .

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
<code>getStringsForSUI</code>	Obtains the SUI/string pairs that match an input SUI and returns the data as a buffer of characters that contains the pairs in XML form. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>StringIdVector</code> .
<code>listStrAttrs</code>	Returns a list of pairs that describes each string attribute including its attribute abbreviation (a 2-3 character string) and a description of the attribute. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>AttrVector</code> .
<code>listMeSHQuals</code>	Returns a list of pairs that describes each MeSH qualifier including its code and name. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>MeSHQualifierVector</code> .
<code>listTermTypes</code>	Returns a list of pairs that describes each term type (TTY) including its code and description. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermTypeVector</code> .
<code>listCooccurrenceTypes</code>	Returns a list of pairs that describes each cooccurrence type (COT) including its code and description. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>CooccurrenceTypeVector</code> .
<code>ListStringTypes</code>	Returns a list of pairs that describes each string type (STT) including its code and description. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>StringTypeVector</code> .
<code>listRelationTypes</code>	Returns a list of pairs that describes each relation type (REL) including its code and description. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>RelationTypeVector</code> .

**KSSRetrieverV2\_1 Interface**

<b>Function</b>	<b>Description</b>
describeUMLChanges	Returns a list of changes that were made to the UMLS for all releases, specific releases, or specific CUIs. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ReleaseDeltaVector</code> .
listDictionaries	Returns a list of dictionary names that can be used as arguments to the function <i>suggestSpelling</i> . If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>DictionaryVector</code> .
suggestSpelling	Returns a list of spelling suggestions from the specified dictionary for the requested term. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>SpellingSuggestionsVector</code> .

**KSSRetrieverV3\_0 Interface**

<b>Function</b>	<b>Description</b>
findSemType	Returns the attributes/properties of the semantic type whose name contains the argument string. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticTypeVector</code> .
getSemTypeProperties	Returns the attributes/properties of the requested semantic type in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticTypeVector</code> .

**KSSRetrieverV3\_0 Interface**

<b>Function</b>	<b>Description</b>
<code>getSemTypeAncestors</code>	Returns the parent tree of the requested semantic type name or unique identifier in XML format. Users have the option of limiting the amount of data returned by the method by specifying whether the complete hierarchical tree should be returned. In unexpanded mode, only the direct line of ancestors without their children (except the one leading to the semantic type requested) is returned. The expanded form will create the entire hierarchical tree for the requested semantic type, including all ancestors, siblings, and cousins. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticTypeVector</code> .
<code>getSemTypeSiblings</code>	Returns the list of siblings of the requested semantic type name or unique identifier. Users have the option of limiting the amount of data returned by the method by specifying whether the complete hierarchical tree should be returned. In unexpanded mode, only the unique identifiers of the children of the requested semantic type's siblings are returned within the <code>HierSemanticType</code> instances. The expanded form will create the entire hierarchical tree for each of the siblings of the requested semantic type, including all descendants. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticTypeVector</code> .
<code>listSemTypeIds</code>	Returns the list of possible semantic type name/unique identifier pairs, either of which can be passed as arguments to the <code>getAssociativeRelations</code> or the <code>getSemTypeProperties</code> methods. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <code>SemNetIdVector</code> .
<code>findSemRelation</code>	Returns the attributes/properties of the semantic relation whose name contains the argument string. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticRelationVector</code> .

**KSSRetrieverV3\_0 Interface**

<b>Function</b>	<b>Description</b>
<code>getSemRelationProperties</code>	Returns the attributes/properties of the requested semantic relation in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticRelationVector</code> .
<code>getSemRelationAncestors</code>	Returns the parent tree of the requested semantic relation name or unique identifier in XML format. Users have the option of limiting the amount of data returned for the method by specifying whether the complete hierarchical tree should be returned. In unexpanded mode, only the direct line of ancestors without their children (except the one leading to the semantic type requested) is returned. The expanded form will create the entire hierarchical tree for the requested semantic relations, including all ancestors, siblings, and cousins. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticRelationVector</code> .
<code>listSemRelationIds</code>	Returns the list of possible semantic relation name/unique identifier pairs, either of which can be passed as arguments to the <code>getAssociativeRelations</code> or the <code>getSemRelationProperties</code> methods. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <code>SemNetIdVector</code> .
<code>existsAssociativeRelation</code>	Returns an indicator as to whether the specified semantic relation holds for the given pair of semantic types. The indicator is either shown as a direct relationship, as an inherited relation indicating that the relationship is established between ancestors for the relationship or as a non-existent relationship. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <code>AssociativeRelExistence</code> .
<code>getAssociativeRelations</code>	Returns the set of triplets where the defined semantic relationship holds between two semantic types. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <code>AssociativeRelationVector</code> .

**KSSRetrieverV3\_0 Interface**

<b>Function</b>	<b>Description</b>
existsHierRelRelation	Returns an indicator as to whether the specified relationship holds for two semantic relation names. This relationship is typically the hierarchical "isa" relationship. The indicator is either shown as a direct relationship, as an inherited relation indicating that the relationship is established between ancestors for the relationship or as a non-existent relationship. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <i>HierRelRelExistence</i> .
listSemGroups	Returns a list of the valid semantic groups within the Semantic Network. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <i>SemGroupVector</i> .
listSemTypes	Returns a list of semantic type name/unique identifier pairs that belong to the requested semantic group. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <i>SemGroupVector</i> .
getSemNetASCIIRecords	Returns the ' ' separated list of ASCII relational records for the requested semantic network table. If the connection to the UMLS database cannot be established, an exception is thrown to the caller.
getSemGroup	Returns the name of the semantic group to which the given concept unique identifier (CUI) belongs. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML returned from this method can be interpreted using the Object Model class <i>SemGroupVector</i> .

**KSSRetrieverV4\_0 Interface**

<b>Function</b>	<b>Description</b>
findBasicSemType	Returns the basic attributes/properties of the semantic type whose name contains the argument string. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <i>HierSemanticTypeVector</i> .

**KSSRetrieverV4\_0 Interface**

<b>Function</b>	<b>Description</b>
<code>findBasicSemRelation</code>	Returns the attributes/properties of the semantic relation whose name contains the argument string. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>HierSemanticRelationVector</code> .
<code>getLexicalRecords</code>	Returns the lexical records from the SPECIALIST Lexicon for the argument term. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>LexicalRecordVector</code> .
<code>listMetaTableName</code>	Returns the list of Metathesaurus database table names each of which can be passed as an argument to the <code>getRawRecords</code> method. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TableNameVector</code> .
<code>listSemNetTableNames</code>	Returns the list of Semantic Network database table names each of which can be passed as an argument to the <code>getSemNetASCIIRecords</code> method. The results are returned in XML format. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TableNameVector</code> .

**KSSRetrieverV4\_2 Interface**

<b>Function</b>	<b>Description</b>
<code>findTerms</code>	Obtains Metathesaurus information on the terms found in a set of sources for a given argument term to be matched exactly and returns the data as a buffer of characters that contain the terms in XML form. This information is accessed by matching the concept unique identifier to the one found in the database. The method also takes the suppressibility flag into account when determining the data to be returned. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>TermCollection</code> .

**KSSRetrieverV4 2 Interface**

<b>Function</b>	<b>Description</b>
<code>describeUMLChanges</code>	Returns a list of changes that were made in a specific UMLS release for all releases, specific releases, or specific CUIs. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>ReleaseDeltaVector</code> .
<code>listDocEntryTypes</code>	Returns a list of document entry descriptions which are the metadata elements describing the table content restrictions and valid values. If the connection to the UMLS database cannot be established, an exception is thrown to the caller. The XML data returned from this function call can be interpreted using the Object Model class <code>DocEntryVector</code> .

**3.5 Using the UMLS SKS Object Model**

The XML data returned for all of the API functions, with the exception of the query method, may be interpreted by the object model. The following sections detail the classes that can interpret the various types of XML documents returned from the API functions.

**3.5.1 Properties Required by a UMLS SKS Client**

Version 2.0.x requires clients to initialize a set of properties that read disk files on the user's local machine to initialize the set of MeSH qualifiers, the set of sources, and the set of string attributes. These properties and a description of their purposes are included below. As a convenience, the set of properties are defined and delivered with the UMLS SKS API download in the `data/client.properties` file.

<b>Property Name</b>	<b>Property Description</b>
<code>KssStringAttributesFile</code>	This property is a full path file name to the location of the String Attributes File. This file is read by the <code>Attr</code> class (part of the UMLS SKS Object Model).
<code>KssMeshQualifierFile</code>	This property is a full path file name to the location of the MeSH Qualifier description file. This file is read by the <code>MeSHQualifier</code> class (part of the UMLS SKS Object Model).
<code>KssSourceFile</code>	This property is a full path file name to the location of the Source descriptions file. This file is read by the <code>Source</code> class (part of the UMLS SKS Object Model).

As of Version 2.1, each of these data items are now available through a set of APIs:

`listStrAttrs()`, `listMeSHQuals()`, and `listSources()` respectively. As a result, there are no specific client properties that must be set for the API libraries to execute properly.

### 3.5.2 Package `gov.nih.nlm.kss.models`

This package contains classes that are not specific to any particular model, but hold data from a number of general purpose methods. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.model` package.

UMLSKS Version	Class	Description
2.1	<code>DictionaryVector</code>	Subclass of <code>KSStringVector</code> that holds <code>String</code> instances and can interpret the XML returned from the <code>listDictionaries</code> method of the <code>KSSRetrieverV2_1</code> class.
4.0	<code>KSCuiVector</code>	Subclass of <code>KSVector</code> that holds instances of UMLSKS objects that are linked to a particular concept unique identifier and can be written to and read from an XML document.
4.0	<code>KXObject</code>	Holds a UMLSKS object that can be written to an XML stream and may have associated query and UMLS release strings.
4.0	<code>KSStringVector</code>	Subclass of <code>KSVector</code> that holds instances of <code>Strings</code> that can be written to and read from an XML document.
4.0	<code>KSVector</code>	Subclass of <code>Vector</code> that holds a list of UMLSKS objects that can be written to an XML stream and may have associated query and UMLS release strings.
2.1	<code>SpellingSuggestionsVector</code>	Subclass of <code>KSStringVector</code> that holds <code>String</code> instances and can interpret the XML returned from the <code>suggestSpelling</code> method of the <code>KSSRetrieverV2_1</code> class.
4.0	<code>TableNameVector</code>	Subclass of <code>KSStringVector</code> that holds <code>String</code> instances containing the database table names supported by the UMLSKS and can interpret the XML returned from the <code>listMetaTableNames</code> and <code>listSemNetTableNames</code> method of the <code>KSSRetrieverV4_0</code> class.
2.0	<code>UMLSYearVector</code>	Subclass of <code>KSStringVector</code> that holds <code>String</code> instances containing the UMLS Releases supported by the UMLSKS and can interpret the XML returned from the <code>describeUMLSVersions</code> method of the <code>KSSRetrieverV2_1</code> class.

### 3.5.3 Package `gov.nih.nlm.kss.models.meta.concept`

This package contains classes that represent the basic details of a concept. A concept within the UMLS/SKS is defined as having a concept name, a concept unique identifier (CUI), a set of terms and variants, a set of definitions, a set of semantic types, and a hierarchical context as related to other concepts. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.concept` package.

UMLS/SKS Version	Class	Description
2.0	<code>Concept</code>	Container class for all information associated with a concept (terms, definitions, semantic types, contexts, associated expressions, relations, co-occurrences, string attributes, and locators).
2.0	<code>ConceptVector</code>	Subclass of <code>Vector</code> that holds <code>Concept</code> instances and can interpret the XML returned from the <code>findConcept</code> , <code>getConceptProperties</code> , <code>findBasicConcept</code> , and <code>getBasicConceptProperties</code> methods of the <code>KSSRetriever</code> and <code>KSSRetrieverV2_1</code> interfaces.
2.0	<code>ConceptId</code>	Container class for the concept unique identifier and concept name.
2.0	<code>ConceptIdVector</code>	Subclass of <code>Vector</code> that holds <code>ConceptId</code> instances and can interpret the XML returned from the <code>findCUI</code> and <code>getConceptName</code> methods of the <code>KSSRetriever</code> and <code>KSSRetrieverV2_1</code> interfaces.
2.0	<code>Definition</code>	Container class for information about a concept's definition, including the string definition and the UMLS source for that definition.
2.0	<code>DefinitionVector</code>	Subclass of <code>Vector</code> that holds <code>Definition</code> instances and can interpret the XML returned from the <code>getDefinition</code> method of the <code>KSSRetriever</code> class.
2.0	<code>SemType</code>	Container class for information about a concept's semantic type, including the type unique identifier and the semantic type name.
2.0	<code>SemTypeVector</code>	Subclass of <code>Vector</code> that holds <code>SemType</code> instances and can interpret the XML returned from the <code>getSemanticType</code> method of the <code>KSSRetriever</code> class.
2.1	<code>StringId</code>	Container class for the pairing of an SUI with a string.
2.1	<code>StringIdVector</code>	Subclass of <code>Vector</code> that holds <code>StringId</code> instances and can interpret the XML returned from the <code>findSUI</code> and <code>getStringsForSUI</code> methods of the <code>KSSRetrieverV2_1</code> class.
2.1	<code>TermId</code>	Container class for the pairing of an LUI with a term name.
2.1	<code>TermIdVector</code>	Subclass of <code>Vector</code> that holds <code>TermId</code> instances and can interpret the XML returned from the <code>findLUI</code> and <code>getTermsForLUI</code> methods of the <code>KSSRetrieverV2_1</code> class.

2.0	StringInfo	Container class for information about the variants of a term, including the string unique identifier, a string type, the string name, and the UMLS sources in which the string was found.
2.0	StringSource	Container class for information about the source of a term variant, including the UMLS source of the variant, the term's type, the string code, and the restriction level.
2.0	Term	Container class for information about a term, including the term's lexical unique identifier, the term name, the UMLS status of the term, and the term's language.
2.0	TermVector	Subclass of <code>Vector</code> that holds <code>Term</code> instances and can interpret the XML returned from the two <code>getTerminology</code> methods of the <code>KSSRetriever</code> class.
2.0	TermCollection	Subclass of <code>Vector</code> that holds <code>TermVector</code> instances and can interpret the XML returned from <code>getTerms</code> method of the <code>KSSRetriever</code> and <code>KSSRetrieverV2_1</code> class and the <code>findTerms</code> method of the <code>KSSRetrieverV4_2</code> class.
2.1	TermType	Container class for the pairing of a term type and its description.
2.1	TermTypeVector	Subclass of <code>Vector</code> that holds <code>TermType</code> instances and can interpret the XML returned from the <code>listTermTypes</code> method of the <code>KSSRetrieverV2_1</code> class.

### 3.5.4 Package `gov.nih.nlm.kss.models.meta.context`

This package contains classes that represent the hierarchical context details of a concept. A `Context` within the UMLSKS is defined as a set of relatives of a particular `Concept`, including ancestors, children, and siblings. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.context` package.

UMLSKS Version	Class	Description
2.0	<code>Context</code>	Container class for all information associated with a concept's context - a set of <code>StringCxt</code> instances.
2.0	<code>ContextVector</code>	Subclass of <code>Vector</code> that holds <code>Context</code> instances and can interpret the XML returned from the <code>getContext</code> method of the <code>KSSRetriever</code> class.
2.0	<code>StringCxt</code>	Container class for the relative's string value and the UMLS source in which the relative was found.
2.0	<code>SourceCxt</code>	Container class for the information about a particular string relative name in a specific UMLS source, including the ancestors, children, and siblings found in that source (instances of class <code>CxtMember</code> ).
2.0	<code>CxtMember</code>	Container class for the information about specific node in the hierarchy, including the node name, the concept unique identifier for the node, the hierarchical code, the relationship value, and a flag indicating whether the node has children.

### 3.5.5 Package `gov.nih.nlm.kss.models.meta.assocExp`

This package contains classes that represent the associated expressions for a concept. An `AssociatedExp` within the UMLS SKS is defined as a relationship expression, the associated expression itself, and the source of that association. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.assocExp` package.

UMLS SKS Version	Class	Description
2.0	<code>AssociatedExp</code>	Container class for all information for a concept's associated expressions.
2.0	<code>AssociatedExpVector</code>	Subclass of <code>Vector</code> that holds <code>AssociatedExp</code> instances and can interpret the XML returned from the <code>getAssocExprs</code> method of the <code>KSSRetriever</code> class or from the overloaded method <code>getAssocExprs</code> in the <code>KSSRetrieverV2_1</code> class.

### 3.5.6 Package `gov.nih.nlm.kss.models.meta.attribute`

This package contains classes that represent the associated expressions for a concept. A `TermAttribute` within the UMLS SKS is defined as a set of string attributes with an associated lexical identifier for the term. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.attribute` package.

UMLS SKS Version	Class	Description
2.0	<code>TermAttribute</code>	Container class for all information for a concept's string attributes.
2.0	<code>TermAttributeVector</code>	Subclass of <code>Vector</code> that holds <code>TermAttribute</code> instances and can interpret the XML returned from the <code>getStringAttributes</code> method of the <code>KSSRetriever</code> class or from the overloaded method <code>getStringAttributes</code> in the <code>KSSRetrieverV2_1</code> class.
2.0	<code>StringAttribute</code>	Container class for information about a particular string, including its unique identifier and the string value itself.
2.0	<code>AttributeContext</code>	Container class for details contained in the <code>StringAttribute</code> that are grouped according to UMLS source.
2.0	<code>AttributeValue</code>	Container class for details of the <code>AttributeContext</code> including the attribute name and its value.
2.0	<code>Attr</code>	Container class for details of a string attribute (name and description)
2.0.1	<code>AttrVector</code>	Subclass of <code>Vector</code> that holds <code>Attr</code> instances and can interpret the XML returned from the <code>listStrAttrs</code> method of the <code>KSSRetrieverV2_1</code> class.

### 3.5.7 Package `gov.nih.nlm.kss.models.meta.cooccurrence`

This package contains classes that represent the co-occurrences of a concept. A `CooccurringConcept` within the UMLSKS is defined as a pair of concepts that occur together in the same “entries” in a particular source. This information includes the CUIs for each of the concepts in the co-occurrence and the details of the source in which the co-occurrence was found. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.cooccurrence` package.

UMLSKS Version	Class	Description
2.0	<code>CooccurringConcept</code>	Container class for all information about a co-occurring concept within the Metathesaurus.
2.0	<code>CooccurrenceVector</code>	Subclass of <code>Vector</code> that holds <code>CooccurringConcept</code> instances and can interpret the XML returned from the <code>getCooccurrences</code> method of the <code>KSSRetriever</code> class.
2.0	<code>COContext</code>	Container class for all information about the context of a co-occurring concept, including the source for the co-occurrence, the type of it, its frequency and a list of the qualifier frequencies of type <code>QualifierFrequencyVector</code> for the co-occurrence.
2.0	<code>QualifierFrequency</code>	Container class for the information about a MeSH Qualifier's frequency, including the qualifier's code and its name represented by the class <code>MeSHQualifier</code> , and the frequency for that qualifier within the source.
2.0	<code>QualifierFrequencyVector</code>	Subclass of <code>Vector</code> that holds <code>QualifierFrequency</code> instances.
2.0	<code>MeSHQualifier</code>	Container for the representation of a MeSH Qualifier including its 2-character code and a <code>String</code> qualifier name.
2.0.1	<code>MeSHQualifierVector</code>	Subclass of <code>Vector</code> that holds <code>MeSHQualifier</code> instances and can interpret the XML returned from the <code>listMeSHQuals</code> method of the <code>KSSRetrieverV2_1</code> class.
2.1	<code>CooccurrenceType</code>	Container for the representation of a cooccurrence type including its 2-character code and a <code>String</code> cooccurrence type description.
2.1	<code>CooccurrenceTypeVector</code>	Subclass of <code>Vector</code> that holds <code>CooccurrenceType</code> instances and can interpret the XML returned from the <code>listCooccurrenceTypes</code> method of the <code>KSSRetrieverV2_1</code> class.

### 3.5.8 Package `gov.nih.nlm.kss.models.meta.locator`

This package contains classes that represent the locators of a concept. A `Locator` within the UMLS/SKS is defined as a set of sources in which the Metathesaurus concept was detected. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.locator` package.

UMLS/SKS Version	Class	Description
2.0	<code>Locator</code>	Container class for all information for a concept's locator information..
2.0	<code>LocatorVector</code>	Subclass of <code>Vector</code> that holds <code>Locator</code> instances and can interpret the XML returned from the <code>getLocator</code> method of the <code>KSSRetriever</code> class or from the <code>getLocators</code> method of the <code>KSSRetrieverV2_1</code> class.

### 3.5.9 Package `gov.nih.nlm.kss.models.meta.meshentry`

This package contains classes that represent the MeSH Entries of a concept and contains classes that describe the Medical Subject Heading information about a specific DUI. A `MeSHEntry` within the UMLS/SKS is defined as the set of term characteristics within MeSH. The `MeSHInfo` class contains the information associated with a given DUI. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.meshentry` package.

UMLS/SKS Version	Class	Description
2.0	<code>MeSHEntry</code>	Container class for all information for the MeSH Entries for a term.
2.0	<code>MeSHEntryVector</code>	Subclass of <code>Vector</code> that holds <code>MeSHEntry</code> instances and can interpret the XML returned from the <code>getMeSHEntries</code> method of the <code>KSSRetriever</code> class.
2.0	<code>MeSHInfo</code>	Container class for all information associated with a specific DUI.
2.0	<code>MeSHInfoVector</code>	Subclass of <code>Vector</code> that holds <code>MeSHInfo</code> instances and can interpret the XML returned from the <code>getMeSHInfo</code> method of the <code>KSSRetriever</code> class.

### 3.5.10 Package `gov.nih.nlm.kss.models.meta.relation`

This package contains classes that represent the relations of a concept. A `Relation` within the UMLS/SKS is defined as a defined relationship between two concepts and documents the concept identifiers for the two, the relationship between them and other relational information. The table below lists the classes and their descriptions for the `gov.nih.nlm.kss.models.meta.relation` package.

UMLS/SKS Version	Class	Description
2.0	<code>Relation</code>	Container class for all information for a concept's relations.
2.0	<code>RelationVector</code>	Subclass of <code>Vector</code> that holds <code>Relation</code> instances and can interpret the XML returned from the <code>getRelations</code> method of the <code>KSSRetriever</code> class.
2.1	<code>RelationType</code>	Container class for the pairing of a relation type and its

UMLSKS Version	Class	Description
		description.
2.1	RelationTypeVector	Subclass of Vector that holds RelationType instances and can interpret the XML returned from the listRelationTypes method of the KSSRetrieverV2_1 class.

### 3.5.11 Package gov.nih.nlm.kss.models.meta.source

This package contains classes that represent a UMLS Source. A Source within the UMLSKS is defined as a source abbreviation, a long name for the source, and a short name for that source. The table below lists the classes and their descriptions for the gov.nih.nlm.kss.models.meta.source package.

UMLSKS Version	Class	Description
2.0	Source	Container class for all information for a UMLS Source.
2.0	SourceVector	Subclass of Vector that holds Source instances and can interpret the XML returned from the listSources method of the KSSRetriever class and from the overloaded listSources method of the KSSRetrieverV2_1 class.

### 3.5.12 Package gov.nih.nlm.kss.models.meta.deltas

This package contains classes that represent the changes made to the UMLS for the various UMLS delivery releases. The table below lists the classes and their descriptions for the gov.nih.nlm.kss.models.meta.deltas package.

UMLSKS Version	Class	Description
2.1	ReleaseDeltaVector	Subclass of Vector that holds ReleaseDelta instances and can interpret the XML returned from the describeUMLChanges method of the KSSRetrieverV2_1 and KSSRetrieverV4_2 classes.
2.1	ReleaseDelta	Container class for information about a specific UMLS release's changes (i.e. contains a list of ConceptDelta instances).
2.1	ConceptDelta	Container class for a specific concept changes made for a particular release. The information includes the CUI in question, a flag indicating the change made (either merged into another CUI or deleted altogether), and the CUI for the merger, if applicable.

### 3.5.13 Package gov.nih.nlm.kss.models.meta

This package contains classes that represent general purpose information about the Metathesaurus, including table entry descriptions that detail the valid values and ranges for the fields in the object model instances. The table below lists the classes and their descriptions for the gov.nih.nlm.kss.models.meta package.

UMLS/SKS Version	Class	Description
4.2	DocEntryVector	Subclass of Vector that holds DocEntryTypeVector instances and can interpret the XML returned from the listDocEntryTypes method of the KSSRetrieverV4_2 class.
4.2	DocEntryTypeVector	Subclass of Vector that holds DocEntryType instances.
4.2	DocEntryType	Container class for the metadata descriptions. The information includes the document entry type, the document entry subtype, the value, and a description. For example, the LAT entry found in the term information can have values as described in the entry where the document entry type is 'LAT'.

### 3.5.14 Package gov.nih.nlm.kss.models.sem.units

This package contains classes that represent the Semantic Network components (semantic type and semantic relation). The table below lists the classes and their descriptions for the gov.nih.nlm.kss.models.sem.units package.

UMLS/SKS Version	Class	Description
3.0	HierSemanticRelationVector	Subclass of Vector that holds HierSemanticRelation instances and can interpret the XML returned from the findSemRelation, getSemRelationProperties, and getSemRelationAncestors methods of the KSSRetrieverV3_0 class.
3.0	HierSemanticRelation	Container class for information about a semantic relation within the Semantic Network.
3.0	HierSemanticTypeVector	Subclass of Vector that holds HierSemanticType instances and can interpret the XML returned from the findSemType, getSemTypeProperties, getSemTypeAncestors, and getSemTypeSiblings methods of the KSSRetrieverV3_0 class and findBasicSemType and getBasicSemTypeProperties methods of the KSSRetrieverV4_0 class.
3.0	HierSemanticType	Container class for information about a semantic type within the Semantic Network.
3.0	SemGroupVector	Subclass of Vector that holds SemGroup instances and can interpret the XML returned from the listSemTypes and listSemGroups methods of the KSSRetrieverV3_0 class.

3.0	SemGroup	Container class for information about a semantic group defined within the Semantic Network.
3.0	SemNetIdVector	Subclass of Vector that holds SemNetId instances and can interpret the XML returned from the listSemTypeIds and listSemRelationIds methods of the KSSRetrieverV3_0 class.
3.0	SemNetId	Container class for information about a semantic type or relation including its name, unique identifier, and type or relation indicator within the Semantic Network.

### 3.5.15 Package gov.nih.nlm.kss.models.sem.rels

This package contains classes that represent the relationships defined within the Semantic Network. These include associative relationships between two semantic types and hierarchical relationships between two semantic relations. The table below lists the classes and their descriptions for the gov.nih.nlm.kss.models.sem.rels package.

UMLSKS Version	Class	Description
3.0	AssociativeRelationVector	Subclass of Vector that holds AssociativeRelation instances and can interpret the XML returned from the getAssociativeRelations method of the KSSRetrieverV3_0 class and the getBasicAssociativeRelations method of the KSSRetrieverV4_0 class.
3.0	AssociativeRelation	Container class for information about the relationship formed between two semantic types within the Semantic Network.
3.0	AssociativeRelExistence	Container class for information describing whether a particular associative relation between two semantic types exists. This class can interpret the XML returned from the existsAssociativeRelation method of the KSSRetrieverV3_0 class.
3.0	HierRelRelExistence	Container class for information describing whether a particular hierarchical relationship between two semantic relations exists. This class can interpret the XML returned from the existsHierRelRelation method of the KSSRetrieverV3_0 class.

### 3.5.16 Package gov.nih.nlm.kss.models.lex

This package contains classes that represent the SPECIALIST Lexicon lexical records. The table below lists the classes and their descriptions for the gov.nih.nlm.kss.models.lex package.

**UMLSKS Class  
Version****Description**

UMLSKS Version	Class	Description
4.0	LexicalRecordVector	Subclass of Vector that holds LexicalRecord instances and can interpret the XML returned from the getLexicalRecords method of the KSSRetrieverV4_0 class.
4.0	LexicalRecord	Container class for information about the lexical records for a given term. This class subclasses the gov.nih.nlm.nls.lexCheck.Lib.LexRecord class.



## 4 Using the XML Query Facility

The UMLS SKS API provides an extensive query facility for developers expert in the layout of the UMLS data within the UMLS SKS and its corresponding database tables. The facility utilizes the power of the eXtensible Markup Language (XML) to request the return of any data found within the UMLS data set. A special language was developed for the UMLS SKS to allow users to develop an XML query that may be issued to the UMLS SKS for processing. This chapter describes this process.

### 4.1 Retrieving Metathesaurus and Semantic Network Data using an XML Query

Retrieving Metathesaurus data using the XML query feature of the UMLS SKS requires an understanding of the UMLS structures, the UMLS SKS internal database design and the theory behind querying a relational database. The returned XML document from the query uses Oracle's XML generation facility to generate the XML, so the XML will not look like the XML generated by the other UMLS SKS API methods. The XML tags for the columns will be the names as they appear in Oracle (i.e. the field names shown in Section 4.3 will be the tags for the generated XML). In addition, the XML returned using this scheme of data retrieval cannot be used in conjunction with the object model classes.

Developers create a string containing an XML document that is then sent via the **query** method of the `KSSRetriever` class. This query must be of the form described in Section 4.2. The return value to the method is an XML document that may be parsed with a standard XML parser and its details may be extracted.

### 4.2 Structure of an XML Query

A UMLS SKS query is an XML document that describes a request for some subset of data from the Metathesaurus. The default namespace is used for all elements in the XML definition. The elements within a query document are described below using regular expression syntax. Sample queries are also provided in Section 0.

Formatting of the regular expressions describing the XML grammar following these conventions:

- Items in **bold** are ASCII characters that should appear exactly as shown in bold.
- The '|' character indicates an 'or' relationship. For example, the *fieldSetGroup* item can be either a *fieldSet* followed by a *fieldSetGroup*, or a single *fieldSet*
- $\epsilon$  denotes an empty value (i.e. may not be present)

```
query-><query version=1.0> release rowset row distinct fields
      constraints ordering </query>
```

```
release-><release/>string<release/> |  $\epsilon$ 
rowset -><rowset>string</rowset>
row -><row>string</row>
distinct-><distinct/> |  $\epsilon$ 
fields-><fields> allFields </fields> | <fields> fieldSetGroup </fields>
allFields-><all> tableSpec </all>
tableSpec -> tableNameSpec tableSpec | tableNameSpec
```

```

fieldSetGroup-> fieldSet fieldSetGroup | fieldSet
fieldSet-><fieldSet> tableNameSpec fieldPart </fieldSet>
tableNameSpec-><table> tableName </table>
tableName->mrcon | mrso | mrdef | mrsty |
           mrrel | mratx | mrcxt | mrcoc | mrsat | mrlo |
           mrxns_eng | mrxnw_eng
fieldPart-><fieldNames> fieldNameSpec </fieldNames> fieldPart |
           <fieldNames> fieldNameSpec </fieldNames>

constraints-><constraints> constraintList </constraints> | ε
constraintList->relation constraintList |
               constraint constraintList | relation | constraint
relation-><relation> relation_op constraintList </relation>
relation_op-><operator> op </operator>
op-> AND | OR | NOT

constraint-><constraint> const </constraint>
const->lhs | lhs const_op rhs
lhs-><lhs> fieldList </lhs> | <lhs> field <lhs>
fieldList-><fieldList> fieldSpec </fieldList>
fieldSpec->fieldNameSpec fieldSpec | fieldNameSpec
fieldNameSpec-><name> fieldNameSpec </name>
field-><field> tableName.fieldName </field>
fieldName->Field name for the appropriate table as described in the following sections
const_op-><op> operator </op> | normStr | normWord | word | like
operator-> oneof | noneof | = | >= | <= | > | < | !=
normStr-><normStr> string </normStr>
normWord-><normWord> string </normWord>
word-><word> string </word>
like-><like> string </like>
rhs-><rhs> query </rhs> | <rhs> stringList </rhs> |
     <rhs> stringSpec </rhs> | <rhs> field </rhs>
stringList-><stringList> strings </stringList>
strings->stringVal strings | stringVal
stringVal-><name> string </name>
stringSpec-><string> string</string>
string->set of characters that is readable by an XML parser (i.e. has the '<' described as &lt; and the '>' described
        as &gt;);

ordering-><orderBy> fieldSpec</orderBy>

```

### 4.3 Metathesaurus and Semantic Network Table Contents

The following sections describe the tables and their field names that may be used as the *<tableName>* and *<fieldName>* values described in the grammar.

#### 4.3.1 Table 'mrcon'

The 'mrcon' table for UMLS releases prior to 2004AA contains the following fields:

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LAT	Language
TS	Term Status
LUI	Lexical Unique Identifier

STT	String Type
SUI	String Unique Identifier
STR	String Value for the concept
LRL	Least Restriction Level
STR2	Normalized version of the concept's string value in all uppercase

#### 4.3.2 Table 'mrso'

The 'mrso' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LUI	Lexical Unique Identifier
SUI	String Unique Identifier
SAB	Source Abbreviation
TTY	Term Type
SCD	Unique identifier or code for string in that source
SRL	Source Restriction Level

#### 4.3.3 Table 'mrdef'

The 'mrdef' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
SAB	Source abbreviation
DEF	Definition

#### 4.3.4 Table 'mrsty'

The 'mrsty' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
TUI	Unique Identifier of the Semantic type
STY	Semantic type

#### 4.3.5 Table 'mrrel'

The 'mrrel' table for UMLS releases **prior** to 2004AA contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI1	Concept Unique Identifier 1
REL	Relationship of second concept to the first
CUI2	Concept Unique Identifier 2
RELA	Relationship attribute
SAB	Abbreviation of the source of the relationship
SL	Source of relationship labels
MG	Machine-generated and unverified indicator

The 'mrrel' table for UMLS release **2004AA** and later releases contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI1	Concept Unique Identifier 1
AUI1	Atomic unique identifier 1
STYPE1	Source asserted type for identifier 1
REL	Relationship of second concept to the first
CUI2	Concept Unique Identifier 2
AUI2	Atomic unique identifier 2
STYPE2	Source asserted type for identifier 2
RELA	Relationship attribute
RUI	Relationship unique identifier
SRUI	Source attributed relationship identifier
SAB	Abbreviation of the source of the relationship
SL	Source of relationship labels
RG	Relationship Group
DIR	Source asserted directionality flag
SUPPRESS	Suppressible flag
CVF	Content view flag

#### 4.3.6 Table 'mrmtx'

The 'mrmtx' table for UMLS releases **prior** to 2004AA contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
SAB	Abbreviation of source of terms in expression
REL	Relationship of meaning of expression to main concept
ATX	Associated expression

#### 4.3.7 Table 'mrcxt'

The 'mrcxt' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
SUI	String Unique Identifier
SAB	Source abbreviation
SCD	Unique identifier or code for string in that source
CXN	The context number (to distinguish multiple contexts in the same source with the same SUI)
CXL	Context member label (i.e. ANC for ancestor of this concept, CCP for concept, SIB for sibling of this concept, CHD for child of this concept).
RNK	For rows with a CXL value of ANC, the rank of the ancestors (e.g. a value of 1 denotes the most remote ancestor in the hierarchy)
CXS	String for context member

CUI2	Unique concept identifier of context member
HCD	Hierarchical number or code of context member in this source
RELA	Relationship attribute providing further categorization of the CXL
XC	A plus (+) sign indicates that the CUI2 for this row has children in this context

The 'mrcxt' table for UMLS release **2004AA** and later releases contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
SUI	String Unique Identifier
AUI1	Atomic unique identifier for this atom
SAB	Source abbreviation
CODE	Unique identifier or code for string in that source
CXN	The context number (to distinguish multiple contexts in the same source with the same SUI)
CXL	Context member label (i.e. ANC for ancestor of this concept, CCP for concept, SIB for sibling of this concept, CHD for child of this concept).
RNK	For rows with a CXL value of ANC, the rank of the ancestors (e.g. a value of 1 denotes the most remote ancestor in the hierarchy)
CXS	String for context member
CUI2	Unique concept identifier of context member
AUI2	Atomic unique identifier for second atom
HCD	Hierarchical number or code of context member in this source
RELA	Relationship attribute providing further categorization of the CXL
XC	A plus (+) sign indicates that the CUI2 for this row has children in this context
CVF	Content view flag

#### 4.3.8 Table 'mrcoc'

The 'mrcoc' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI1	Concept Unique Identifier 1
CUI2	Concept Unique Identifier 2
SOC	Abbreviation of the source of the co-occurrence information
COT	Co-occurrence type
COF	Co-occurrence frequency
COA	Attributes of the co-occurrence

The 'mrcoc' table for UMLS release **2004AA** and later releases contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI1	Concept Unique Identifier 1
AUI1	Atomic unique identifier 2
CUI2	Concept Unique Identifier 2
AUI2	Atomic unique identifier 2
SAB	Abbreviation of the source of the co-occurrence information
COT	Co-occurrence type
COF	Co-occurrence frequency
COA	Attributes of the co-occurrence
CVF	Content View Flag

#### 4.3.9 Table 'mrsat'

The 'mrsat' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LUI	Term Unique Identifier
SUI	String Unique Identifier
SCD	Unique identifier or code for entry in the source of the attribute
ATN	Attribute name
SAB	Abbreviation of the source of the attribute
ATV	Attribute value

The 'mrsat' table for UMLS release **2004AA** and later releases contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LUI	Term Unique Identifier
SUI	String Unique Identifier
UI	Atom or Relationship identifier
STYPE	Source asserted type for the identifier in CODE
CODE	Unique identifier or code for entry in the source of the attribute
ATUI	Attribute unique identifier
SATUI	Source asserted attribute identifier
ATN	Attribute name
SAB	Abbreviation of the source of the attribute
ATV	Attribute value
SUPPRESS	Suppressible flag
CVF	Content View Flag

**4.3.10 Table 'mrlo'**

The 'mrlo' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
ISN	Name of information source or database in which concept appears
FR	Frequency count of number of occurrences of concept in the information source (optional)
UN	Meaning of the frequency
SUI	Unique identifier of the string if name used in information source appears in MRCON
SNA	Actual name that occurs in the information source if not otherwise present in the Metathesaurus
SOUI	Unique identifier of record in which the concept appears in the source

The 'mrlo' table for UMLS release **2004AA** and later releases contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
AUI	Atomic unique identifier
ISN	Name of information source or database in which concept appears
FR	Frequency count of number of occurrences of concept in the information source (optional)
UN	Meaning of the frequency
SUI	Unique identifier of the string if name used in information source appears in MRCON
SNA	Actual name that occurs in the information source if not otherwise present in the Metathesaurus
SOUI	Unique identifier of record in which the concept appears in the source
CVF	Content View Flag

**4.3.11 Table 'mrnxns\_eng'**

The 'mrnxns\_eng' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LUI	Lexical Unique Identifier
SUI	Unique identifier of the string if name used in information source appears in MRCON
LAT	Language
NSTR	Normalized string value

NSTR2	Normalized string value with spaces removed and all letters capitalized
-------	---

#### 4.3.12 Table 'mrxnw\_eng'

The 'mrxnw\_eng' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LUI	Lexical Unique Identifier
SUI	Unique identifier of the string if name used in information source appears in MRCON
LAT	Language
NWD	Normalized word value

#### 4.3.13 Table 'mrwx\_eng'

The 'mrwx\_eng' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LUI	Lexical Unique Identifier
SUI	Unique identifier of the string if name used in information source appears in MRCON
LAT	Language
WD	Normalized word value

#### 4.3.14 Table 'mrcui'

The 'mrcui' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
YEAR	UMLS Release
FLAG	Indicator for whether the concept was deleted (value of DEL) or merged (value of SY)
MCUI	When the flag indicates merge, this contains the CUI into which the CUI was merged

The 'mrcui' table for UMLS release **2004AA** and later releases contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
YEAR	UMLS Release
REL	Relationship
RELA	Relationship attribute
MAPREASON	Reason for mapping
CUI2	Unique identifier for the second concept
MAPIN	Indicates presence in subset

**4.3.15 Table ‘mrsab’**

The ‘mrsab’ table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
VCUI	CUI for the versioned source
RCUI	CUI for the root source
SAB	Abbreviation of the root source
STR	Official name for the source
STR2	Abbreviation of the versioned source
SF	Source family
VER	Version indicator
MSTART	Date at which the source first appeared in the Metathesaurus
MEND	Date at which this source disappeared from the Metathesaurus
IMETA	Metathesaurus release in which this source first appeared
RMETA	Metathesaurus release in which this source disappeared
SLC	Contact information for obtaining the source's license
SCC	Contact information for information about the source's content
SRL	Source restriction level
TFR	Number of terms for this source in MRCON/MRSO
CFR	Number of CUIs associated with this source
CXTY	The type of context for this source from the UMLS documentation
TTYL	Term-type list from the source
ATNL	Attribute name list from the source
LAT	Language indicator
CENC	Character encoding value
CURVER	Current Version Flag (Y/N)
SABIN	Source in Metamorphosys subsystem flag (Y/N)
SSN	Field added in <b>2004AA</b> and later UMLS releases that indicates the source's short name

**4.3.16 Table ‘mshqual’**

The ‘mshqual’ table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CODE	MeSH Qualifier Code (2-3 characters)
NAME	MeSH Qualifier description

**4.3.17 Table ‘strattr’**

The ‘strattr’ table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
ATN	Attribute name
ATEN	Attribute description

**4.3.18 Table 'ttys'**

The 'ttys' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
TTY	Term type
STR	Description of the term type

**4.3.19 Table 'cots'**

The 'cots' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
COT	Cooccurrence type (2-3 characters)
STR	Cooccurrence type description

**4.3.20 Table 'stts'**

The 'stts' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
STT	String type
STR	Description of the string type

**4.3.21 Table 'rels'**

The 'rels' table for UMLS releases prior to **2004AA** contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
REL	Relation type
STR	Description of the relation type

**4.3.22 Table 'srdef'**

The 'srdef' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
RT	Record type (STY=Semantic Type or RL=Relation)
UI	Unique identifier of the semantic type or semantic relation
NAME	Name of the semantic type or semantic relation
TN	Tree number of the semantic type or semantic relation
DEF	Definition of the semantic type or relation
EX	Examples of Metathesaurus concepts with this semantic type (STY records only)
UN	Usage note for semantic type assignment (STY records only)
NH	Semantic type and its descendants allow the non-human flag (STY records only)
ABBREV	Abbreviation of the semantic relation name (RL records only)

RIN Inverse of the semantic relation (RL records only)

#### 4.3.23 Table 'srgrp'

The 'srgrp' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
ABBREV	Abbreviation for the semantic group
GRP	Group name
STNAME	Semantic type name belonging to the semantic group
STUI	Unique identifier of the semantic type belonging to the semantic group

#### 4.3.24 Table 'srstr'

The 'srstr' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
SRLEFT	Argument 1 (name of a semantic type or semantic relation)
REL	Relation ("isa" or the name of a non-hierarchical semantic relation)
SRRIGHT	Argument 2 (name of a semantic type or semantic relation); if this field is blank this means that the semantic type or semantic relation is one of the top nodes of the Network.
LS	Link Status (D = Defined for the arguments and its children; B = Blocked; DNI = Defined but Not Inherited by the children of the arguments) The relations expressed in this table are binary relations and the arguments are ordered pairs. The relations are stated only for the top-most node of the "isa" hierarchy of the semantic types to which they may apply.

#### 4.3.25 Table 'srstre1'

The 'srstre1' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
LTUI	Argument 1 semantic type unique identifier
REL	Semantic relation unique identifier
RTUI	Argument 2 semantic type unique identifier

#### 4.3.26 Table 'srstre2'

The 'srstre2' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
SRLEFT	Argument 1 semantic type name
REL	Semantic relation name
SRRIGHT	Argument 2 semantic type name

**4.3.27 Table 'mrconso'**

The 'mrconso' table for UMLS release **2004AA** and later contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept Unique Identifier
LAT	Language
TS	Term Status
LUI	Lexical Unique Identifier
STT	String Type
SUI	String Unique Identifier
ISPREF	Indicates preferred term for the language indicated for the row
AUI	Atomic unique identifier
SAUI	Source asserted unique identifier
SCUI	
SDUI	
SAB	Source abbreviation
TTY	Term type
CODE	Unique identifier or code for the string in the source
STR	String Value for the concept
SRL	Least Restriction Level
SUPPRESS	Suppressible flag
CVF	Content View Flag
STR2	Normalized version of the concept's string value in all uppercase

**4.3.28 Table 'mrdoc'**

The 'mrdoc' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
TYPE	Type of the meta data
MKEY	Key to the type of meta data
SUBKEY	Key to the abbreviation for the entry of the MKEY type
VALUE	Value for the SUBKEY

**4.3.29 Table 'mrhier'**

The 'mrhier' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept unique identifier
AUI	Atomic unique identifier
CXN	
PAUI	
SAB	
RELA	
PTR	
HCD	
CVF	Content view flag

**4.3.30 Table 'mrhist'**

The 'mrhier' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
CUI	Concept unique identifier
UI	
SAB	
VER	
CHANGETYPE	
CHANGEKEY	
CHANGEVAL	
REASON	

**4.3.31 Table 'mrmmap'**

The 'mrmmap' table contains the following fields that may be specified as

<u>Field Name</u>	<u>Description</u>
MAPSETCUI	
MAPSETSAB	
MAPSUBSETID	
MAPRANK	
FROMID	
FROMSID	
FROMEXPR	
FROMTYPE	
FROMRULE	
FROMRES	
REL	
RELA	
TOID	
TOSID	
TOEXPR	
TOTYPE	
TORULE	
TORES	
MAPRULE	
MAPTYPE	
MAPATN	
MAPATV	
CVF	Content View Flag

**4.4 XML Query Example**

A number of sample queries are provided with the API delivery, including queries for all of the convenience API functions describe previously. The XML query that returns the details of a concept, similar to the `getConcept` API method, is described here.

The following example query requests the return of all the fields for UMLS release 2003AC in the "mrcon", "mrso", "mrsty", and "mrcxt" tables where the concept unique identifier is 'C0024109'.

```

<?xml version="1.0"?>
<query xmlns="http://umlsks4.nlm.nih.gov"
      xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      version="1.0">

  <release>2003AC</release>
  <distinct/>

  <fields>
    <all>
      <table>mrcon</table>
      <table>mrso</table>
      <table>mrsty</table>
      <table>mrcxt</table>
    </all>
  </fields>

  <constraints>
    <relation>
      <operator>AND</operator>
      <constraint>
        <lhs><field>mrcon.cui</field></lhs>
        <op>=</op>
        <rhs><string>C0024109</string></rhs>
      </constraint>
      <constraint>
        <lhs><field>mrcon.ts</field></lhs>
        <op>!</op>
        <rhs><string>s</string></rhs>
      </constraint>
      <constraint>
        <lhs><field>mrcon.cui</field></lhs>
        <op>=</op>
        <rhs><field>mrso.cui</field></rhs>
      </constraint>
      <constraint>
        <lhs><field>mrcon.lui</field></lhs>
        <op>=</op>
        <rhs><field>mrso.lui</field></rhs>
      </constraint>
      <constraint>
        <lhs><field>mrcon.sui</field></lhs>
        <op>=</op>
        <rhs><field>mrso.sui</field></rhs>
      </constraint>
      <constraint>
        <lhs><field>mrcon.cui</field></lhs>
        <op>=</op>
        <rhs><field>mrsty.cui</field></rhs>
      </constraint>
      <constraint>
        <lhs><field>mrcon.cui</field></lhs>
        <op>=</op>
        <rhs><field>mrcxt.cui</field></rhs>
      </constraint>
    </relation>
  </constraints>

```

```
<orderBy>
  <name>mrcon.cui</name>
  <name>mrcon.sui</name>
  <name>mrso.sab</name>
  <name>mrcxt.scd</name>
  <name>mrcxt.cxn</name>
  <name>mrcxt.rnk</name>
</orderBy>
</query>
```



## 5 Using the UMLSKS Socket Server

The UMLSKS provides an interface to client programs through a TCP/IP socket. Clients using this interface send XML requests through the socket to the UMLSKS server, which in turn executes the request and returns the result of the query in its XML form. This chapter describes the acceptable XML commands that may be issued through the socket server and details the configuration of a client using the socket server.

### 5.1 Connecting to the UMLSKS Socket Server

The UMLSKS Socket Server is a TCP/IP server that is running on the machine **'umlks.nlm.nih.gov'** at port **8042**. Connect to this socket and issue XML commands to the UMLSKS and receive XML data results.

The Socket Server can accept requests to execute API commands in the form of an XML query. The following sections describe the structures of each of the queries accepted by this version of the UMLSKS. Each query must be ended with the string **%%** and a new line character.

### 5.2 General Queries

#### 5.2.1 XML Query getCurrentUMLSVersion

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getCurrentUMLSVersion version="1.0"/>
```

#### 5.2.2 XML Query getUMLSVersions

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getUMLSVersions version="1.0"/>
```

#### 5.2.3 XML Query getSWVersion

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getSWVersion version="1.0"/>
```

#### 5.2.4 XML Query query

The XML query grammar for this function is described in Chapter 4.

#### 5.2.5 XML Query describeCurrentUMLSVersion

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<describeCurrentUMLSVersion version="1.0"/>
```

#### 5.2.6 XML Query describeUMLSVersions

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<describeUMLSVersions version="1.0"/>
```

### 5.2.7 XML Query listDocEntryTypes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listDocEntryTypes version="1.0">
  <docEntryType>DOC_ENTRY_TYPE</docEntryType>
  <docEntrySubType>DOC_ENTRY_SUB_TYPE</docEntrySubType>
</listDocEntryTypes>
```

## 5.3 Metathesaurus Queries

### 5.3.1 XML Query listDictionaries

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listDictionaries version="1.0"/>
```

### 5.3.2 XML Query suggestSpelling

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<suggestSpelling version="1.0">
  <dictionary>DICTIONARY</dictionary>
  <term>TERM_NAME</term>
</suggestSpelling>
```

where:

<i>DICTIONARY</i>	is the dictionary to be used to suggest the spelling. This dictionary is one of the dictionary names returned from the query <i>listDictionaries</i> .
<i>TERM_NAME</i>	is the term to be spell checked.

### 5.3.3 XML Query getConceptName

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getConceptName version="1.0">
  <release>RELEASE</release>
  <cu>CXXXXXXX</cu>
  <sablist>
    SABLST
  </sablist>
  <language>LANGUAGE</language>
</getConceptName>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXX</i>	is the concept unique identifier (CUI)

<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the language restriction

### 5.3.4 XML Query findCUI

The XML query grammar for this function follows one of the following structures:

```
<?xml version="1.0"?>
<findCUI version="1.0">
  <release>RELEASE</release>
  <conceptName>CONCEPT_NAME</conceptName>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  MATCHING_TECHNIQUE
  <noSuppressibles/>
</findCUI>
```

-or-

```
<?xml version="1.0"?>
<findCUI version="1.0">
  <release>RELEASE</release>
  <semtype>SEMANTIC_TYPE</semtype>
</findCUI>
```

-or-

```
<?xml version="1.0"?>
<findCUI version="1.0">
  <release>RELEASE</release>
  <sab>SAB</sab>
  <scd>SCD</scd>
</findCUI>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CONCEPT NAME</i>	is the concept name
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the language restriction

<i>MATCHING_TECHNIQUE</i>	is one of the following: <code>&lt;normStr/&gt;</code> <code>&lt;normWord/&gt;</code> <code>&lt;word/&gt;</code> <code>&lt;exact/&gt;</code> <code>&lt;truncRight/&gt;</code> <code>&lt;truncLeft/&gt;</code> <code>&lt;approx/&gt;</code>
<code>&lt;noSuppressibles/&gt;</code>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.
<i>SEMANTIC_TYPE</i>	is the semantic type whose concept identifiers are to be returned if the second form is used.
<i>SAB</i>	is the source abbreviation of interest if the third form is used.
<i>SCD</i>	is the unique code in the source if the third form is used.

### 5.3.5 XML Query `getConceptProperties`

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<getConceptProperties version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/>
</getConceptProperties>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <code>&lt;dbyear&gt;</code> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SABLIST</i>	is the optional list of sources of the form:  <code>&lt;sab&gt; SAB &lt;/sab&gt;</code>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<code>&lt;noSuppressibles/&gt;</code>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.6 XML Query `findConcept`

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<findConcept version="1.0">
  <release>RELEASE</release>
  <conceptName>CONCEPT_NAME</conceptName>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  MATCHING_TECHNIQUE
  <noSuppressibles/>
</findConcept>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CONCEPT NAME</i>	is the concept name
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<i>MATCHING_TECHNIQUE</i>	is one of the following: <b>&lt;normStr/&gt;</b> <b>&lt;normWord/&gt;</b> <b>&lt;word/&gt;</b> <b>&lt;exact/&gt;</b> <b>&lt;truncRight/&gt;</b> <b>&lt;truncLeft/&gt;</b> <b>&lt;approx/&gt;</b>
<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.7 XML Query `getBasicConceptProperties`

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<getBasicConceptProperties version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/>
</getBasicConceptProperties>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.8 XML Query findBasicConcept

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findBasicConcept version="1.0">
  <release>RELEASE</release>
  <conceptName>CONCEPT_NAME</conceptName>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  MATCHING_TECHNIQUE
  <noSuppressibles/ >
</findBasicConcept>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CONCEPT_NAME</i>	is the concept name
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<i>MATCHING_TECHNIQUE</i>	is one of the following: <b>&lt;normStr/&gt;</b> <b>&lt;normWord/&gt;</b> <b>&lt;word/&gt;</b> <b>&lt;exact/&gt;</b> <b>&lt;truncRight/&gt;</b> <b>&lt;truncLeft/&gt;</b> <b>&lt;approx/&gt;</b>

<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.
---------------------------------	--

### 5.3.9 XML Query getTerminology

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<getTerminology version="1.0">
  <release>RELEASE</release>
  <cu>CXXXXXXXX</cu>
  <sablist>
    SABLST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/ >
</getTerminology>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SABLST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.10 XML Query getTerms

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<getTerms version="1.0">
  <release>RELEASE</release>
  <term>TERM_NAME</term>
  <sablist>
    SABLST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/>
</getTerms>

```

-or-

```

<?xml version="1.0"?>
<findTerms version="1.0">
  <release>RELEASE</release>
  <term>TERM_NAME</term>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/>
</findTerms>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>TERM NAME</i>	is the term name
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.11 XML Query getDefinition

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<getDefinition version="1.0">
  <release>RELEASE</release>
  <cu>CXXXXXXX</cu>
  <sablist>
    SABLIST
  </sablist>
</getDefinition>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation

### 5.3.12 XML Query getSemanticType

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getSemanticType version="1.0">
  <release>RELEASE</release>
  <cu>CXXXXXXXX</cu>
</getSemanticType>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)

### 5.3.13 XML Query getContext

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getContext version="1.0">
  <release>RELEASE</release>
  <cu>CXXXXXXXX</cu>
  <sablist>
    SABLST
  </sablist>
</getContext>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SABLST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation

### 5.3.14 XML Query getAssocExprs

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getAssocExprs version="1.0">
  <release>RELEASE</release>
  <cu>CXXXXXXXX</cu>
  <sab>SAB</sab>
</getAssocExprs>
```

-or-

```

<?xml version="1.0"?>
<getAssocExprs version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sablist>
    SABLIST
  </sablist>
</getAssocExprs>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SAB</i>	is the source abbreviation
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation

### 5.3.15 XML Query getCooccurrences

The XML query grammar for this function is:

```

<?xml version="1.0"?>
<getCooccurrences version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sab>SAB</sab>
  <cot>COT</cot>
</getCooccurrences>

```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SAB</i>	is the source abbreviation
<i>COT</i>	is the co-occurrence type

### 5.3.16 XML Query getRelations

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getRelations version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <rel>RELATION</rel>
</getRelations>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<i>RELATION</i>	is the optional relationship name

### 5.3.17 XML Query getStringAttributes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getStringAttributes version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sab>SAB</sab>
  <language>LANGUAGE</language>
  <attrName>ATTRIBUTE_NAME</attrName>
</getStringAttributes>
```

-or-

```
<?xml version="1.0"?>
<getStringAttributes version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXXX</cui>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <attrName>ATTRIBUTE_NAME</attrName>
</getStringAttributes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later
----------------	---

	versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXX</i>	is the concept unique identifier (CUI)
<i>SAB</i>	is the source abbreviation
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<i>ATTRIBUTE_NAME</i>	is the optional attribute name

### 5.3.18 XML Query getLocator –or- getLocators

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getLocator version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXX</cui>
</getLocator>
```

-or-

```
<?xml version="1.0"?>
<getLocators version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXX</cui>
</getLocators>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXX</i>	is the concept unique identifier (CUI)

### 5.3.19 XML Query getMeSHEntries

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getMeSHEntries version="1.0">
  <release>RELEASE</release>
  <term>TERM_NAME</term>
</getMeSHEntries>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>TERM_NAME</i>	is the term name

### 5.3.20 XML Query getMeSHInfo

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getMeSHInfo version="1.0">
  <release>RELEASE</release>
  <dui>DUI</dui>
</getMeSHInfo>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>DUI</i>	is the unique identifier

### 5.3.21 XML Query getRawRecords

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getRawRecords version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXX</cui>
  <language>LANGUAGE</language>
  <tableName>TABLE_NAME</tableName>
</getRawRecords>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXX</i>	is the concept unique identifier (CUI)
<i>LANGUAGE</i>	is the optional language restriction
<i>TABLE NAME</i>	is the name of the table to be returned

#### 5.3.21.1 XML Query query

The XML query grammar for this function is described in Chapter 4.

### 5.3.22 XML Query describeSource

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<describeSource version="1.0">
  <release>RELEASE</release>
  <sab>SAB</sab>
</describeSource>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest. If not specified, then the current UMLS release is used. Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SAB</i>	is the source abbreviation

### 5.3.23 XML Query listSources

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listSources version="1.0">
  <release>RELEASE</release>
</listSources>
```

where:

<i>RELEASE</i>	is the optional UMLS of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

### 5.3.24 XML Query findLUI

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findLUI version="1.0">
  <release>RELEASE</release>
  <termName>TERM_NAME</termName>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/>
</findLUI>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>TERM NAME</i>	is the term name
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.25 XML Query getTermsForLUI

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getTermsForLUI version="1.0">
  <release>RELEASE</release>
  <lui>LUI</lui>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
  <noSuppressibles/>
</getTermsForLUI>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>LUI</i>	is the lexical unique identifier (LUI)
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction
<b>&lt;noSuppressibles/&gt;</b>	is an optional field. If specified, then the suppressible synonyms will be included in the search for a matching term.

### 5.3.26 XML Query findSUI

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findSUI version="1.0">
  <release>RELEASE</release>
  <string>STRING</string>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
</findSUI>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>STRING</i>	is the string to be mapped

<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction

### 5.3.27 XML Query `getStringsForSUI`

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getStringsForSUI version="1.0">
  <release>RELEASE</release>
  <sui>SUI</sui>
  <sablist>
    SABLIST
  </sablist>
  <language>LANGUAGE</language>
</getStringsForSUI>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SUI</i>	is the string unique identifier (SUI)
<i>SABLIST</i>	is the optional list of sources of the form:  <b>&lt;sab&gt; SAB &lt;/sab&gt;</b>  where <i>SAB</i> is the source abbreviation
<i>LANGUAGE</i>	is the optional language restriction

### 5.3.28 XML Query `listStrAttrs`

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listStrAttrs version="1.0">
  <release>RELEASE</release>
</listStrAttrs>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

### 5.3.29 XML Query `listMeSHQuals`

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listMeSHQuals version="1.0">
  <release>RELEASE</release>
</listMeSHQuals>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLS SKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

### 5.3.30 XML Query describeUMLSChanges

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<describeUMLSChanges version="1.0">
  <release>RELEASE</release>
  ...
  <release>RELEASE-N</release>
  <cui>CUI</sablister>
  ...
  <cui>CUI-N</cui>
</describeUMLSChanges>
```

-or-

```
<?xml version="1.0"?>
<describeUMLSChanges version="1.0">
  <umlsRelease>UMLS_RELEASE</umlsRelease>
  <release>RELEASE</release>
  ...
  <release>RELEASE-N</release>
  <cui>CUI</sablister>
  ...
  <cui>CUI-N</cui>
</describeUMLSChanges>
```

where:

<i>UMLS_RELEASE</i>	is the optional identifier for the version of the UMLS whose changes are to be returned. If this value is not specified, then the current release's changes are described.
<i>RELEASE, RELEASE-N</i>	is the optional set of UMLS releases of interest. If not specified, all releases are requested.
<i>CUI, CUI-N</i>	is the optional concept unique identifier of interest. If not specified, all CUIs are requested.

### 5.3.31 XML Query listTermTypes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listTermTypes version="1.0">
  <release>RELEASE</release>
</listTermTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

### 5.3.32 XML Query listCooccurrenceTypes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listCooccurrenceTypes version="1.0">
  <release>RELEASE</release>
</listCooccurrenceTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

### 5.3.33 XML Query listStringTypes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listStringTypes version="1.0">
  <release>RELEASE</release>
</listStringTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.12.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

### 5.3.34 XML Query listRelationTypes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.

### 5.3.35 XML Query listMetaTableNames

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listMetaTableNames version="1.0">
  <release>RELEASE</release>
</listMetaTableNames>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	---

### 5.3.36 XML Query listRelationTypes

The XML query grammar for this function is:

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	--

## 5.4 Semantic Network Queries

### 5.4.1 XML Query findSemType

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findSemType version="1.0">
  <release>RELEASE</release>
  <contains>STRING</contains>
  <expandTree/>
</findSemType>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>STRING</i>	is the required string that the returned semantic type(s) should contain in its name
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic type(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.2 XML Query getSemTypeProperties

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getSemTypeProperties version="1.0">
  <release>RELEASE</release>
  <semtype>SEMTYPE</semtype>
  <expandTree/>
</getSemTypeProperties>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SEMTYPE</i>	is the required name/unique identifier for the semantic type
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic type(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.3 XML Query getSemTypeAncestors

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getSemTypeAncestors version="1.0">
  <release>RELEASE</release>
  <semtype>SEMTYPE</semtype>
  <expandTree/>
</getSemTypeAncestors>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SEMTYPE</i>	is the required name/unique identifier for the semantic type
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic type(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.4 XML Query getSemTypeSiblings

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <semtype>SEMTYPE</semtype>
  <expandTree/>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SEMTYPE</i>	is the required name/unique identifier for the semantic type
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic type(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.5 XML Query listSemTypeIds

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listSemTypeIds version="1.0">
  <release>RELEASE</release>
</listSemTypeIds>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	---

### 5.4.6 XML Query findSemRelation

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findSemRelation version="1.0">
  <release>RELEASE</release>
  <contains>STRING</contains>
  <expandTree/>
</findSemRelation>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>STRING</i>	is the required string that the returned semantic relation(s) should contain in its name
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic relation(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.7 XML Query getSemRelationProperties

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <semrel>SEMREL</semrel>
  <expandTree/>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SEMREL</i>	is the required name/unique identifier for the semantic relation
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic relation(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.8 XML Query getSemRelationAncestors

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <semrel>SEMREL</semrel>
  <expandTree/>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SEMREL</i>	is the required name/unique identifier for the semantic relation
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic relation(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.9 XML Query listSemRelationIds

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listSemRelationIds version="1.0">
  <release>RELEASE</release>
</listSemRelationIds>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	---

### 5.4.10 XML Query existsAssociativeRelation

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <lhs_semtype>LHS_SEMTYPE</lhs_semtype>
  <semrel>SEMREL</semrel>
  <rhs_semtype>RHS_SEMTYPE</rhs_semtype>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>LHS_SEMTYPE</i>	is the required name/unique identifier for the semantic type appearing on the left hand side of the relationship
<i>SEMREL</i>	is the required name/unique identifier for the semantic relation
<i>RHS_SEMTYPE</i>	is the required name/unique identifier for the semantic type appearing on the right hand side of the relationship

### 5.4.11 XML Query getAssociativeRelations

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <lhs_semtype>LHS_SEMTYPE</lhs_semtype>
  <semrel>SEMREL</semrel>
  <rhs_semtype>RHS_SEMTYPE</rhs_semtype>
  <expandTree/>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>LHS_SEMTYPE</i>	is the required name/unique identifier for the semantic type

	appearing on the left hand side of the relationship
<i>SEMREL</i>	is the required name/unique identifier for the semantic relation
<i>RHS_SEMTYPE</i>	is the required name/unique identifier for the semantic type appearing on the right hand side of the relationship
<b>&lt;expandTree/&gt;</b>	is an optional field. If specified, then the entire hierarchy for the semantic types and semantic relation will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.12 XML Query existsHierRelRelRelation

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <lhs_semrel>LHS_SEMREL</lhs_semrel>
  <semrel>SEMREL</semrel>
  <rhs_semrel>RHS_SEMREL</rhs_semrel>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>LHS_SEMREL</i>	is the required name/unique identifier for the semantic relation appearing on the left hand side of the relationship
<i>SEMREL</i>	is the required name/unique identifier for the semantic relation
<i>RHS_SEMREL</i>	is the required name/unique identifier for the semantic relation appearing on the right hand side of the relationship

### 5.4.13 XML Query listSemGroups

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listSemGroups version="1.0">
  <release>RELEASE</release>
</listSemGroups>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
----------------	---

### 5.4.14 XML Query listSemTypes

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listSemTypes version="1.0">
  <release>RELEASE</release>
  <semgroup>SEMGROUP</semgroup>
</listSemTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>SEMGROUP</i>	is the required name for the semantic group

### 5.4.15 XML Query getSemGroup

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listRelationTypes version="1.0">
  <release>RELEASE</release>
  <cui>CXXXXXXX</cui>
</listRelationTypes>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>CXXXXXXX</i>	is the concept unique identifier (CUI) whose semantic group is to be returned.

### 5.4.16 XML Query findBasicSemType

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findBasicSemType version="1.0">
  <release>RELEASE</release>
  <contains>STRING</contains>
  <expandTree/>
</findBasicSemType>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>STRING</i>	is the required string that the returned semantic type(s) should contain in its name

<code>&lt;expandTree/&gt;</code>	is an optional field. If specified, then the entire hierarchy for the semantic type(s) will be returned, including all ancestors, siblings, and children in expanded form.
----------------------------------	--

### 5.4.17 XML Query `findBasicSemRelation`

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<findBasicSemRelation version="1.0">
  <release>RELEASE</release>
  <contains>STRING</contains>
  <expandTree/>
</findBasicSemRelation>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <code>&lt;dbyear&gt;</code> as the release specifier tag.
<i>STRING</i>	is the required string that the returned semantic relation(s) should contain in its name
<code>&lt;expandTree/&gt;</code>	is an optional field. If specified, then the entire hierarchy for the semantic relation(s) will be returned, including all ancestors, siblings, and children in expanded form.

### 5.4.18 XML Query `listSemNetTableNames`

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<listSemNetTableNames version="1.0">
  <release>RELEASE</release>
</listSemNetTableNames>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <code>&lt;dbyear&gt;</code> as the release specifier tag.
----------------	---

## 5.5 *SPECIALIST Lexicon Queries*

### 5.5.1 XML Query `getLexicalRecords`

The XML query grammar for this function is:

```
<?xml version="1.0"?>
<getLexicalRecords version="1.0">
  <release>RELEASE</release>
  <term>TERM_NAME</term>
</getLexicalRecords>
```

where:

<i>RELEASE</i>	is the optional UMLS release of interest (defaults to the current release if not specified). Version 2.1 and later versions of the UMLSKS will continue to accept the tag <b>&lt;dbyear&gt;</b> as the release specifier tag.
<i>TERM</i>	is the term whose lexical records are to be returned.