

CACORE SIW AND UML MODEL LOADER

Version 4.1 - User's Guide



**NATIONAL[®]
CANCER
INSTITUTE**

Center for Biomedical Informatics and Information Technology

TABLE OF CONTENTS

About This Guide	1
Purpose	1
Audience	2
Topics Covered	2
Additional References	3
Text Conventions Used	4
Credits and Resources	5
 Chapter 1	
Overview of caCORE	7
caCORE Overview	7
Components of caCORE	8
<i>Semantic Integration Workbench (SIW)</i>	8
<i>Enterprise Vocabulary Services (EVS)</i>	9
<i>Cancer Data Standards Registry (caDSR)</i>	9
<i>UML Model Loader</i>	9
<i>Software Development Kit (SDK)</i>	9
<i>Common Security Model (CSM)</i>	9
<i>Common Logging Module (CLM)</i>	9
UML Modeling for use with caCORE	9
Modeling Constraints	10
Naming Best Practices	11
Generating the XMI File for the caCORE Process	13
 Chapter 2	
Introduction to Semantic Integration	15
Semantic Integration and Semantic Annotation	15
Terms You Should Know	16
Getting From Model to Annotation to Integration	18
Data Element/Common Data Element (DE/CDE)	20
Value Domains	21
Enumerated and Non-Enumerated Value Domains	21
Using Local Value Domains	23
<i>Local Value Domains and the UML Loader</i>	23
<i>Creating Local Value Domains</i>	24
<i>Value Meanings</i>	25
<i>Pointing a UML Attribute to a Local Value Domain</i>	26

Chapter 3

Overview of the SIW	27
Introduction	27
Pre-Requisites to Using the SIW	28
Possessing a Valid XMI File	28
Installing/Verifying Installation of Java Web Start	28
Launching the SIW	29
SIW Steps/Options	31
Review Unannotated XMI File (Model Owner)	32
Perform XMI Roundtrip (Model Owner)	32
Run Semantic Connector (Model Owner)	33
Curate XMI File (Vocabulary Reviewer)	33
Review Annotated XMI File (Model Owner)	34
Generate Default GME Tags	34
GME Cleanup	35
Package Filtering	35
Using the SIW Viewer	35
Navigation Tree	36
Detail View	37
<i>Viewing an Unannotated XMI File</i>	39
<i>Viewing an Annotated XMI File</i>	40
<i>Viewing Associations</i>	42
Errors/Log Panel	42
<i>Errors Tab</i>	42
<i>Log Tab</i>	44
Setting Viewer Preferences	44
View Associations in the Class Tree	45
Display UML Description Last	45
Automatically Search EVS on EVS Link	45
Display Primary Concept First	46
Display Inherited Attributes	46
Sort Element by Name	46
Use Pre-Thesaurus to Validate Concepts	47
Don't show warnings about mapping an inherited attribute to a CDE	47
Show GME Tags	48
Show Concept Code and Concept Name Summary for Classes and Attributes	48
Saving Changes to a File	48
Updating UML Model Definitions with SIW-Generated Changes	49

Chapter 4

Reviewing An Unannotated File51

Opening an Unannotated XMI File	51
Annotation Basics and Viewing an Unannotated File	55
UML Element Descriptions and Annotations	56
Association Annotations	57
Next Steps - RoundTrip or Semantic Connector	58

Chapter 5

Using XMI RoundTrip59

What XMI Roundtrip Mode Does	59
Running XMI Roundtrip	60
Next Steps - Review Annotated File and Semantic Connector	65

Chapter 6

Using the Semantic Connector67

What the Semantic Connector Does	67
Running the Semantic Connector	69
Next Steps - Review Annotated File and Curate XMI File	71

Chapter 7

Curating An XMI File73

What is Curating an XMI File?	73
Opening an XMI File for Curation	74
Adding EVS Concepts to an Element in Curate Mode	76
Searching EVS for Concept Information	78
Editing Concept Information in Curate Mode	81
Replacing Mapped Concepts in Curate Mode	82
Changing the Order of Mapped Concepts in Curate Mode	82
Deleting Mapped Concepts in Curate Mode	84
Editing Concept Information Without Changing the Concept Code	85
Editing Concept Information While Changing the Concept Code	85
Applying Changes to All Similar Nodes	86
Validating XMI File Concept Information Against EVS	87
Verifying the Curated XMI File	89
Next Step - Review the Annotated XMI File	90

Chapter 8

Reviewing An Annotated File91

Purpose for Reviewing an Annotated XMI File	91
Opening an Annotated XMI File for Review	93
Changing Concept Information in Review Mode	95

Adding EVS Concepts to an Element in Review Mode	96
Replacing Mapped Concepts in Review Mode	97
Changing the Order of Mapped Concepts in Review Mode	98
Deleting Mapped Concepts in Review Mode	99
Mapping UML Attributes to caDSR CDEs	99
<i>Mapping a UML Attribute to an Existing Common Data Element</i>	<i>100</i>
Mapping a UML Attribute to a Value Domain	104
Re-Verifying the Reviewed XMI File	105

Chapter 9

Using GME Annotations109

Overview of GME Annotations	109
Sample Scenarios for Using GME Options	111
Scenario One: Annotating a New Model With New Classes	112
Scenario Two: Annotating a New Model With Reused Classes	112
Scenario Three: Annotating a New Version of an Existing Model	112
Scenario Four: Making Updates to an Annotated Model	113
Generating Default GME Tags	113
Clearing GME Tags	117

Chapter 10

Understanding the UML Model Loader121

Overview of UML Loader Process	122
Submitting a UML Model to caDSR	124
Relating (Classifying) Metadata Items to UML Models	124
Classifying UML Model Metadata (Classification Schemes and Classification Scheme Items)	126
Alternate Names and Alternate Definitions	128
<i>Creating Alternate Names and Alternate Definitions for Object Classes</i>	<i>128</i>
<i>Creating an Alternate Names and Alternate Definitions for Data Elements</i> 129	
<i>Creating Alternate Names and Alternate Definitions for a Property</i>	<i>130</i>
<i>Creating Alternate Names and Alternate Definitions for Data Element Concepts</i>	<i>131</i>
Using Pre-mapped CDEs (Common Data Elements)	131
Creating and Updating Concepts in caDSR	131
Mapping a UML Class to an Object Class	132
Mapping a UML Attribute to a Property	133
Mapping a UML Class to a Value Domain	134
Value Meanings	136
Permissible Values	137
Using a Value Domain Defined within the Model	137

Mapping Data Element Concepts	138
Mapping Data Elements	139
Mapping UML Associations to Object Class Relationships	140
Mapping UML Inheritance	141
Reviewing and Verifying Registered Metadata	143
Using the UML Model Browser	144
<i>Viewing all registered elements for a Context or Project</i>	144
<i>Searching in the UML Model Browser</i>	145
Using the CDE Browser	147
<i>Viewing all CDEs for a Context, Classification Scheme or Classification</i> <i>Scheme Item</i>	147
<i>Searching in the CDE Browser</i>	148
Appendix A	
Troubleshooting	151
The Status Bar	151
Apply Changes vs. Saving Changes	151
Multiple Tabs Open	151
Search Dialog Box	152
Glossary	153
Index	157

ABOUT THIS GUIDE

This preface provides information about the *caCORE SIW and UML Model Loader v4.1 User Guide*.

Topics in this preface include:

- [Purpose](#) on this page
- [Audience](#) on this page
- [Topics Covered](#) on page 2
- [Additional References](#) on page 3
- [Text Conventions Used](#) on page 4
- [Credits and Resources](#) on page 5

Purpose

The purpose of this guide is to describe the Semantic Integration Workbench (SIW) and UML Model Loader components of caCORE. caCORE is the cancer Common Ontologic Representation Environment and is an open-source, standards-based, semantics computing environment and tool set created by the National Cancer Institute's Center for Biomedical Informatics and Information Technology (CBIT).

The SIW provides a way for UML model owners to bind the elements in their models to terms from a controlled vocabulary (EVS concepts). This binding is referred to as "annotation." Annotation is the process of tagging the items in your model with definitions of terms already resident in EVS. The purpose of annotating using a controlled vocabulary is to identify disparate terms in similar models that are being used to describe similar model elements that are used in a similar way.

For example, Model A uses the term "lastName" and Model B uses the term "surname" and Model C uses the term "familyName", it is likely that they all mean the same thing and each can be semantically identified using the same EVS concepts.

Once the model is annotated with EVS concepts, the concept codes and concept code combinations are used to identify and, where possible, link your model to metadata elements that have already been registered by other UML models. This linking is the key to semantically integrating your model with other registered UML models. And semantic integration is one way to ensure that your information can be shared between and analyzed by systems across the cancer research community.

After your model has been fully annotated and brought semantically in line with other existing systems, you can select to have your model registered in caDSR. caDSR is the Cancer Data Standards Registry, and consists of a database and toolset that are used to create, edit, and reuse common data elements. These data elements (also known as DEs or CDEs) represent common pieces of metadata used by registered UML models and which can be re-used by other models as appropriate.

The UML Model Loader is the tool used to register UML model metadata into the caDSR. Registering in caDSR loads your model's elements into the caDSR registry, identifies the elements as being owned by or used by your model, and makes them available for use by other models.

This guide provides an overview of the SIW and the UML Loader and explains how these two components can be used to achieve semantic integration of your UML model with other organizations' models.

Audience

The information in this guide is directed at UML model owners who want to bring their model semantically in line with other similar organizations' systems, and (if desired) to register their system's metadata into caDSR. This guide is also designed to provide other types of readers (non-programmers) with a basic overview of the CBIIT's purpose and methods for semantic integration between dispersed research information systems.

This guide also contains information directed at EVS curators who are using the SIW to annotate the XMI files submitted to CBIIT for curation.

This guide provides information about how to use the SIW to bind the elements from any UML model to metadata concepts located in EVS, or to data elements already registered in caDSR.

Topics Covered

The primary information covered in this guide is the use of the SIW by both model owners and EVS curators. This information includes detailed explanations and procedures for using the tool, as well as information on why each step is important.

Because the UML Model Loader is used exclusively by CBIIT personnel, the information included in this guide is limited to an overview of the end result of the UML loading process. This guide also provides basic instructions for using the CDE Browser and UML Model Browser to search the caDSR for your model's loaded elements.

Below is a list of the chapters contained in this guide and a brief overview of the information you will find in each chapter:

- [*Chapter 1, Overview of caCORE*](#), on page 7 provides overview information about the caCORE processes and tools. This chapter also contains a section about UML modeling and some brief information about properly constructing a model for use with the caCORE toolset.
- [*Chapter 2, Introduction to Semantic Integration*](#), on page 15 provides an overview of what semantic annotation is, what semantic integration is, why they

are important for interoperability between systems published on the grid, and how the SIW works to partially automate this step in caCORE process.

- *Chapter 3, Overview of the SIW*, on page 27 provides the basic information you need to know to use the SIW, including an overview of the sections of the SIW viewer and their functionality, and details about the viewer preferences available, how to change the default preferences, and how these changes affect the display of your model information in the SIW viewer.
- *Chapter 4, Reviewing An Unannotated File*, on page 51 provides information and procedures for using the Review Unannotated XMI File option of the SIW.
- *Chapter 5, Using XMI RoundTrip*, on page 59 provides information and procedures for using the XMI Round Trip option of the SIW.
- *Chapter 6, Using the Semantic Connector*, on page 67 provides information and procedures for using the Semantic Connector option of the SIW.
- *Chapter 7, Curating An XMI File*, on page 73 provides information and procedures for using the Curate an XMI File option of the SIW.
- *Chapter 8, Reviewing An Annotated File*, on page 91 provides information and procedures for using the Review Annotated XMI File option of the SIW.
- *Chapter 9, Using GME Annotations*, on page 109 provides information regarding the use of GME namespace tags in your UML model, including a set of scenarios that may help you define when to use which step of the SIW to add these tagged values to your model.
- *Chapter 10, Understanding the UML Model Loader*, on page 121 provides an overview of the process (including the expected outcome) of registering your model's metadata in caDSR. This chapter also provides basic information on using caDSR tools to browse for your model's elements once they are loaded.
- *Appendix A, Troubleshooting*, on page 151 provides basic information regarding a few of the more common issues users encounter.
- *Appendix B, Glossary*, on page 153 provides definitions for acronyms and terms used throughout this guide, as well as for terms that relate to caCORE in general.

Additional References

For more information about caCORE Semantic Integration tools and other caCORE components referenced in this guide, refer to the caCORE page of the caBIG web site, which contains information about all of the tools available in the caCORE suite: https://cabig.nci.nih.gov/concepts/caCORE_overview.

The caCORE Wiki also provides a centralized place for overviews and links to additional information and release notes. The caCORE Wiki can be found at: <https://wiki.nci.nih.gov/x/BIAe>.

Specific locations on the wiki that may be of interest include:

- **SIW Wiki Page**, which contains links to the Release Notes for the past several releases: <https://wiki.nci.nih.gov/x/QYEI>.
- **caDSR Wiki Page**, which contains descriptions of each of the tools in the caDSR toolset, links to release notes, and links to the URLs for each of the available tools: <https://wiki.nci.nih.gov/x/QYEI>.
- **XMI Tag Reference**, which contains a list of the XMI tagged values used by the caCORE products, a description of each tag, and notes what product(s) use each tag: <https://wiki.nci.nih.gov/x/SYI8>.

In addition to the wiki, the Semantic Integration Tools Project on GForge provides extensive information regarding both past and ongoing development of the SIW and UML Loader: <http://gforge.nci.nih.gov/projects/siw/>.

Text Conventions Used

This section explains conventions used in this guide. The various typefaces represent interface components, keyboard shortcuts, toolbar buttons, dialog box options, and text that you type.

Convention	Description	Example
Bold	Highlights names of option buttons, check boxes, drop-down menus, menu commands, command buttons, or icons.	Click Search .
URL	Indicates a Web address.	http://domain.com
text in SMALL CAPS	Indicates a keyboard shortcut.	Press ENTER.
text in SMALL CAPS + text in SMALL CAPS	Indicates keys that are pressed simultaneously.	Press SHIFT + CTRL.
<i>Italics</i>	Indicates emphasis on particular words or phrases.	To undo your changes, select a different node <i>before</i> clicking Apply .
<i>Italics</i>	Used for references to other documents, sections, figures, or tables.	See <i>Figure 4.5</i> .
<i>Italic boldface monospaced type</i>	Represents text that you type.	In the New Subset text box, enter <i>Proprietary Proteins.</i>
Note:	Highlights information of particular importance	Note: This concept is used throughout the document.
{ }	Surrounds replaceable items.	Replace {last name, first name} with the Principal Investigator's name.

Credits and Resources

The following table lists the development and management teams that contributed to this document.*

System and Process Development	Documentation	Project and Product Management
Christophe Ludet ³	Bronwyn Gagne ²	Denise Warzel ¹
Claire Wolfe ⁴	Jill Hadfield ¹	Greg Raley ¹
Ashwin Mathur ⁵		Noble Dzakpasu ⁵
¹ National Cancer Institute Center for Biomedical Informatics and Information Technology (NCI CBIIT)	² 5AM Solutions	³ Oracle
⁴ TerpSys	⁵ Ekagra Technologies	

Table 2.1 Semantic Integration Workbench development and management teams

**This contract has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Contract No. HHSN261200800001E.*

The following information can be used to contact CBIIT Application Support if you are in need of assistance using the Semantic Integration Workbench.

- CBIIT Application Support Website: <http://ncicb.nci.nih.gov/NCICB/support>
- Telephone: 301-451-4384
- Toll free: 888-478-4423
- Email: ncicb@pop.nci.nih.gov

CHAPTER

1

OVERVIEW OF caCORE

This chapter provides an overview of caCORE and a brief discussion about the purpose of the caCORE process. This includes information about each of the tools in the caCORE toolset. Because the caCORE process is based on UML modeling, this chapter also contains a section that discusses UML modeling for caCORE, including a list of best practices and modeling constraints.

Topics in this chapter include:

- *caCORE Overview* on this page.
- *UML Modeling for use with caCORE* on page 9
- *Generating the XMI File for the caCORE Process* on page 13

Additional information is available on the FAQs - UML Modeling page on the caCORE Wiki at: <https://wiki.nci.nih.gov/x/7wFy>.

caCORE Overview

The NCI Center for Biomedical Informatics and Information Technology (CBIIT) is an organization that provides biomedical informatics support and integration capabilities to the cancer research community. In order to assist with this integration, CBIIT has developed caCORE, the cancer Common Ontologic Representation Environment (caCORE). caCORE is a data management framework designed for researchers who need to be able to access and provide data across a large number of data sources.

By providing a common data management framework, caCORE helps streamline the informatics development throughout academic, government, and private research labs and clinics. The components of caCORE support the semantic consistency, clarity, and comparability of biomedical research data and information. caCORE is open-source enterprise architecture for NCI-supported research information systems, built using formal techniques from the software engineering and computer science communities.

The four characteristics of a caCORE-compatible system include:

- Model Driven Architecture (MDA)
- n-tier architecture with open Application Programming Interfaces (APIs)
- Use of controlled vocabularies, wherever possible
- Registered metadata

The use of MDA and n-tier architecture, both standard software engineering practices, allows for easy access to data by other applications. The use of controlled vocabularies and registered metadata, less common in conventional software practices, requires specialized tools that are generally unavailable.

As a result, CBIIT (in cooperation with the NCI Office of Communications) has also developed the Enterprise Vocabulary Services (EVS) system to supply controlled vocabularies, and the Cancer Data Standards Repository (caDSR) to provide a dynamic metadata registry. The Semantic Integration Workbench (SIW) and UML Model Loader provide a process-bridge between the EVS, the caDSR and any UML model.

When a system meets all four of the development characteristics listed above, it is said to be “caCORE-like.” Such systems provide several advantages:

- With its open APIs, the end user (whether human or machine) can retrieve data without having to understand the implementation details of the underlying data system.
- The maintainer of the resource can move the data or change implementation details without affecting the ability of remote systems to access the data.
- Most importantly, the system is *semantically interoperable*; it uses runtime-retrievable information to provide an explicit definition and complete data characteristics for each object and attribute supplied by the data system.

Components of caCORE

The components that comprise caCORE include: SIW, EVS, caDSR, UML Loader, SDK, CSM, and CLM. The following subsections provide a brief description of each component.

For more details on other caCORE components, refer to the caCORE Overview web page at https://cabig.nci.nih.gov/concepts/caCORE_overview, which directs you to other product-specific informational pages.

You may also want to view the caCORE Wiki (<https://wiki.nci.nih.gov/x/BI Ae>), which provides overviews of the caCORE toolset and links to product-specific documentation.

Semantic Integration Workbench (SIW)

The SIW is a tool designed to help UML model owners map the elements of their model to metadata concepts already defined in the NCI Thesaurus and, where possible, to data elements already registered in the caDSR. Essentially the SIW provides an easy way to semantically bind the elements from one UML model to terms or definitions used by elements in other systems’ models. The ultimate purpose is to see that systems use the same terms for identical items, allowing those systems to more easily gather and/or share data.

Enterprise Vocabulary Services (EVS)

EVS provides controlled vocabulary resources that support the life sciences domain, implemented in a description logics framework. EVS vocabularies provide the semantic raw material from which data elements are constructed.

Cancer Data Standards Registry (caDSR)

The caDSR is a metadata registry based upon the ISO/IEC 11179 standard. It is used to register the descriptive information (metadata) needed to render cancer research data reusable and interoperable. The caBIO, EVS, and caDSR data classes are registered in the caDSR, as are the data elements used on NCI-sponsored clinical trials case report forms, as well as from caBIG- and CBIIT-sponsored UML models.

UML Model Loader

The UML Model Loader is a tool used by CBIIT to register the metadata elements from a system's UML model into caDSR. The UML Loader helps automate the process of binding UML model metadata elements to registered caDSR components, and where necessary, create new components that correspond to the elements in the model.

Software Development Kit (SDK)

The caCORE SDK is designed to assist researchers and application developers with creating caCORE-compatible APIs for their system. When use of the caCORE SDK is combined with the use of controlled vocabularies and registered metadata, the resulting software system is considered to be a “semantically integrated caCORE-like system” because the runtime-accessible data elements available through the exposed APIs will have been defined using controlled terminology. These elements will therefore be easily understood and interpreted by other caCORE-like systems.

Common Security Model (CSM)

The Common Security Model (CSM) provides a flexible solution for application security and access control with three main functions:

- Authentication to validate and verify a user's credentials
- Authorization to grant or deny access to data, methods, and objects
- User Authorization Provisioning to allow an administrator to create and assign authorization roles and privileges.

Common Logging Module (CLM)

The Common Logging Module (CLM) provides a separate service under caCORE for audit and logging capabilities. It also comes with a Web-based locator tool.

Client applications can use the CLM directly, without the application using any other components such as the CSM.

UML Modeling for use with caCORE

All caCORE products are based on UML modeling and use XMI files of the system's representational model. What the caCORE tools do with respect to the XMI output of your model is explained in more detail in the documentation for each of the tools.

Since the XMI file you export from your UML modeling software may be used across the caCORE toolset, you want to be sure that the model and the elements within it meet the conventions either recommended or required by the various caCORE tools.

The purpose for this section of this guide is to identify those things that need to be done or verified in the model before exporting the XMI file for use.

The topics in this section include:

- *Modeling Constraints*, below
- *Naming Best Practices* on page 11

Enterprise Architect (EA) and ArgoUML are the officially supported UML modeling tools for use with the caCORE suite of products.

Throughout this guide you will notice the use of XMI to describe the file exported/saved from the UML modeling software and processed by the caCORE tool. XMI is simply the format of the file being processed. It is understood that EA can create an .xmi file and that ArgoUML creates a file with a .uml extension. For more information on the proper output from each of these modeling tools, see *Generating the XMI File for the caCORE Process* on page 13.

Modeling Constraints

Some of the caCORE tools have recommendations and constraints for the UML models to be used with the programs. Those constraints most pertinent to the SIW and UML loader are identified below.

Constraint 1: Ignored UML Elements—While UML class elements are recognized by caCORE tools, and those classes may contain both attributes and operations, operations are ignored by the caCORE tools.

Constraint 2: Attribute Types—Each attribute must have a type assigned to it. Each Java wrapper class must be defined within the model under the appropriate java package (e.g., java.lang.String). For a complete list of allowed datatypes, see <http://cadsrsiw.cni.nih.gov/datatype-mapping.xml>.

In accordance with object-oriented design principles, attributes of a class that are of a complex type should be modeled as associations. The only exception is if you create a “caDSR Value Domain” stereotyped class in your UML model. For a UML attribute to use this value domain, the model owner must either select the LVD from the drop-down list in the SIW or add the tagged value “CADSR Local Value Domain” / “My Value Domain” as described in the next chapter in *Pointing a UML Attribute to a Local Value Domain* on page 26.

Constraint 3: Recognized Relationships—The caCORE tools recognize association and generalization (inheritance) relationships. In addition, the UML Loader records aggregations and compositions when registering metadata, but for the purposes of generating CDEs, these types of relationships are treated as simple associations.

Constraint 4: Association End Role Names—Both ends of each association should be given a role name (source role and target role).

Constraint 5: Association End Multiplicity—The multiplicity of each association end must be specified at both ends.

Constraint 6: Association End Navigability (Directionality)—The navigability of each association must be specified.

Constraint 7: Packages and the Logical Model—Classes in the model need to be placed under at least one package under the Logical Model package (for example, add a package named “domain” and place the classes under that). There can be any number of packages under the Logical Model package and you may add them to whatever level depth you choose. The only real constraint is that the classes must belong to at least one package that does not have a space in the name. Packages with spaces in the name are treated as if they do not exist. For example, a package with the name “gov.nci.my model.domain” is treated as “gov.nci.domain”.

If you are using local value domains rather than existing caDSR value domains, you are also strongly encouraged (though not required) to place all of your LVDs into a separate package under the Logical Model.

Constraint 8: UML Descriptions — UML classes and attributes must have textual descriptions associated with them. This means that a definition for each class and each attribute must be entered into the model using the CADSR_Description tagged value. The SIW will present an error for any element that does not contain descriptions for an element. This information is necessary to let the EVS curators know what the purpose of the element is and therefore to properly map it to an EVS concept.

The text provided in the CADSR_Description tagged value is the information that appears in the UML Class Definition or UML Attribute Description field of the SIW. You may have up to eight CADSR_Description tagged values for each element (CADSR_Description, CADSR_Description2, CADSR_Description3, etc.). If you have multiple CADSR_Description tagged values, the text is concatenated, up to a maximum of 2000 characters. If you are using EA, you are limited to 256 characters per tag.

Note: The SIW still recognizes the legacy tagged values for classes and attributes (“documentation” and “description” respectively), however the CADSR_Description tagged value is preferred. Furthermore, when creating or editing element descriptions through the SIW, the SIW will write the CADSR_Description tags back to the XMI file, and not the legacy tagged values.

Constraint 9: Java Limitations—Because the SDK produces a Java-based system, class and attributes names are subject to the conventions defined by the Java language. UML models should *not* use Java reserved words as class or attribute names (i.e., int, long, class, interface, void, etc.). Also, model elements must not use hyphens, angle brackets or other reserved characters that will result in Java compilation errors.

Naming Best Practices

To ensure semantic interoperability, you should pay close attention to the class and attribute naming conventions established by the Sun Microsystems [Java Bean Specification](#). Decisions as to the names you apply to your UML element can affect many things, including the automated mapping of the elements to EVS concepts through the Semantic Connector, the mapping of the elements to common data elements through the Roundtrip step, as well as the generation of the system code by the SDK Code Generator.

The following list, while not exhaustive, provides some broad outlines regarding naming best practices that you should consider during modeling.

Adopt a consistent naming convention, using camel case for class and attribute names—Where a term is used and re-used within the model, you should make sure that if the intent and use of the term is the same, that the naming convention and description for the term is also consistent throughout the model. In addition, the Sun Microsystems [Java Bean Specification](#) establishes the use of camel case for classes and attributes, and is briefly described below:

- Classes
 - One word class names are capitalized (e.g., “Patient” or “Location”).
 - Multiple word class names should have no space between words and the first and subsequent words are all capitalized. (e.g., SequenceVariant or PatientAddress).
- Attributes
 - One-word attribute names should be all lowercase (e.g., “name” or “gender”).
 - Multiple word attribute names should have no space between words and the first word should be lowercase, while the first letter of second and subsequent words should be capitalized (e.g., streetNumber or initialDiagnosisDate).

The camel case convention is not simply a “nice to have” because it enhances readability; the Semantic Connector step of the SIW parses element names based on camel case, mapping each part of the name it finds to EVS concepts. In this way, using camel case in the model helps to automate the annotation process.

Avoid abbreviations and acronyms—To the extent possible and reasonable, avoid the use abbreviations and acronyms and use full terms instead (e.g., use ‘DatabaseReference’ instead of ‘DbRef’).

Do not use jargon—Avoid the use of ‘jargon’ terms where standard terms exist (e.g., use “microarray” not “chip”).

For the above two items, all you have to do is keep in mind that jargon, abbreviations and acronyms tend to be industry-specific or regionally-based and are not universal.

Do not repeat class names in attributes —Since the attribute is already associated with the class, repeating the class name is not necessary.

For example, for a class element called “Gene” you would use the attribute “name” not “geneName.” Using the latter format causes the Semantic Connector to unnecessarily map the concept code(s) for “Gene” multiple times for a single element, which is semantically undesirable. While any repetitive naming should be caught during the curation process, it is good practice to keep these out of the model in the first place.

Do not use Java reserved words as class or attribute names—Using Java reserved words prevents the SDK Code Generator from compiling the generated system code. This is also stated as [Constraint 9: Java Limitations](#) on page 11.

Generating the XMI File for the caCORE Process

The caCORE products, including the SIW, use a pre-processor called the XMI Handler to extract the XML tags needed to perform a semantic annotation of UML classes, attributes, and associations from the UML model. It also extracts various other model features and any caCORE specific tagged values that were entered by the model owner.

If you are using EA, you must export the XMI file using the `.xmi` extension. In addition, you must be sure that the DTD option is *not* checked and that the EA Roundtrip option *is* checked.

If you are using ArgoUML as your modeling tool, you can simply save your model as a file with a `.uml` extension.

CHAPTER 2

INTRODUCTION TO SEMANTIC INTEGRATION

This chapter provides an introduction to semantic integration and how it fits into the caCORE process. Since semantic integration is a fairly broad topic, we have attempted to break down the component pieces of the semantic integration process into smaller, more easily understood parts.

Topics in this chapter include:

- *Semantic Integration and Semantic Annotation* on page 15.
- *Terms You Should Know* on page 16.
- *Getting From Model to Annotation to Integration* on page 18.
- *Data Element/Common Data Element (DE/CDE)* on page 20.
- *Value Domains* on page 21.

Semantic Integration and Semantic Annotation

The first thing you have to understand is the difference between semantic integration and semantic annotation.

Semantic Annotation is the process of adding tagged values to a UML model to describe the semantics of the elements in the model. These semantics are described using terms from a common or “controlled” vocabulary. Basically, semantics are the language you use. You annotate your terms so that if you use the term “agent” it is clear that you mean “a medication or other item used to produce an effect” and not “a real-estate representative” or “a government spy”. If everyone using the term “agent” also uses the same definition from the common vocabulary, then we are all clear that we are using the same language (semantics). When all of the terms in your model are mapped to these definitions, your model is considered to be fully semantically annotated and is ready to be loaded to caDSR.

Semantic Integration is the process of loading these annotated models into a centralized registry (in this case, caDSR) and equating the elements in the models with common data elements, bringing together all of the annotated representations of data into a single, accessible location. When the elements in your model become represented in caDSR as caDSR metadata, your model is considered to be semantically integrated.

For the caCORE process, proper semantic annotation requires that each UML class and attribute (and in some cases value domain) in the model gets mapped to one or more metadata concepts from a controlled vocabulary. At CBIIT, the preferred vocabulary is the NCI Thesaurus, maintained by the Enterprise Vocabulary Services (EVS) staff. It is the association of a model's elements with these controlled concepts that allows for unambiguous interpretation of UML model objects and mapping between objects from different domains. The resulting data elements are more sharable and interoperable.

The SIW tool helps automate the semantic annotation process by providing some level of automated mapping as well as a GUI for manual mapping of model elements to EVS concepts.

Once your model is annotated, the CBIIT staff uses the UML Model Loader tool to load your annotated model to caDSR, binding your model elements to caDSR metadata, thus semantically integrating your model.

The caDSR metadata registry, based upon the ISO/IEC 11179 standard, registers the descriptive information needed to render cancer research data reusable and interoperable. For more information about the ISO/IEC 11179 standard, see http://isotc.iso.ch/livelink/livelink/fetch/2000/2489/lttf_Home/PubliclyAvailableStandards.htm??Redirect=1.

For more details on caDSR metadata and how it relates to your UML model elements, see *Data Element/Common Data Element (DE/CDE)* on page 20 and *Value Domains* on page 21.

Terms You Should Know

In order to understand the information being provided in this guide, it is important that you understand certain terms as they are used within the context of the caCORE environment and associated toolset. These terms also appear in *Appendix B, Glossary*, on page 153. However, they are important enough and pervasive enough in this text to be called out separately and defined in relation to semantic integration.

Metadata —Metadata is data about data, or is sometimes is described as the definition of a piece of information. For example, a person's name might be "Joe Smith." The data about the person is "Joe Smith." The metadata for that information is "name" and is defined as "a term that specifically identifies the item to which it is attached."

While "name" can function as a piece of metadata, the term "name" doesn't mean much by itself. The idea of a name can apply to a person, a drug, a facility, or any number of items. It is not the "name" that creates complexity but the attributes that can be associated with "name." If a person is filling out a form, there is probably a place for "First Name" and for "Last Name" and possibly for "Maiden Name." And while these

conventions are clear to some users, not all cultures use “first” as “given name” and “last” as “family name.”

In order for systems to be able to assemble and share the same sort of information from all participants, they need to be certain that the information exists in their systems in the same way. This is done by using and reusing the same metadata tags for the same information across systems.

UML Model—According to Wikipedia, UML or Unified Modeling Language is “a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model.” So that basically, a UML model is a graphical representation (picture) of a software application.

Using UML to model your software lets you identify the individual pieces of the application and relate them to one another. The caCORE toolset, including the SIW and the UML Loader is compatible with .xmi files exported from Enterprise Architect (EA) and with .uml files saved from ArgoUML. Throughout this guide, however, both types of files are referred to as XMI files, regardless of file extension.

XMI—XMI stands for “XML Metadata Interchange” and is a standard for exchanging metadata information via Extensible Markup Language (XML). The most common use of XMI is as an interchange format for UML models.

Class—According to Wikipedia, a class is “a programming language construct that is used to group related attributes and methods...a cohesive package of a particular kind of metadata.” For example, your model might contain a class “Patient” which is described in your model as “A person being acted upon by a medical professional.” The details of the object are defined by the attributes for the class (such as name or address or gender). The class is defined in the model, but through semantic integration and the SIW, is also defined using concepts from the EVS controlled vocabulary. The attributes for the class (which are also mapped to EVS concepts) provide further details about the class and are defined separately.

Attribute—An attribute defines the properties of an object. In terms of a UML model, the attributes associated with a class provide details about the use of the class in the system. For example, the model may have a class called “patient” and that class has several attributes associated with it including “gender”, “race”, “ethnicity”, “id” or other descriptors.

Each of the attributes in a model will be mapped to concepts or definitions so that each time they are used in the same way, the same definition follows. For example “id” may be defined as “Identifier - an item that uniquely identifies an object,” and “gender” may be defined as “the assemblage of properties that determines reproductive role.”

Mapping UML model elements to EVS concepts ensures that whenever an object is used in a particular way (even across models), the definition of the object for that use is the same and is unambiguous; it means the same thing every time.

Association—An association is simply the relationship between classes in a model, and includes details about that relationship (multiplicity, directionality, etc.). Associations also have “roles” which define how the element relates to the other elements it is associated with.

Tagged Value—A tagged value is a UML construct that represents a name-value pair. It can be attached to anything in a UML model and is used by UML modeling tools to store specific information about the element to which it is attached.

The caCORE toolset uses a variety of tagged values from the UML model to perform a variety of tasks. The SIW in particular uses element description tagged values to identify the purpose or meaning of the element in the model. Specifically, the model owner needs to add tagged values called CADSR_Description to each element in the model to define or describe the element. These descriptions help the EVS curation team properly map the element to EVS concepts. For more information regarding the tagged values used by the SIW and UML Loader, see the XML Tag Reference wiki page located at: <https://wiki.nci.nih.gov/x/SY18>.

EVS Concepts—Each metadata term in the NCI Thesaurus has a code, a name, and a definition for the term as well as an indication of the source of the definition. These are called “concepts” and function as the controlled vocabulary that are mapped to the elements in your UML model. Mapped concepts become tagged values for the elements in the model.

Unannotated and Annotated XMI Files—An Unannotated XMI file is one that has been exported from the UML modeling software but has not yet had all of its elements mapped to either EVS concepts or caDSR CDEs.

When you select the SIW step to Review Unannotated XMI file, the SIW hides concept annotation errors, because by definition, an unannotated file is missing concept mapping. Reviewing an unannotated file allows you to focus on any syntactic errors in the file (elements without definition or descriptions). This allows you to review and (where necessary) add those definitions before attempting to annotate the file.

An Annotated XMI file is one where all of the elements in the model have been mapped to EVS concepts or to caDSR CDEs. The elements in the file have been annotated this with additional information.

When you select the SIW step to Review Annotated XMI file, the SIW shows all errors in the file that will need to be fixed before the model can be loaded into caDSR. These include elements where concept information is missing, where duplicate mapping exists, or where invalid value domains have been used for elements, among other items.

Getting From Model to Annotation to Integration

As stated earlier in this chapter, annotation is the process of mapping a model's elements to concepts in a controlled vocabulary. This mapping is done by adding tagged values to the model elements, thus annotating the XMI file with these tagged values. The SIW helps you do this mapping and adds the proper tagged values to the XMI file.

But how does this really work?

Each element in the model must have at least one concept mapped to it. The first concept mapped to an element is called the Primary Concept. The Primary Concept must reflect the basic meaning or idea of the element. Each concept added after the Primary Concept is called a Qualifier Concept. Each Qualifier Concept mapped to an element provides more detailed information about the element.

For example, your model has an element named “patientBirthDate.” The fundamental idea of the element is that it is a date. So the Primary Concept mapped to that element might be the EVS concept called “date” indicating a point in time. The element then might also have two Qualifier Concepts mapped to it called “birth” and “patient.” The combination (referred to as Summary Concept) is then identified with the element as “Patient Birth Date” and has three associated concept codes, three concept names and three concept definitions, one for each mapped concept.

Once the annotated version of the model has been approved by the model owner and the EVS curation team, the XMI file is sent to the caDSR team to be transformed into caDSR metadata via the UML Loader.

The UML Loader uses the concept information (concept codes, names, and definitions) associated with each model element and uses that to find matching caDSR metadata. Once it finds a match, it associates the model elements (class, attribute, and value domain combination) to a caDSR common data element or CDE. When all of the elements in your model have been mapped to caDSR CDEs, your model is considered to be semantically integrated. That is, it is integrated, by language, with other registered system models.

This concept mapping done through the SIW acts as a bridge between the elements in the UML model and the metadata elements in caDSR, and provides a common vocabulary for the use and definitions of terms across models.

At this point you should be starting to understand why semantic integration is useful. If one developer registers their model in caDSR, then a different model owner can compare their model against the registered model and reuse existing elements from the registered model for those same elements in their model. Where the new model has elements not used before, new metadata elements are created for caDSR, which are then available for use by another system’s model owner.

Eventually you have a large number and wide range of systems that are using the same metadata for the same objects, making it possible for those systems to communicate and share data with one another more easily than if those systems had been created without using a common vocabulary or common metadata standards.

Data Element/Common Data Element (DE/CDE)

A Data Element (DE) or Common Data Element (CDE) is defined as the unique combination of a Data Element Concept and a Value Domain. The terms *data element* and *common data element* (or the abbreviations DE and CDE) are often used interchangeably.

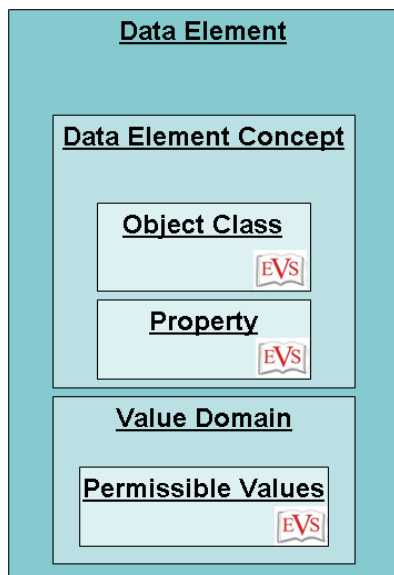


Figure 2.1 Graphical representation of a caDSR Data Element

A Data Element Concept (DEC) is the unique combination of an Object Class and Property. Object Classes correspond to UML model classes, and Properties correspond to UML model attributes. This means that if your UML model has a class that has four attributes associated with it, there will be four DEC's created for that item, one for each attribute/class combination.

As each of those DEC's is associated with a value domain, an existing data element is reused or a new one is created. So if there are four DEC's associated with a class (one for each class/attribute combination), there will be four data elements also associated (when the value domains corresponding to the datatypes associated with each attribute are added to each DEC).

This means that each piece of metadata associated with your model is derived from a combination of the definitions or descriptions established for the item the metadata represents. Remember that each element in your model was defined using one or more EVS concepts, so the full definition and intent of each element is understood, unambiguous and given commonality through the controlled EVS vocabulary. So when the elements of your UML model are mapped to caDSR metadata, you know that the metadata accurately represents the characteristics of the model element (class, attribute, value domain) to which it is mapped.

Graphically, the correlation between UML model elements and the component parts of a DE looks like this:

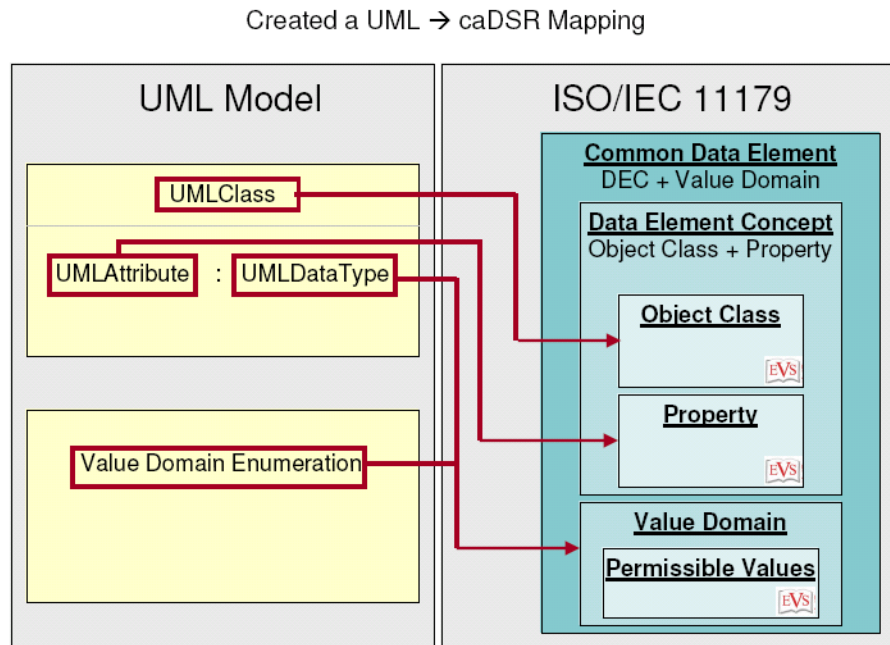


Figure 2.2 Mapping between UML model elements and a caDSR CDE

This mapping or transformation of UML model elements into caDSR metadata is performed initially by the UML Loader, using the concept information provided through the annotation process. After your model has been loaded into caDSR, the Roundtrip mode of the SIW can apply this mapping back into your XMI file for re-loading into the UML model, to use for the next version of the system.

Value Domains

Value domains are defined as the “domain of possible values” for an element in the model. This is typically expressed as the type of information that the system can accept as valid data for an item. For example, your model has an attribute “id” and the system that the model is based on has restricted all IDs to be numeric. The value domain for that attribute might be “java.lang.Integer.” This identifies the type of values that your use of “id” can have. Value domain datatypes are an indication of the form that a value will have. Examples include string, integer, and character.

So for example, let’s say your piece of data is the size of a tumor in millimeters. Some model owners may call the attribute “Tumor.mm” and map Millimeter Size concepts to the attribute. Another model owner may call the attribute Tumor.size and map just the Size concept to the attribute. Actually the second is more correct. The “mm” part of this attribute is the REPRESENTATION of the data, not what that attribute is. The trick to value domains is to tease out the representation from the semantic meaning of the attribute.

Enumerated and Non-Enumerated Value Domains

Value domains can be “enumerated” or “non-enumerated.” Non-enumerated value domains mean that there are restrictions to the possible values for the item (numeric

alphanumeric) but the possible values are not specified. Enumerated value domains mean that the possible values for the item are explicitly stated. So for an attribute like “gender” the value domain might be enumerated and limited to “male” and “female” or to “m” and “f” depending on how the system recognizes the informational input.

If you work more deeply into the idea of enumerated vs. non-enumerated value domains, you actually find that you have different kinds of value domains.

- Simple non-enumerated: the possible values are only restricted to the type of values available for the value domain (numeric or integer), though you can identify a unit of measure (like millimeters) to further restrict the type of values.
- Simple enumerated: a value domain with a list of permissible values.
- Enumerated by reference document: these are actually identified as non-enumerated (N) because there is not an explicit set of permissible values, but a reference to where you can find the list of acceptable values.
- Enumerated by reference concept: similar to the previous type, except that the reference containing the list of acceptable values is a Concept instead a reference document.
- Enumeration constrained by a Parent Concept: these value domains do have an explicit list of permissible values, but there is also an EVS concept that acts as a parent concept. This means that the permissible values for this value domain must be concepts that are children of the parent (though it doesn’t have to include all children concepts).

For those value domains that are enumerated by concept, these VDs must be tagged in the model using the “ValueDomainConceptCode” tag to identify the appropriate EVS concept. As with all tagging, this can be done in the modeling software itself or through the SIW by mapping the VD and its associated value meanings to the appropriate EVS concepts. See [Adding EVS Concepts to an Element in Review Mode](#) on page 96 for instructions.

You can map to existing caDSR value domains or define the value domains within your model. Most model owners use existing caDSR value domains to map to their attributes, and you are strongly encouraged to use existing caDSR value domains whenever possible. The SIW makes it easy to map attributes to existing value domains because it allows you to search the caDSR for the value domain you need and then select it for mapping.

When the SIW performs this mapping, it adds the following tags to the model attribute(s) indicate the value domains for the object:

- CADSR_VD_ID (the Public ID of the value domain)
- CADSR_VD_VERSION

For more information on these tags as well as the ValueDomainConceptCode tag, see the XMI Tag Reference wiki page located at: <https://wiki.nci.nih.gov/x/SYI8>.

Value domains can also be defined within the model. These VDs are called “local value domains” and are discussed in more detail in the next section.

Using Local Value Domains

As stated above, most model owners use existing caDSR value domains to map to their attributes, and you are strongly encouraged to use caDSR value domains whenever possible. However, you may create local value domains (LVD) within your model by creating classes and assigning them the proper stereotype. You will also need to add the mandatory tagged values. Some of these tags are listed in [Creating Local Value Domains](#) on page 24 below. You should also refer to the XMI Tag Reference wiki page at <https://wiki.nci.nih.gov/x/SYI8> for a complete list of the XMI tagged values recognized and used by the SIW and the UML Loader.

If you are using LVDs in your model, review the information in this and the next section closely, so that you fully understand how using LVDs can affect the loading of your model to caDSR.

In order to use LVDs for attribute mapping, you must create the value domains in the model. If the LVDs reside in the model, you can use the SIW to map the attribute to the LVD (as long as the attribute is not already mapped to an existing CDE, which by definition comes with a value domain already).

If you are using LVDs in your model, while you can put the value domains any where you want to, you are strongly encouraged to keep your LVDs in a separate package. Many people use a package conveniently called “ValueDomains” and place the package directly under the Logical Model.

Keeping all of your LVDs in a separate package allows you to easily exclude all of the value domains from other caCORE program processing (like the SDK or the UML Loader) by simply excluding that package. The SIW shows the value domains in the same place (the ValueDomains node of the tree) regardless of where the value domains exist in the model.

Local Value Domains and the UML Loader

If you are using LVDs, when your model is submitted for registration into caDSR, the UML Loader will try to match the LVD to an existing value domain with the same long name and permissible values. If there is no match, the UML Loader will create new value domains and load them into caDSR.

You also have the option to map an LVD to a caDSR value domain. If you are using LVDs in your model and a previous version of your model has already been loaded into caDSR, you should attempt to map your LVDs to the previous ones that now exist in caDSR. This mapping can be done through the SIW or in the model itself by adding the appropriate tags to the attributes (CADSR_VD_ID and CADSR_VD_VERSION).

The point of mapping your LVDs to existing caDSR value domains (either yours from a previous model or other appropriate existing VDs) is that doing so will make reloading your model easier because the UML Loader will no longer try to find a match based on long name. Furthermore, if the UML Loader finds a matching VD in caDSR but the attached permissible values do not exactly match, the Loader will stop loading the model because it does not know how to handle permissible values discrepancies. This will then require time-consuming investigation and intervention on the part of the model owner and the CBIIT team to rectify.

However be advised that if you have made changes to an LVD in your model and you do map it to an existing caDSR value domain, the value domain in caDSR will not be changed in any way to reflect any changes to the LVD that might be in your model.

Creating Local Value Domains

For the UML Loader to create an LVD for use with the model's attributes, the model owner must first create a UML class in the model and stereotype it as a "caDSR Value Domain" or as "enumeration." When a UML class is stereotyped as "caDSR Value Domain" the SIW identifies it as a value domain rather than an Object Class. The name of the stereotyped class will become the name of the value domain in caDSR, unless you map the class to EVS concepts, in which case the class will take on the naming convention mapped through those concepts.

caDSR naming conventions specify that the representation type should be the last term in the value domain name, such as in "State Code" where the representation term is "Code" and not "State."

LVDs should be created in a separate package from the regular domain objects.

The following six tagged values must be present in the "caDSR Value Domain" or the "enumeration" class in order to be recognized as a valid value domain.

- **caDSR_ValueDomainDefinition** - Used as the Value Domain Preferred Definition.
- **caDSR_ValueDomainDatatype** - Used as the underlying datatype for the value domain and must be one of the valid caDSR datatypes.
- **caDSR_ValueDomainType** - Used as the underlying value domain type and must be one of 'E' (for Enumerated value domains) or 'N' (for Non-enumerated value domains).
- **caDSR_ConceptualDomainPublicID** - The combination of conceptual domain public id and version must point to an existing conceptual domain in the caDSR.
- **caDSR_ConceptualDomainVersion** - The Conceptual Domain tagged value for this class should be entered as either a whole number or a decimal e.g. 1 or 1.0 for "Version 1.0".
- **caDSR_RepresentationPublicID** - The Representation Public Id tagged value for this class should be entered as the public id of an existing caDSR Representation Term.
- **caDSR_RepresentationVersion** - the Representation Version tagged value for this class should be entered as either a whole number or a decimal e.g. 1 or 1.0.

The list of optional tags listed below are used to indicate a "top level" concept for a value domain. This is also referred to as a "referenced value domain."

In a non-enumerated (N) value domain, the reference to a top level concept indicates that the values are not explicitly listed as attributes of the value domain class, but that data values for the attribute are constrained to children of the top level concept.

When a top level concept is present for an enumerated domain (E), this indicates that the permitted data values are listed explicitly as attributes of the value domain class and are children of the top level concept.

The following is a list of the optional tagged values for value domains:

- ValueDomainConceptCode
- ValueDomainConceptDefinition
- ValueDomainConceptDefinitionSource
- ValueDomainConceptPreferredName
- ValueDomainQualifierConceptCodeN
- ValueDomainQualifierConceptDefinitionN
- ValueDomainQualifierConceptDefinitionSourceN
- ValueDomainQualifierConceptPreferredNameN
- NCI_VD_UOM
- NCI_VD_DISPLAY_FORMAT
- NCI_VD_MIN_LENGTH
- NCI_VD_MAX_LENGTH
- NCI_VD_DECIMAL_PLACE
- NCI_VD_HIGH_VALUE
- NCI_VD_LOW_VALUE

For a complete (and regularly updated) list of the XML tagged values used by the SIW and UML Loader, refer to the XML Tag Reference wiki page at <https://wiki.nci.nih.gov/x/SYI8>. For more information on mapping value domains in caDSR, see *Mapping a UML Class to a Value Domain* on page 122

Value Meanings

Each attribute within a “CADSR Value Domain” or “enumeration” class is considered a “permissible value.” Mapping concepts to those attributes creates “value meanings.” Adding permissible values (attributes) to a value domain class and then mapping them to concepts is the preferred way to create value meanings. Annotation for permissible values is done the same way as it is for attributes with the exception that the tag names are different:

- ValueMeaningConceptCode
- ValueMeaningConceptDefinition
- ValueMeaningConceptDefinitionSource
- ValueMeaningConceptPreferredName
- ValueMeaningQualifierConceptCodeN

- ValueMeaningQualifierConceptDefinitionN
- ValueMeaningQualifierConceptDefinitionSourceN
- ValueMeaningQualifierConceptPreferredNameN

The value meaning name is created by concatenating the concepts mapped to the permissible value. The value meaning description is created by concatenating the concept definitions mapped to the permissible value.

Pointing a UML Attribute to a Local Value Domain

In order to indicate that a UML attribute should use a value domain defined within the model, the model owner can either add a tagged value of type “CADER Local Value Domain” to the attribute in the model or use the SIW to map the LVD to the attribute.

For example, the model owner has created a class that has been stereotyped “CADER Value Domain” with a name of “MyValueDomain.” For a UML attribute to use this value domain, the model owner can add a tagged value called “CADER Local Value Domain” / “MyValueDomain” to the attribute in the model before exporting.

You may, however, find it easier to use the Value Domain drop-down list in the SIW Viewer to choose the appropriate LVD for mapping from the list of LVDs pulled from the model.

CHAPTER

3

OVERVIEW OF THE SIW

This chapter provides information regarding how to open the Semantic Integration Workbench (SIW), an overview of the options available in the SIW, how to navigate the SIW Viewer, and how to customize the viewer to your personal viewing preferences. The last sections of this chapter provide instructions for saving changes to your XMI file and for updating your UML model with the changes made through the SIW.

Topics in this chapter include:

- *Introduction* on page 27
- *Pre-Requisites to Using the SIW* on page 28
- *Launching the SIW* on page 29
- *Using the SIW Viewer* on page 35
- *Setting Viewer Preferences* on page 44
- *Saving Changes to a File* on page 48
- *Updating UML Model Definitions with SIW-Generated Changes* on page 49

Introduction

The Semantic Integration Workbench (SIW) is designed to help UML model owners work through the semantic annotation process, and to remove (whenever possible) the need for users to understand the more complicated details of semantically integrating their model to other models registered in the caDSR. The caDSR is a metadata registry managed by CBIIT that is designed to provide consistently managed metadata for use by other research systems and data repositories.

The SIW essentially helps UML model owners bring the metadata for their models into line with the models used by other facilities. To do this, the SIW contains the following capabilities:

- Automates the searching for and (where possible) the matching of UML elements to items already resident in a controlled vocabulary (EVS).
- Allows users to create and/or edit element description tags in the SIW interface rather than having to update the model separately and re-export.
- Streamlines semantic annotation by offering direct queries to the NCI Thesaurus, inserting the concept information from the search into the user's file in the SIW, creating the appropriate tag names automatically. This eliminates syntax errors in the final XMI file.
- Allows for curation of the XMI file which maps existing or new EVS concepts (not yet in NCI Thesaurus) to classes and attributes. (Curation is performed by CBIIT personnel.)
- Allows for mapping of attributes to existing caDSR data elements, either automatically through the Roundtrip step or manually through the SIW viewer.
- Allows review of the final XMI file before it is loaded, providing for individual review and acceptance of each entry.
- Ensures that files submitted for loading into the caDSR have been checked for missing information and validated using [Silver Level Compatibility rules](#).

Pre-Requisites to Using the SIW

Using the SIW requires that you have a valid XMI file exported from either Enterprise Architect (EA) or ArgoUML and that you have the Java Web Start application installed on your computer. For more information, see the sections that follow.

Note: Throughout this guide, the term “XMI” refers to the type of file output from either your UML modeling software or the SIW itself. If you are using Enterprise Architect, the output file will have an .xmi extension; if you are using ArgoUML, the output file will have a .uml extension. In either case, these are referred to as XMI files.

Possessing a Valid XMI File

Before you can use the SIW, you need to have a valid file for the SIW to work with. If you are starting with an un-annotated file, be sure you have a valid export of an .xmi file from EA or have saved a .uml file from ArgoUML. If you have not generated this file yet, see [Generating the XMI File for the caCORE Process](#) on page 13. If you are using EA and your file was not exported using the appropriate options, the SIW will not be able to parse the file properly for use.

Note: The “Fix XMI” task (used in previous versions) is no longer needed to prepare the file for use with SIW. Therefore the XMI file can be imported and exported from your modeling tool to make changes without losing SIW concept annotations.

Installing/Verifying Installation of Java Web Start

The SIW is a Java Web Start application. If Java Web Start is installed on your computer, the SIW will launch when you open the SIW URL. SIW always checks to be sure you are using the most current version of Java Web Start. If you are not, the SIW automatically retrieves the update for you.

If you do not have Java Web Start installed, you will receive a File Download box when you attempt to access the SIW URL.

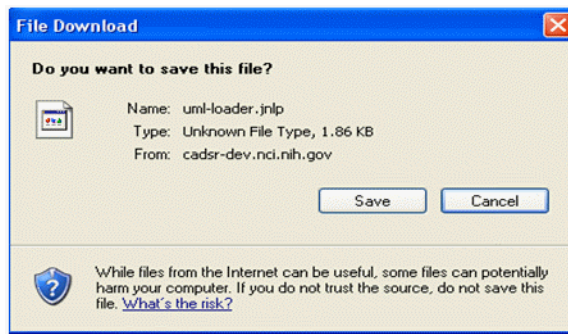


Figure 3.1 SIW File Download dialog box

If you receive this dialog box, it means that the SIW did not find the application needed to open the SIW file. Click **Cancel**, and go to the Sun web site (<http://java.sun.com>) to download and install Java Web Start.

Launching the SIW

Once you have a valid UML model file to use, and that you have Java Web Start installed, you can begin using the SIW.

The instructions in this section and throughout this chapter use the “browser-launch” method of accessing the SIW. This is done purely for convenience and you do not need a browser to access the SIW. If you prefer, you may use any of the following methods:

- On UNIX or LINUX, type `javaws` at the command prompt
- On Windows, launch the Java Webstart application (the program may appear in your Start menu)
- Save the `siw.jnlp` file to your computer and double-click it to launch the SIW.
- Open the SIW URL: <http://cadsrsiw.nci.nih.gov/>.

To launch the SIW:

1. Access the following URL using your browser: <http://cadsrsiw.nci.nih.gov/>.

2. The Java Web Start dialog box appears, showing the Semantic Integration Workbench start-up progress. If you have never used the SIW or not used it in some time, this may take a few minutes.

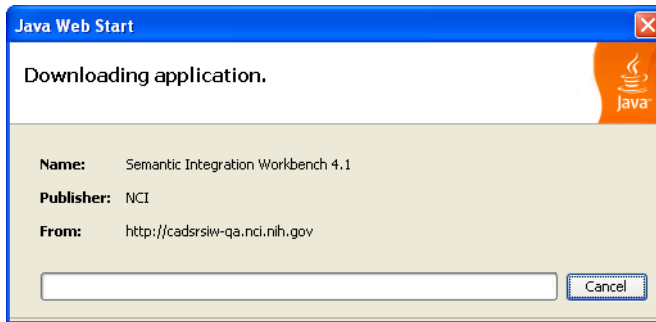


Figure 3.2 Java Web Start dialog box and progress bar

3. If you receive a Security Warning dialog box, enable the **Always trust content from this publisher** checkbox and then click **Run**.



Figure 3.3 Security Warning dialog box requesting approval to run the application

Note: During installation, portions of the application are downloaded to cache, but the entire application itself is not downloaded.

After the start-up process completes, the SIW opens, starting with the SIW Welcome screen.

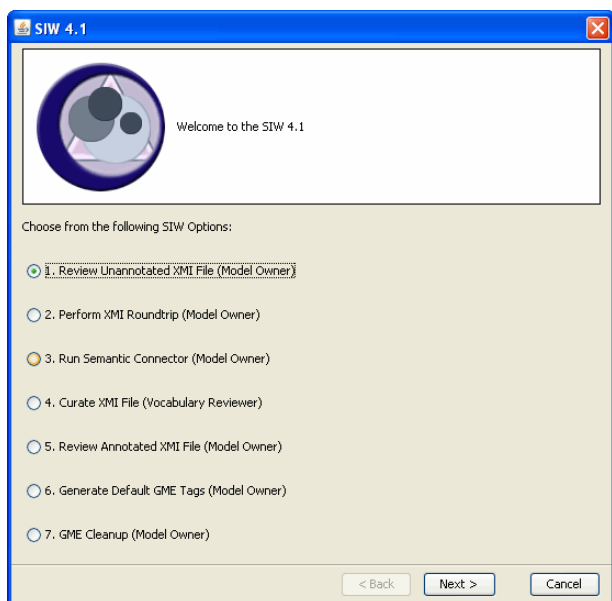


Figure 3.4 Semantic Integration Workbench Welcome Screen

The top of the SIW Welcome screen shows which version of the SIW you are working with (it should be the latest release version). The main portion of the screen lists the available options or steps for using the SIW.

SIW Steps/Options

For the most part, the SIW steps should be performed in the order in which they appear on the Welcome screen. However, some of the steps are optional and only apply if you are working with a new version of a previously loaded XMI file, or if you intend to have your model uploaded to the caDSR or to Grid-enable your service.

That being said, there are essentially two kinds of model owners: those who want/need the EVS Curation team to perform the majority of the concept mapping for their model, and those who want to perform as much of the mapping as possible themselves before handing the model off to the EVS curators.

If you plan to ask the curation team to do most of the concept mapping for you, the order of the SIW steps you will use is roughly as follows:

1. Review Unannotated XMI File: Use this step to review the XMI file for any errors that must be fixed in the model itself and re-exported to XMI.
2. Perform XMI Roundtrip: This step only applies if you have a new version of a previously registered model or you based your model on another registered model. Otherwise see the next step.
3. Run Semantic Connector: Use this step to allow the SIW to automatically map EVS concepts to your model elements based on the elements names.

After running the Semantic Connector, you can hand the file off to the EVS Curation team, allowing them to work through and fix the automated annotations. After they are

finished, they will hand the file back to you for review and approval of their mappings, after which you can use the SIW to add GME Namespace tags if appropriate.

If you plan to work through as much of the concept mapping yourself as you can, the order of the SIW steps you will use is roughly as follows:

1. **Review Unannotated XMI File:** Use this step to review the XMI file for any errors that must be fixed in the model itself and re-exported to XMI.
2. **Perform XMI Roundtrip:** This step only applies if you have a new version of a previously registered model or you based your model on another registered model. Otherwise see the next step.
3. **Run Semantic Connector:** Use this step to allow the SIW to automatically map EVS concepts to your model elements based on the elements names.
4. **Review Annotated XMI File:** Use this step to review the automated concept mappings done by the Semantic Connector. Through this step you can add more concepts, remove unneeded concepts, and change the order of the mapped concepts. Keep in mind that the Semantic Connector cannot “understand” the meanings of the elements in your model. It bases its mappings on the names. It requires human intervention to determine which are correct.

After reviewing and fixing the automated mappings, you can hand the file off to the EVS Curation team, allowing them to work through and fix the remaining annotations. After they are finished, they will hand the file back to you for review and approval of their mappings, after which you can use the SIW to add GME Namespace tags if appropriate.

All of options available through the SIW are identified and described briefly in the sections that follow. The remaining chapters of this guide provide more detailed instructions on using each of the SIW options.

Review Unannotated XMI File (Model Owner)

This step allows the model owner to view the XMI representation of their UML model. It provides an easy way to check the model for missing object definition tags or other problems with the file that will require changes before continuing. This step should be performed with all newly exported files before running any other SIW options.

When you view an XMI file in “Unannotated” mode, the errors listed include only syntactic errors, such as missing element descriptions. This allows you to easily see where additional information must be added to or changed in order for the SIW to successfully annotate the model (map to EVS concepts or caDSR data elements) in later steps.

See [Chapter 4, Reviewing An Unannotated File](#), on page 51 for more information on this step.

Perform XMI Roundtrip (Model Owner)

This step automatically annotates the XMI file (or updates existing annotations) based on another previously loaded model. If you have a previously registered version of your system, or you based your system on a previously registered model, the Roundtrip task can save you a considerable amount of time by automatically mapping your XMI file's

attributes to the caDSR common data elements (CDEs) used by the registered model you identify for the Roundtrip task. The automated matching is based on the new model having used exactly the same attribute names as the identified model.

When running roundtrip you can select whether or not to have the SIW automatically insert GME namespace tags for the reused elements in your model that point to the existing XSD schema for those elements. This is done by unchecking the **Exclude Namespaces from Roundtrip** checkbox. In this case, the following UML elements will have tagged values inserted by roundtrip:

- Packages (optionally)
- Classes (XMLElement. XMLNamespace roundtrip is optional)
- Attributes
- Associations

If the GME alternate names do not exist in caDSR, or you select to Exclude Namespaces from Roundtrip, the XMI Roundtrip mode will not insert anything back into the input file.

Since it is expected that model owners may run roundtrip more than once for a given file, especially when a model reuses portions of another model, the XMI Roundtrip step will insert GME tags for the reused projects. The results of this may be that one model contains packages that map to different namespaces.

The SIW will only insert GME tags if those tags do not already exist in the model. The SIW will not update or replace existing GME tagged values.

If you plan to use GME namespace tags in your model, you are strongly encouraged to review [Chapter 9, Using GME Annotations](#), on page 109. You should also read [Sample Scenarios for Using GME Options](#) on page 111 to determine what order the SIW steps should be run in order to generate the appropriate GME annotations for your model.

See [Chapter 5, Using XMI RoundTrip](#), on page 59 for more information on performing this step.

Run Semantic Connector (Model Owner)

This step launches the Semantic Connector, which performs an EVS search against the name of each element in the XMI file (using camel case to identify those elements with multiple parts). When matches are found, the SIW attaches one or more EVS concepts to each element. This process produces an annotated XMI file (with “FirstPass” appended to the front of the file name) that can then be reviewed by the owner (if desired) before being submitted for curation.

See [Chapter 6, Using the Semantic Connector](#), on page 67 for more information on performing this step.

Curate XMI File (Vocabulary Reviewer)

This step is performed by the EVS concept curation team. During this step, the EVS team uses new or existing EVS concepts to annotate the model, and adds and removes concepts as they determine necessary. The curated XMI file is then returned to the model owner for review and final approval.

This and the next step of the SIW process are typically the most iterative, meaning that very often, the file is returned to the curation team after model owner review, and then back to the model owner after re-curation and verification.

See [Chapter 7, Curating An XMI File](#), on page 73 for more information on this process.

Review Annotated XMI File (Model Owner)

This step, along with the Curate XMI File step (above) are typically performed several times until all of the model elements are mapped properly. This step allows model owners or model reviewers to review element annotations, search EVS for concepts, and where necessary, change the mapping between a class or attribute and its EVS concept(s). Skilled users may also choose to map annotated elements to existing caDSR CDEs and/or value domains.

This mode of the SIW also provides the ability to run validation checks to ensure that the concept information in the XMI file corresponds with EVS concept information, or that any differences between the element descriptions in the XMI file and the mapped EVS definitions are determined (by the model owner) to be appropriate and valid.

After curation and review, the model owner can “verify” each element in the file. Once all elements have been verified by the model owner, the file is considered to be complete. The final outcome of this step is a fully annotated XMI file that can be used to register the model into caDSR, and which can also be used as input to the next version of the model.

See [Chapter 8, Reviewing An Annotated File](#), on page 91 for more information on performing this step.

Generate Default GME Tags

The Generate Default GME Tags step of the SIW creates GME namespace tagged values in the XMI file that contain URLs that identify the XSD schema for your model.

If you use SIW to create default GME tags, the SIW wizard will ask you for the Project Name, the Project Version, the Context where your model will reside, and the top level package where GME tags are to be added.

GME information can be added at the model, package, class, attribute and association level. Currently, the Generate Default GME Tags option add tags at all of these levels. At the association level, however, only those roles that have names will be tagged. In other words, only the Source and Destination ends of an association that have been given role names will be tagged with GME namespace tags.

If you decide you don't want to use the defaults, then you can specify the actual URIs using caAdapter.

In the SIW Viewer, you can review the GME tags for:

- Package (XMLNamespace)
- Classes (XMLElement)
- Attributes (XMLLocReference)
- Associations (SourceXMLRefLoc and TargetXMLRefLoc)

For each of these, the SIW provides a read-only text field allowing you to review this information. If you want to modify the values, however, you must do so either in the modeling tool or in caAdapter.

If you plan to add GME namespace tags to your model, you are strongly encouraged to review [Chapter 9, Using GME Annotations](#), on page 109. For additional details and information regarding GME annotations, see the SIW wiki page devoted to the GME Namespace tags: <https://wiki.nci.nih.gov/x/klAl>.

You may also see [Generating Default GME Tags](#) on page 113 for the procedures for performing this step.

GME Cleanup

Once a GME tag exists in the XMI file, SIW will not override it. If you wish to recreate GME namespace tags for your model, you should first use the GME Cleanup functionality. This removes all GME tags from the XMI file. This is useful for instances when an XMI file is fully annotated with GME tags, but you want to replace those tags, or let SIW regenerate default values.

Keep in mind that both the Roundtrip step and the Create Default GME Tags step will only create GME tags for those components that do not already have these tags. By using GME Cleanup prior to running either of these steps, you can guarantee that all of the components that need the new tags will have them added.

If you plan to add GME namespace tags to your model, you are strongly encouraged to review [Chapter 9, Using GME Annotations](#), on page 109.

See [Clearing GME Tags](#) on page 117 for the procedures for performing this step.

Package Filtering

All of the SIW options described above can be run on a full XMI file, or you may choose to run each step on only selected elements of the file. When you identify the file to parse, you may select to **Choose Classes and Packages**. Selecting this option adds a screen to the processing wizard that lists the Packages within the XMI file. Expanding a Package shows the classes contained within that package and lets you identify only those you want to process for that step.

Using the SIW Viewer

The SIW viewer is the window that appears whenever you select to review or curate your XMI file. The viewer shows the model elements parsed from your XMI file and details about each of those elements. What details you see depends on several things, including the current state of your XMI file (unannotated, annotated, curated). However, the basic attributes of the SIW viewer remain the same, regardless of where you are in the semantic integration process.

This section discusses navigation as well as the functionality available through the SIW viewer, including setting Viewer Preferences. Since the viewer is used to both review and make changes to your XMI file, detailed information regarding how to use specific functionality in the viewer resides in the sections that discuss the steps in which those actions should take place.

Topics in this section include:

- [Navigation Tree](#) on page 36
- [Detail View](#) on page 37
- [Errors/Log Panel](#) on page 42
- [Setting Viewer Preferences](#) on page 44.

The SIW viewer window has three sections: the navigation tree, the detail view and the errors/log panel.

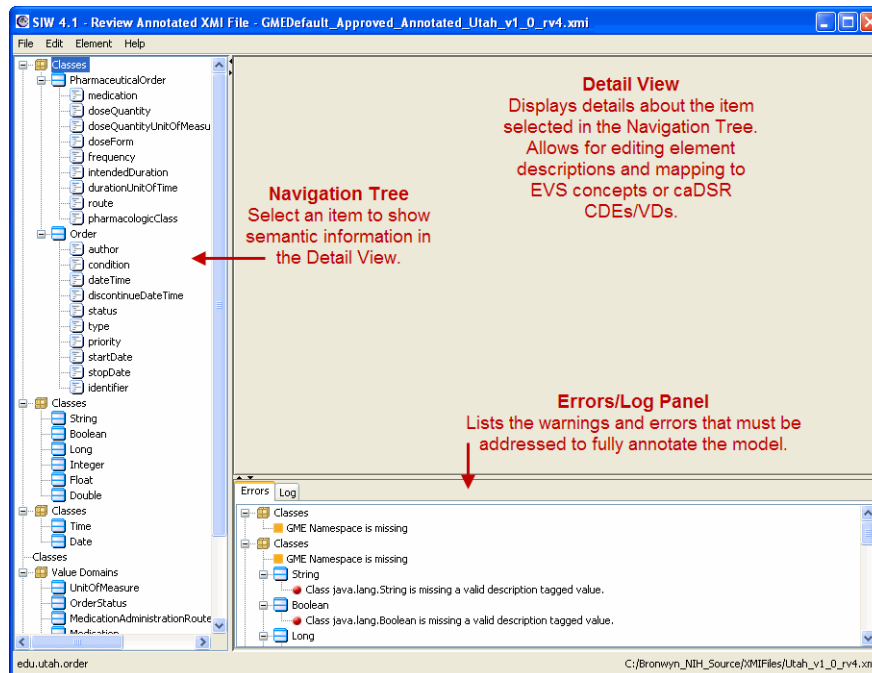


Figure 3.5 SIW Viewer with sections labeled

As is typical of “Windows Explorer-like” applications, you can change the size of the sections of the window by hovering your mouse over the border of that section until the mouse cursor becomes a two-way arrow, and then clicking and dragging the border in the direction of the change you want to make.

Navigation Tree

The navigation tree, located on the left side of the viewer, is a tree-view of all of the elements resident in the model (or packages if only a portion of the file was processed). Selecting an element from the tree opens a tab in the detail view, showing detailed information for the selected item.

If you are viewing a curated or annotated file, the navigation tree may include checkmarks on elements in the file indicating that the mappings for those elements have been reviewed and checked as “Human Verified” or “Model Owner Verified.”

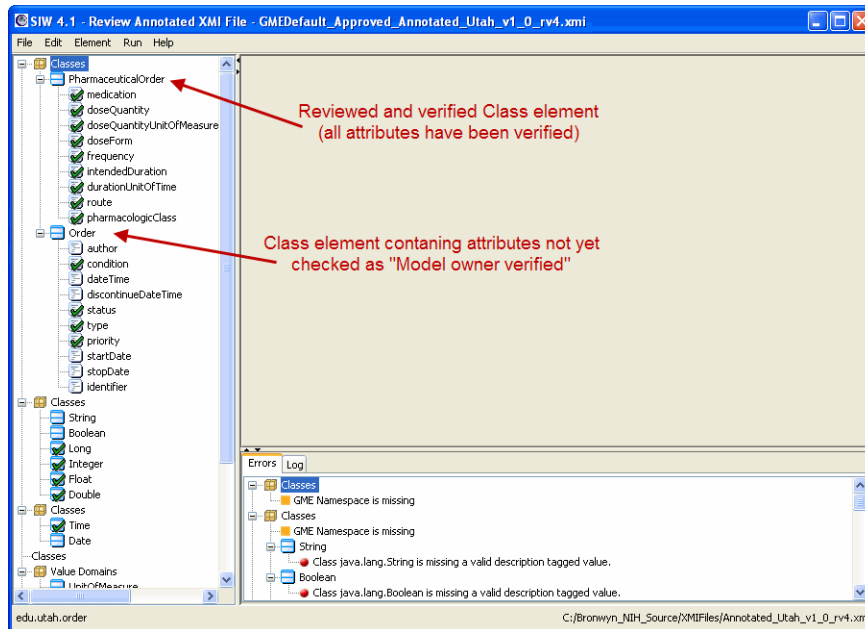


Figure 3.6 Navigation Tree showing fully verified and partially verified classes

When the viewer first opens, all nodes of the tree are expanded to display all internal nodes, however each node can be expanded or collapsed by clicking the + (plus) or - (minus) sign to the left of the node.

By default, navigation tree items are listed in the order of the structure of your model as indicated in your XMI file. In addition, classes are listed first, with associations appearing at the bottom of the tree. You can change these aspects of the navigation tree display using SIW Viewer Preferences (available through the Edit menu in the viewer). See [Setting Viewer Preferences](#) on page 44 for more information.

Detail View

The detail view is located in the top-right section of the viewer and is blank when the viewer first opens. Highlighting an element in the Navigation tree opens a tab for that item in the detail view. What information appears in the tab depends on what type of item is selected (class, attribute or association) and the current state of the file you are working with (annotated, unannotated, curated, reviewed, etc.).

For example, if you click on a class or attribute node, the detail view may show only UML description text or it may show concept and value domain information, depending on whether or not the selected item has been annotated with concept information yet.

Navigation Note—The detail view can contain multiple tabs at any given time, and will open a new tab for different types of elements selected in the navigation tree. The title on each tab corresponds to the node information it contains.

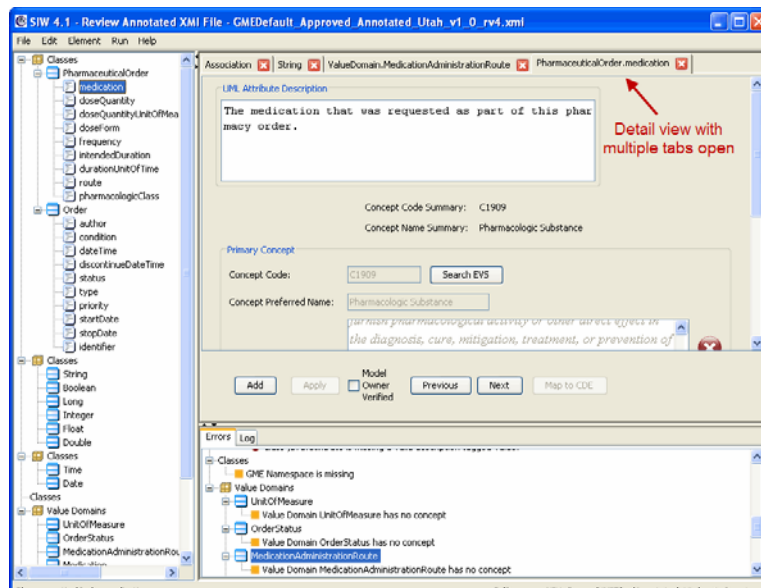


Figure 3.7 SIW Viewer with multiple tabs open in the Detail View

To close a tab, click the “X” located to the right of the node name on the tab.

Besides definition and/or concept information about the node selected, the detail view also contains buttons that perform a variety of tasks. Some buttons, like **Map to CDE**, do not appear until the file and/or the node selected has reached a certain level of annotation. Other buttons always appear and function as follows:

- **Previous** - Changes the selection in the navigation tree to the item immediately preceding (above) the currently displayed element.
- **Next** - Changes the selection in the navigation tree to the item immediately following (below) the currently displayed element.
- **Add** - Opens blank fields for entering concept information, allowing you to add a Primary Concept or Qualifier Concept (if a Primary Concept is already mapped) to the item.
- **Search EVS** - Opens a search box allowing you to search the NCI Thesaurus for concept information to add or update for the item. The **Search EVS** button always appears next to concept code field, either for existing mapped concepts or as a function of clicking the **Add** button (to populate blank concept information fields).
- **Apply** - Applies the changes made to the concept information. The Apply button is disabled until all concept fields are populated for newly added concepts, or until a change is made to existing concept information.

You must click **Apply** before navigating away from the changed item's node, or your changes will be discarded. The SIW prompts you to apply your changes if you attempt to view another node before clicking **Apply**.

Important: Clicking **Apply** is *not* the same as saving the file. Apply simply applies your changes to the currently open session. You must still save the file (**File > Save** or **File > Save As**) to create an updated XMI file containing the applied changes.

The following sections describe what information appears in the detail view and how that information is displayed, depending on the type of node selected and the information available for the selected item.

Viewing an Unannotated XMI File

An “unannotated” XMI file is one where none or some of the classes and attributes in the file have been annotated with EVS concepts. If you are working with a new version of a new system that has never been through the caCORE semantic integration process, it is likely that your file contains no annotation. If a previous version of your UML model has been through semantic integration, some (even most) of your model elements may be mapped to either EVS concepts or to caDSR data elements, however there are likely new elements that need to be annotated.

When viewing an unannotated file, if you select a class or attribute from the navigation tree and the detail view remains empty (no information appears), this indicates that the element does not have a description tag in the model. Model owners must provide descriptions for each element in the model.

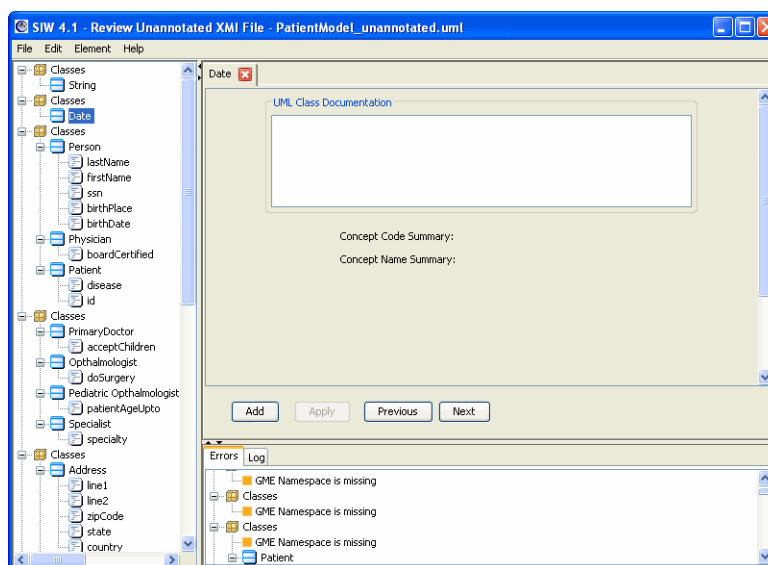


Figure 3.8 Class element selected, with no concept information or UML definition

UML element descriptions are required because they allow EVS curators understand the model owner’s purpose and intent for the element. This provides for more accurate mapping to EVS concepts and for easier creation of new concepts when necessary.

The element descriptions in the UML model should accurately describe the element to which they are attached. There are two ways to apply these descriptions to the elements in your model: adding tagged values to the model, or entering the description in SIW, which creates these tags for you.

When creating the model itself, you can use a tag called **CADSR_Description** to tag these elements in the model. If this tag is present, the SIW automatically uses this information to populate the UML element description field. This allows model owners to

differentiate caDSR-specific descriptions from other descriptors used for the element in the model. If the model element does not have any CADSR_Description tags, the other existing description tags are used to provide a description for the element.

If you open your XMI file in SIW and see that some or all of your element descriptions are wrong or missing, you can enter or edit those descriptions directly in the UML element description field in SIW. Later, if you re-import the annotated XMI file back into your model, the elements whose description you changed will have new/edited CADSR_Description tagged values associated with them.

If you select a class or attribute node and a UML description appears, but there is no concept information, this means that the element contains a description in the UML model, however the item has not yet been mapped to an EVS concept.

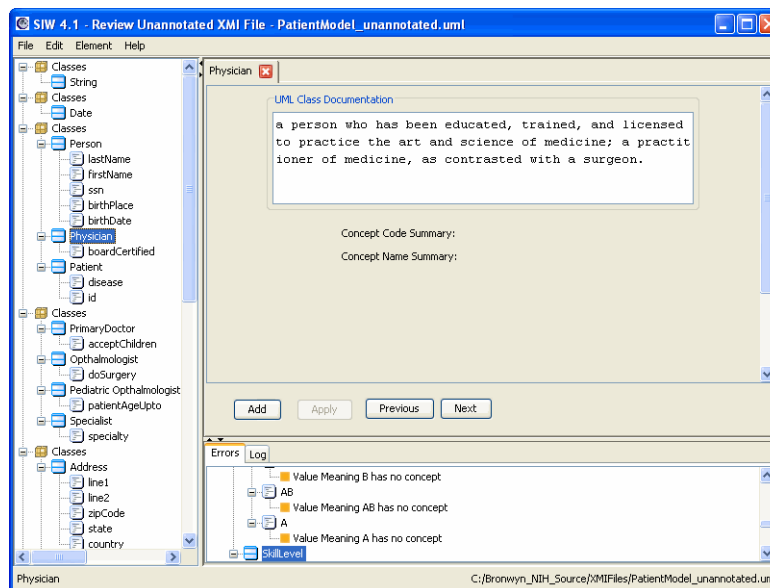


Figure 3.9 Class element selected showing a UML definition but no concept information.

For these elements, you can manually add concept information to the element (by clicking **Add**), or wait to process the file through the Semantic Connector to have the element mapped for you (using the closest match to the element name found in EVS).

Viewing an Annotated XMI File

An “annotated” XMI file is one where all of the elements in the file have been mapped to either EVS concepts or directly to caDSR data elements. If you select a class or attribute from the navigation tree and the detail view displays concept information for the element, this means that the class or attribute has been mapped to one or more concepts in EVS or “annotated.”

By default, concept information appears in “semantic order” meaning that it appears in an order consistent with how the concept names will be positioned when forming the name of the element in caDSR.

Each element in a model must be mapped to a Primary Concept that identifies the basic or essential meaning of the element. If the full meaning of the element cannot be described with a single concept, then Qualifier Concepts are added to the mapping.

The number of Qualifier Concepts mapped depends on the complexity of the element's meaning or use within the model.

For example, a model has a class element called “Study” that contains several attributes, one of which is named “additionalPatientHistory”. The class element “Study” likely has only a single concept mapped to it; a concept with a definition of “study” similar to that provided for the element.

However the attribute “additionalPatientHistory” will likely have multiple concepts mapped to it, one for each of the ideas or meanings indicated by the parts of the attribute name: *additional*, *patient*, and *history*. In this case a Primary and two Qualifier concepts.

The Primary concept probably represents “history”, because according to the definition, that is the main point of this element. The first qualifier concept represents “patient” and the second qualifier concept represents “additional”. Again, this mapping is done based on the element descriptions provided by the model owner.

When displayed in the SIW viewer, the Qualifier concepts are shown first with the Primary concept appearing last. This is done by default to show how the concept mappings work together to provide the same meaning as the model element.

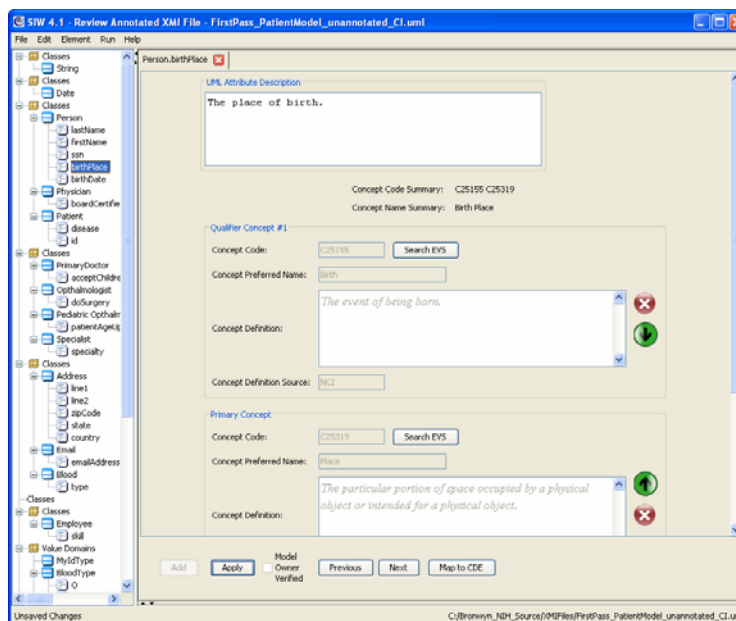


Figure 3.10 Attribute element selected, showing mapped concept information

The detail view shows the element description at the top, followed by the Qualifier concepts (if there are any) with the Primary concept below that. The value domain (if mapped) appears at the bottom. Notice that the Concept Summary Code and the Concept Summary Name are the codes and names of each of the mapped concepts in semantic order (qualifier(s) + primary = concept name). This Concept Summary Name should match or closely match the actual name or intent of the element.

If you prefer, you can change the order in which concept information appears in the detail view using options available in the Viewer Preferences. See [Setting Viewer Preferences](#) on page 44 for more information.

Navigation Note—The concept information section of the detail view will likely scroll (and may scroll significantly) because it contains more information than can be displayed at once. [Figure 3.10](#) above shows an expanded detail view, in order to display as many pieces of the concept information as possible. However, if you cannot see all of the concept information for an element, scroll to see if the information lies above or below the visible portion of the view.

The detail view also contains additional buttons located to the right of the concept definition fields: up and down arrows and an “X” button. More detailed information about these buttons appears in [Changing Concept Information in Review Mode](#) on page 95.

Viewing Associations

By default, associations appear at the bottom of the navigation tree. If you select an association node from the navigation tree, four tabs appear in the detail view: Detail, Role, Source, and Target. The elements defined for an association (the Role, the Source and the Target) can be annotated with concepts just like other elements in the model, although this is not required for the file to be considered fully annotated.

The **Detail** tab shows the connection details for the association, including direction, source class, source role, source multiplicity, target class, target role and target multiplicity.

If the associated elements are annotated with concept information, the tabs corresponding to those elements display that concept information. Those tabs also contain the same buttons as other element detail tabs (**Previous**, **Next**, **Add**, **Apply**, and **Search EVS** where applicable). This allows you to manually map those association elements to EVS concepts if desired.

While you can manually map an association’s elements to EVS concepts, you cannot change an association’s syntactic description through the SIW.

Errors/Log Panel

Located in the bottom-right section of the viewer, the errors/log panel provides two tabs containing information about the file currently open in the viewer. The Errors tab lists all errors and warnings flagged during processing of the XMI file.

Located behind the Errors tab in the bottom-right section of the viewer is the Log tab, which provides a log of the file parsing process.

Errors Tab

One valuable feature of the SIW is the ability to quickly isolate and identify errors in the source files, primarily where there is missing or inaccurate information. When the SIW encounters issues while processing a file, those items are flagged as either errors or warnings, and are then displayed in the Errors tab at the bottom of the SIW viewer. The Errors tab provides a description of the error and identifies the source of the error.

The tree structure on the Errors tab shows all errors relevant to a class item as sub-nodes of the class node. When you select a class item in the Errors tab, SIW navigates to that element in the Navigation Tree allowing you to easily review and address all identified issues for the class.

Note: The error tree does not appear if the file contains no errors.

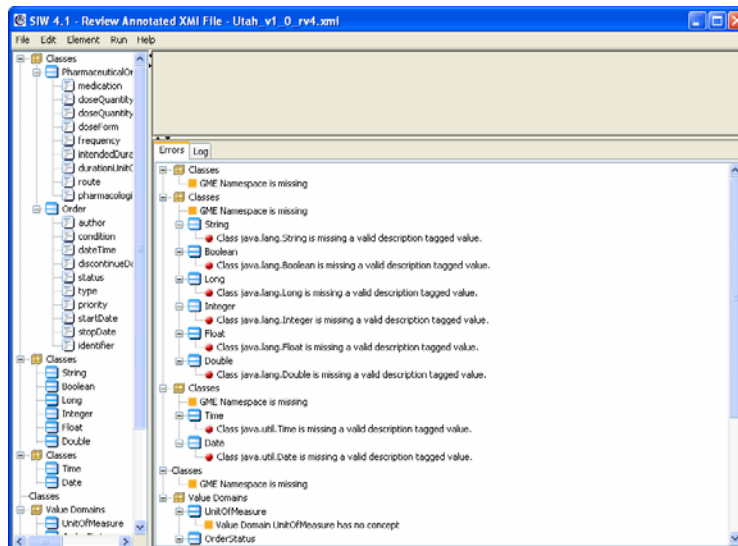


Figure 3.11 SIW Viewer showing an expanded Errors tab

The SIW performs checks for five basic types of errors:

- **Concept Errors** - There are several validation rules that are applied to each model. Any violations of these rules appear in the Errors tab with the source of the error and a description of the problem. Concept errors are most often triggered by missing class or attribute information (missing UML description tagged values or missing concept information).
- **Duplicate Errors** - If two Object Classes (OCs) have the same Public Id or Preferred Name a duplicate mapping error will appear. A duplicate mapping error can also occur if two data elements belonging to the same OC have the same Preferred Name.
- **Association Errors** - Associations that are missing names on the end of the association that has an arrowhead will cause an error. For example, you need a target name for source-->destination associations; you need a both a target name and a source name for bi-directional associations (source <-> destination).
- **Value Domain Errors** - If a definition, Vdtype, datatype, Public Id, or Version are either missing or invalid, an error will appear. If a value domain is in the model but is not used by a data element in the model, this will cause a warning message to appear. Finally, if two value domains are mapped to the same concepts this will also cause an error.
- **GME Errors/Warnings** - If a GME Tag exists but the GME Namespace is not a valid URI, the SIW reports an error. If the GME tag is missing, the SIW reports a warning (because GME tags are not required).

The information in the Errors tab can be exported by right-clicking in anywhere in the Errors tab and selecting **Export Errors**.

Important Note about updating the Errors tab—The Errors tab information is generated at the time you open the file; the SIW does not update the error list while you are editing. If you have made changes want to regenerate the error listing, save the file and exit the SIW. Then re-open the SIW and select the file you saved. This updates the Errors tab information to reflect the status of the updated file.

Log Tab

The Log tab provides detailed parsing and runtime information about the currently open file. Each log event is represented as a line in the log tab. Log events are organized into four categories:

- Errors: These include errors encountered in the SIW, in the model being parsed or both.
- Warnings
- Info: Events that may be helpful to users
- Debug: Information that may be useful to developers.

Right-clicking in the Log tab allows you to add or remove any of these categories from the tab. This may be useful if you need to see only certain types of log information.

Setting Viewer Preferences

The SIW viewer allows users to set preferences for using the SIW. Most of the Preferences options available have to do with how the information in the viewer is presented, however a few of the options control the functionality of certain features available through the viewer.

To open the SIW Viewer Preferences dialog box, select **Edit > Preferences**.

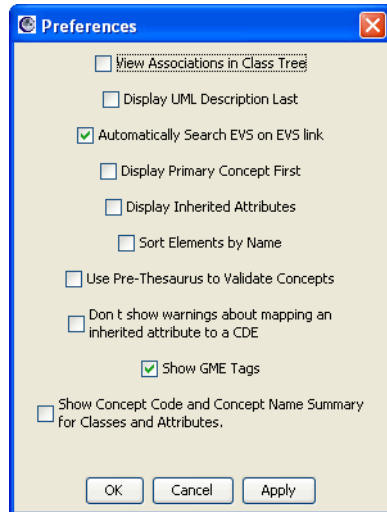


Figure 3.12 SIW Viewer Preferences dialog box

By default, the **Automatically Search EVS on EVS Link**, **Show GME Tags**, and **Show Concept Code and Concept Name Summary for Classes and Attributes** options are enabled. All others are disabled by default. All of the options available are described in more detail in the sections that follow.

View Associations in the Class Tree

By default, model associations are listed all together at the bottom of the navigation tree. Enabling the **View Associations in the Class Tree** option in the SIW Viewer Preferences changes the navigation tree view so that associations appear as children to (nodes beneath) their related classes. This is useful when trying to identify the associations for a particular class.

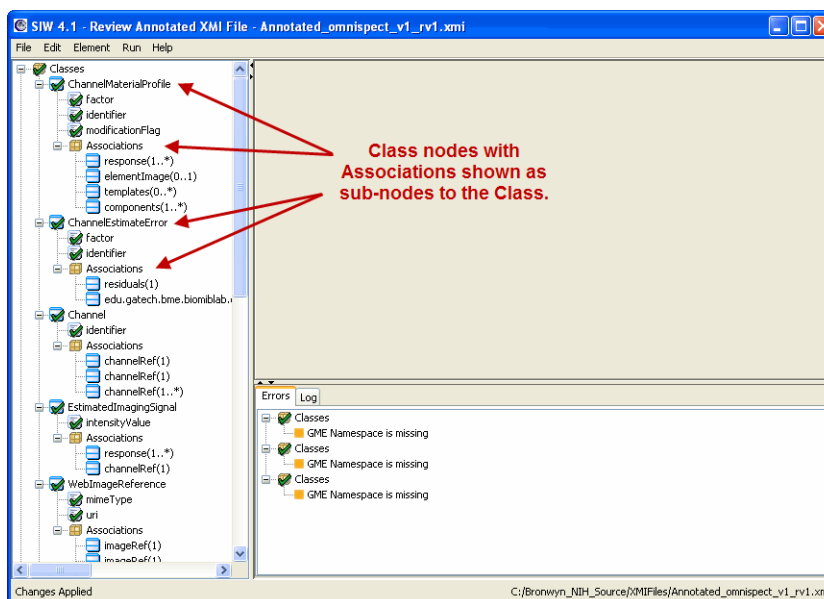


Figure 3.13 SIW Viewer with Associations listed under the Class nodes in the Navigation Tree

Note: The View Associations in the Class Tree option does not appear in the Curate XMI mode of the SIW because EVS curators do not work with association elements, and therefore associations do not appear in the SIW viewer in this mode.

Display UML Description Last

As noted in the [Detail View](#) section of [Using the SIW Viewer](#) on page 35, by default the UML descriptions for each element are listed first in the detail view, followed by the concept and value domain information where present.

Enabling the option **Display UML Description Last** in the viewer Preferences changes the display so that the UML description appears at the bottom of the detail view instead of the top.

Automatically Search EVS on EVS Link

The **Automatically Search EVS on EVS Link** option is enabled by default. This means that if there is text in the Preferred Concept Name field when you click **Search EVS**, the SIW automatically searches EVS for a synonym of that term and returns the results if any are found.

When this preference is disabled, the search dialog box appears, but no search is run without manually entering search criteria and clicking Search.

Note: If you disable this option and then click **Search EVS**, the previous search information may appear in the search dialog box. This information is NOT cleared between searches. Replace the Search text box text and click Search to search EVS.

Display Primary Concept First

As noted in the [Detail View](#) section of [Using the SIW Viewer](#) on [page 37](#), by default the Primary Concept information for each element is listed below any Qualifier Concepts mapped to the item, which mimics the semantic order for the name of the element.

Enabling the option **Display Primary Concept First** in the viewer Preferences changes the display so that the Primary Concept appears above the mapped Qualifier Concepts. In addition, if there are multiple Qualifier Concepts for the item, they are now listed in numerical rather than semantic order.

The UML description still appears at the top of the Detail view for each element unless you have enabled the **Display UML Description Last** option (see [Display UML Description Last](#) above).

Display Inherited Attributes

Enabling the **Display Inherited Attributes** changes the navigation tree to show the inherited attributes for any class that inherits attributes from a super class. The attributes appear in an **Inherited Attributes** node below the class.

Enabling this option is useful for if you know you have inherited attributes for class items, in order to call attention to them.

Sort Element by Name

As noted in the [Navigation Tree](#) section of [Using the SIW Viewer](#) on [page 36](#), by default the elements in the navigation tree appear in the order in which they exist in the UML model (as indicated in the exported XML file).

Enabling the option **Sort Element by Name** in the viewer Preferences changes the display so that the elements in the navigation tree are sorted alphabetically. Each class

appears in alphabetical order (by name), with each of the elements within the class also listed alphabetically.

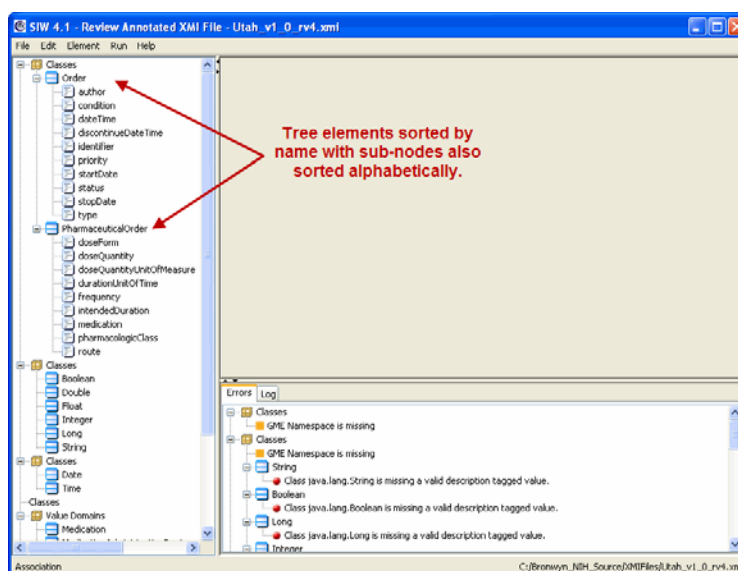


Figure 3.14 SIW Viewer with Navigation Tree nodes sorted alphabetically by name.

Deselecting this option resets the navigation tree to again display the classes and their associated elements in the order they exist in the XML file.

Use Pre-Thesaurus to Validate Concepts

By default, if you select **Validate Concepts** from the Run menu (available in either the Curate XML File mode or Review Annotated XML File mode), the SIW validates the concept information in the XML file against the information in the NCI Production Thesaurus. Enabling the **Use Pre-Thesaurus to Validate Concepts** option causes the SIW to use the Pre-Production NCI Thesaurus to validate concepts instead.

The Pre-Thesaurus is pre-release version of the next release of the NCI Thesaurus, publicly accessible through the DTS server. As is implied, the Pre-Production Thesaurus is more recent than the Production version and as such contains newer concepts. Changing this option in your Viewer Preferences may be useful if you are working with an XML file that you know contains mapping to newer EVS concepts.

Don't show warnings about mapping an inherited attribute to a CDE

If you have inherited attributes in your model, and you attempt to map an inherited attribute to a CDE, by default the SIW displays an error message informing you that if you map the attribute to a data element, the corresponding class will automatically be mapped to the corresponding object class.

If you are aware of this correlation and plan to map the inherited attributes to CDEs anyway, you may want to hide the warning message. Enable this option in the Viewer Preferences to tell the SIW to not show this message.

Show GME Tags

If your XMI file has had the GME tags generated for it, you can view those tags in the SIW viewer when you open your annotated file. GME tags are shown by default and appear for elements below the mapped concept information.

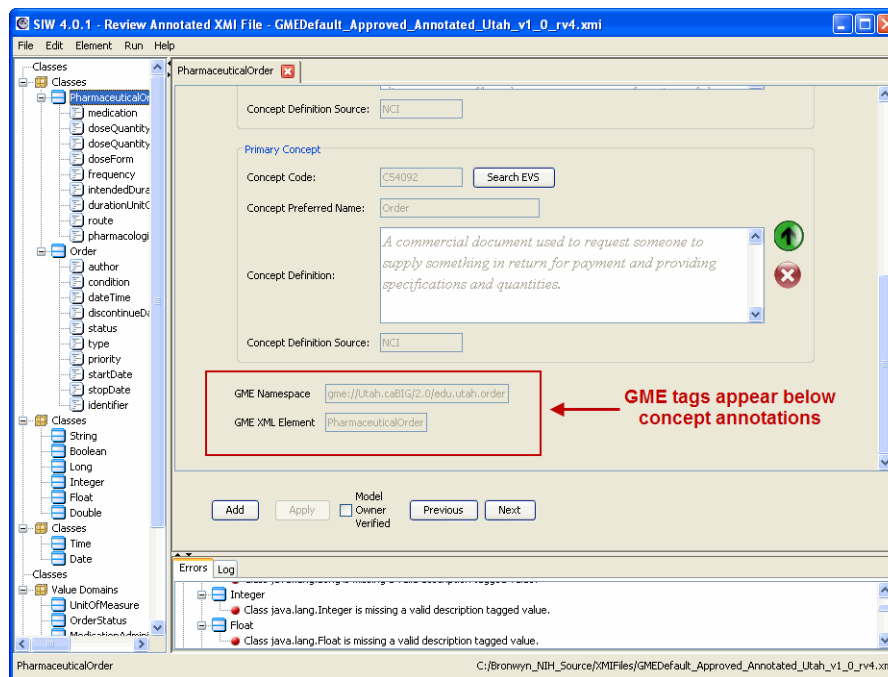


Figure 3.15 SIW Viewer with GME tags shown for Class element

If you do not want to view the GME information for the model, de-select the Show GME tags option in the SIW Viewer preferences.

Show Concept Code and Concept Name Summary for Classes and Attributes

By default, the Summary Concept Code and Summary Concept Name appear below the element description in the detail view for classes and attributes. You can, however deselect (uncheck) this Viewer Preference and hide the concept summary information from the view.

Saving Changes to a File

Clicking **Apply** after making changes to an element is not the same as saving the XMI file. Apply simply applies your changes to the currently open session. You must still save the file using the **File > Save** or **File > Save As** commands to create an updated XMI file containing the applied changes.

Use **File > Save** to overwrite the original file you opened in the SIW viewer. Use **File > Save As** to specify a different name or location for the file.

The Status Bar at the bottom of the SIW viewer window informs you whether or not the file was successfully saved.

XMI files that were created using Enterprise Architect are always saved using an “.xmi” file extension. XMI files created using ArgoUML are always saved using a “.uml” file extension.

Curated and verified files should be saved with the term “annotated” appended to the front of the filename. For example, if the original file name was `FirstPass_myModel.xmi` then you would save the file as `Annotated_FirstPass_myModel.xmi`.

Reviewed and verified files should be saved with either the term “approved” or “fixed” appended to the front of the filename.

Use “fixed” if you are annotating your file before handing it off to the Curation team, or if you have made changes to file based on recommendations by the Curation team and are sending it back to them for another pass.

Use “approved” if you are simply reviewing and marking the concept mapping done by the Curation team as Model Owner Verified.

For example, if the original file name was `Annotated_FirstPass_myModel.xmi`, then you would save the reviewed file as `Approved_Annotated_FirstPass_myModel.xmi`.

Files that have been processed through the Generate Default GME Tags step are automatically saved with the term “GMEDefault” appended to the front of the filename. For example if the original file name was `Annotated_FirstPass_myModel.xmi`, then the new file with the GME tags is automatically saved in the same location as `GMEDefault_Annotated_FirstPass_myModel.xmi`.

Files that have been processed through the GME Cleanup step and had their GME tags removed are automatically saved with the term “GMECleanup” appended to the front of the filename. For example, if the original file name was `Annotated_FirstPass_myModel.xmi`, then the new file with the GME tags removed is automatically saved in the same location as `GMECleanup_Annotated_FirstPass_myModel.xmi`.

Updating UML Model Definitions with SIW-Generated Changes

One problem frequently encountered in the semantic integration process is the discovery that the element description tags from the UML model are not present, or that they need to be updated. Prior to the SIW, there was no way to conveniently get these changes into the XMI file for loading to caDSR. Since these tags are mandatory, these tags must be present and should be as accurate as possible.

As is described in [UML Element Descriptions and Annotations](#) on page 56, the SIW recognizes the `CADSR_Description` tag for use as element description tagged values. You can either add these tagged values to the model before exporting to XMI for the SIW, or you can enter these descriptions through the SIW interface.

If you use the SIW interface to add or update your model’s element descriptions, you can then save the file from SIW and re-import it back into your modeling software. Both the element descriptions and all EVS concept curation is maintained and is now a part of the model information. This capability helps streamline the process of creating a caCORE-compatible system.

Entering or editing the element description information in the SIW generates `CADSR_Description` tagged values for the element. You can have up to eight of these tags per element if needed (EA imposes a limit of 255 characters per tag). When re-

importing the XMI file back into your UML modeling software, the elements in the model will be annotated with these tagged values.

CHAPTER 4

REVIEWING AN UNANNOTATED FILE

This chapter provides detailed information on how to use the SIW to view and review your initial XMI file. The error log in unannotated mode does not show concept errors, because logically, an unannotated file does not have completed concept mapping. However the error log will show other issues with the XMI file that must be addressed at some point in the semantic annotation process. Addressing these issues before diving into concept mapping and annotation helps streamline the entire process.

Topics in this chapter include:

- *Opening an Unannotated XMI File* on page 51
- *Annotation Basics and Viewing an Unannotated File* on page 55
- *Next Steps - RoundTrip or Semantic Connector* on page 58

Opening an Unannotated XMI File

The first option listed on the SIW Welcome screen is the **Review Unannotated XMI File (Model Owner)** option. As indicated, this step is performed by the owner of the UML model.

Reviewing an unannotated XMI file is the process of reviewing an XMI file that has been exported from your UML modeling program (either EA or ArgoUML).

The purpose of viewing your unannotated XMI file through the SIW is so that the program can flag any problems it finds in the model. This allows you to easily review and fix those issues before continuing, and to confirm that the XMI file you may be using for the caCORE SDK is valid for use with the SIW. Typically the errors reported in this mode are due to missing tagged values (e.g., UML description tags).

Input To Step: The original UML model in XMI format, exported from either EA (saved with an .xmi file extension) or ArgoUML (saved with a .uml file extension). The filename can be anything you like, as long as you can browse to and select it.

Example: `myModel.xmi` for an EA export; `myModel.uml` for an ArgoUML export.

Output From Step: If you make changes to the file during this step, you can save the file using the **File > Save** or **File > Save As** command. As is typical, using **File > Save** will overwrite the existing file. However, you may want to use **File > Save As** in order to identify that this file differs from the actual UML model output.

While you can save the file using any name you like, we recommend using a naming convention consistent with the naming convention already established for prior releases (if applicable), or a naming convention that will carry to future versions.

To Review an Unannotated XMI file:

1. Be sure you have a valid UML model export file to review. For information on exporting your UML model, see [Generating the XMI File for the caCORE Process](#) on page 13.
2. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.
3. From the SIW Welcome screen, select option **1. Review Unannotated XMI File (Model Owner)** and click **Next**. The file selection screen appears.

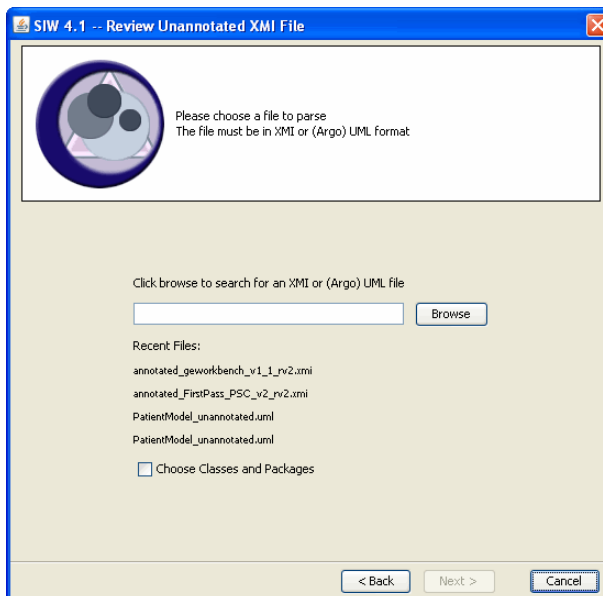


Figure 4.1 Select a file for SIW to parse and display for review

If you have used this mode of the SIW before, the last five files used for this option appear in a Recent Files list located below the filename text box.

4. Identify the file you want to review. You have the following options:
 - If the file you want to review appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename.
 - **Type** the full path and filename into the text box.
 - Click **Browse** to browse for and select the file exported from EA or ArgoUML.

Note: The Browse function defaults to selecting (showing) only files with an .xml file extension. If you are using a .uml file, select **UML Files** from the **Files of Type** drop-down list at the bottom of the Open dialog box.

5. If you want to review only portions of the file, enable the **Choose Classes and Packages** checkbox. Otherwise the SIW will parse the entire file for review.
6. Click **Next**. If you enabled the **Choose Classes and Packages** checkbox, the package selection screen appears, listing all of the packages in the file.

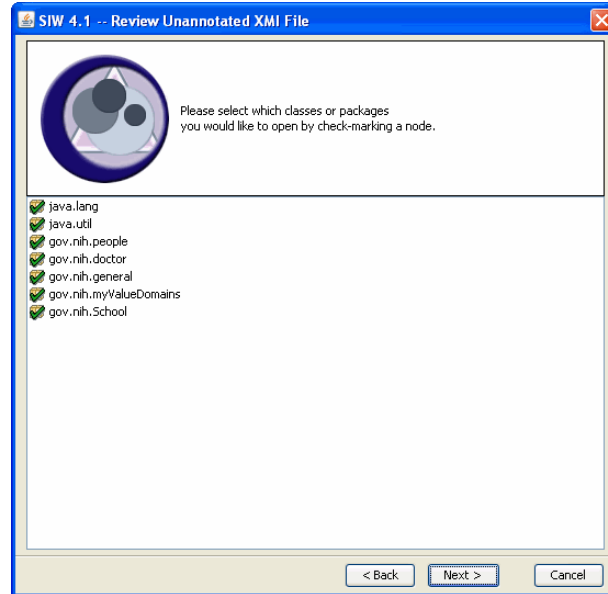


Figure 4.2 Package selection screen - lists the packages resident in the file

7. All packages listed are checked by default. Click a **checkmark** to open a list of the items contained within the package, and then click to clear the checkmarks next to any items you do not want the SIW to parse for review. Then click **Next**.

The SIW displays the file selection screen with a progress bar at the bottom showing the current action and overall progress. How long this process will take is

determined by the size of the XMI file or packages you are using as well as the number of errors or warnings the SIW must generate.

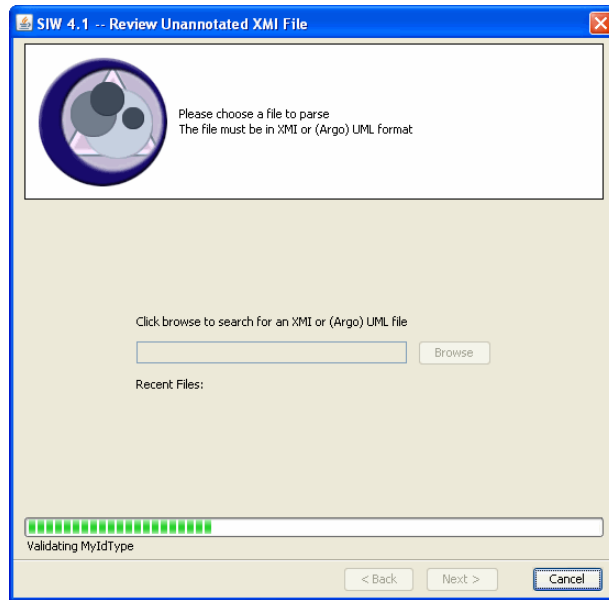


Figure 4.3 Screen and progress bar showing SIW processing of XMI file

When the SIW is finished, you may receive a Fatal Error in Model dialog box. This lets you know that there are errors in the file that must be fixed before further semantic integration can take place.

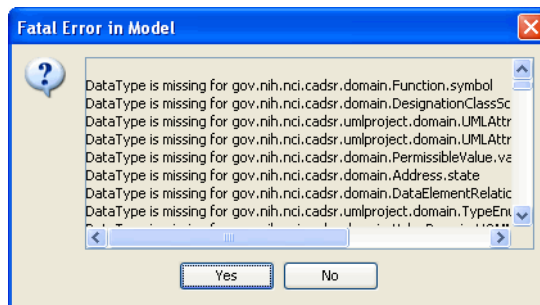


Figure 4.4 Fatal Error dialog box identifies the errors found in the XMI file

If the Fatal Error in Model dialog box appears, you have two options:

- **Click No** - This exits the SIW completely. If you know what the problems are, you can fix the errors in your model, re-export and attempt to review the unannotated file again.
- **Click Yes** - This opens the SIW viewer, with the errors and warnings appearing in the Errors tab.

If your file contains no parsing errors, the SIW viewer appears automatically when parsing is finished.

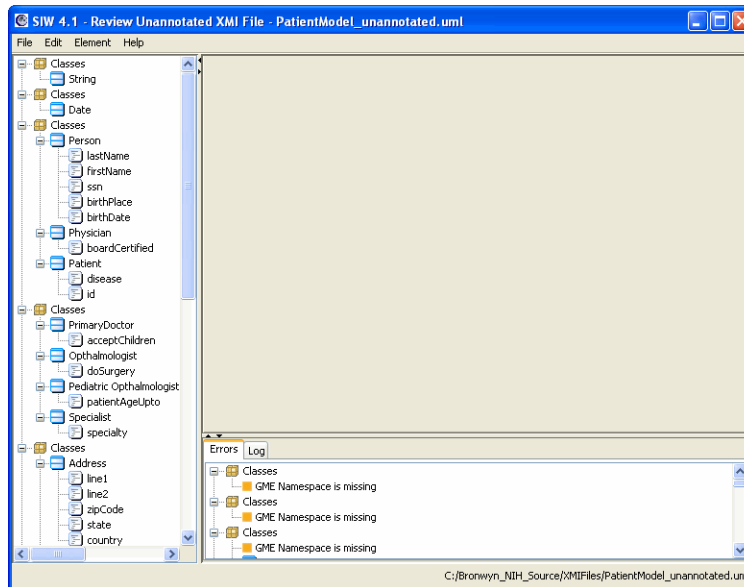


Figure 4.5 SIW Viewer - showing an unannotated XMI file with errors

Use the SIW viewer to review the errors encountered in the exported file, and to determine what changes need to be made in your UML model. Once you have made those changes, you can re-export the XMI file and open it again in the SIW to see if further changes need to be made before continuing.

See [Using the SIW Viewer](#) on page 35 for more information on how to navigate the SIW viewer to view the various components of your XMI file.

Annotation Basics and Viewing an Unannotated File

The point of viewing an unannotated file in the SIW is to see what items need to be changed in the model (and a new file exported) before you can continue using SIW to annotate the file. This mode of the SIW allows you to be sure that the information resident in the file is clear and accurate and that you have addressed the errors reported by the SIW. This may mean several iterations of file review along with changes to the UML model and re-export to review the new file.

Once the unannotated file is complete, it can move forward through the semantic integration process, providing annotations for the model elements.

The purpose of annotating an XMI file is to map the elements in the file to concept information from EVS. This mapping can only be done if the names of the elements are accurate and the element descriptions provided in the model are sufficiently clear as to the meaning, purpose, or intent of the element.

The following sections provide basic information regarding the relationship between the elements in your UML model and the concept information to which those items will ultimately be mapped.

UML Element Descriptions and Annotations

Each element in the model must have a UML element description. These descriptions are provided through the use of tagged values in the model. The purpose of these descriptions is to provide the EVS concept curators with some understanding as to the model owner's intended meaning of an attribute or class. This allows the curators to more accurately map each element to the appropriate EVS concept(s).

Historically these tagged values were defined as “documentation” tags for classes and “description” tags for attributes. However a new tag has been defined for the SIW called CADSR_Description and can be used as a tagged value for either classes or attributes.

If the UML model contains CADSR_Description tags, that information appears in the SIW as the UML element description. This allows model owners to provide caDSR-specific definitions for each model, which are then pulled into the SIW (and ultimately the caDSR during registration). If the model does not contain these tags, the legacy documentation and description tags are still used.

If there are multiple description tags for an element, the SIW concatenates these tags to display a single UML definition in the SIW. For this concatenation to occur, the format you must use for the CADSR_Description tags is as follows:

- CADSR_Description
- CADSR_Description2
- CADSR_Description3

You may add up to eight CADSR_Description tags if necessary (EA has a limit of 255 characters per tag). The UML definition field can include up to 2000 characters.

These descriptions, along with the naming convention used for the elements in the model, provide the information necessary to properly annotate the XMI file.

For a file to be considered “annotated,” each UML element, classes as well as attributes, must be mapped to at least one EVS concept. A concept is made up of the fields described in [Table 4.1](#).

Concept Fields	Example
Code	C16612
Name	Gene
Definition	The physical and functional unit of ...
Definition Source	NCI-GLOSS

Table 4.1 UML element concepts

The process of mapping to EVS concepts is done in two ways:

- Automatically through the Semantic Connector, which uses the name of the element to determine what concepts are appropriate for mapping (applying camel case logic to recognize segments of an element name);

- Manually through the curation process, which is done by the EVS curation team using the element descriptions as a guide to the intended definition, intent, or meaning of each element, mapping the concepts appropriately.

Keep this information in mind as you review your unannotated XMI file. The more accurate and complete the information in the file is, the smoother the annotation process will be.

Association Annotations

Associations are the relationships between items in the UML model. The relationship information for an association can be viewed in the SIW, although it cannot be changed. However, each of the items involved in the relationship (Role, Source, and Target) can have concept information associated with them.

Like all concept annotations, associations elements can have a primary concept with qualifier concepts (as applicable). However, because these concepts are linked to association elements, the tagged values created for each item differ from those created for classes and attributes.

You should consider annotating your associations in order to provide semantic clarity for the associations. In addition, the SIW will return a warning if there are more than two associations between two given classes.

For reference, the following tagged values are used for annotating associations:

1. To annotate the role:

- AssociationRoleConceptCode
- AssociationRoleConceptPreferredName
- AssociationRoleConceptDefinition[_n]
- AssociationRoleConceptDefinitionSource
- AssociationRoleQualifierConceptCodeN
- AssociationRoleQualifierConceptPreferredNameN
- AssociationRoleQualifierConceptDefinitionN[_n]
- AssociationRoleQualifierConceptDefinitionSourceN

2. To annotate the source:

- AssociationSourceConceptCode
- AssociationSourceConceptPreferredName
- AssociationSourceConceptDefinition[_n]
- AssociationSourceConceptDefinitionSource
- AssociationSourceQualifierConceptCodeN
- AssociationSourceQualifierConceptPreferredNameN
- AssociationSourceQualifierConceptDefinitionN[_n]

- AssociationSourceQualifierConceptDefinitionSourceN

3. To annotate the target:

- AssociationTargetConceptCode
- AssociationTargetConceptPreferredName
- AssociationTargetConceptDefinition[_n]
- AssociationTargetConceptDefinitionSource
- AssociationTargetQualifierConceptCodeN
- AssociationTargetQualifierConceptPreferredNameN
- AssociationTargetQualifierConceptDefinitionN[_n]
- AssociationTargetQualifierConceptDefinitionSourceN

Next Steps - RoundTrip or Semantic Connector

Once you have reviewed your unannotated XMI file and addressed any issues in the model that you can at this point, it is time to move onto the automated steps of the SIW. Which you choose depends on how your model was created.

If you created your model based on another registered model or on a previously registered version of your system, the next step of the SIW for you is the RoundTrip XMI File step. For more information, see [Chapter 5, Using XMI RoundTrip](#), on page 59.

If this is a completely new model, the next step of the SIW for you is the Semantic Connector. For more information, see [Chapter 6, Using the Semantic Connector](#), on page 67.

CHAPTER 5

USING XMI ROUNDTrip

This chapter provides detailed information on the purpose of the XMI Roundtrip of the SIW and procedures for processing your XMI file through the Roundtrip process.

The XMI Roundtrip step is designed for use by model owners who have created a new model based on an existing registered UML model or who have a new version of a previously registered model of their system.

Topics in this chapter include:

- [What XMI Roundtrip Mode Does](#) on page 59
- [Running XMI Roundtrip](#) on page 60
- [Next Steps - Review Annotated File and Semantic Connector](#) on page 65

Note: If you have or plan to add GME tagged values in your model, review [Chapter 9, Using GME Annotations](#), on page 109 before using the procedures in this chapter.

What XMI Roundtrip Mode Does

The XMI Roundtrip option is performed by the model owner in an attempt to automatically annotate his or her model with existing data from caDSR. Using this step, a model owner can automatically annotate a new version of a model based on existing mappings located in a previously loaded model. The automated matching is based on the new model having used exactly the same names for the classes and attributes as the previous model.

The XMI Roundtrip is an optional step in the semantic integration process but can save time in the following two situations:

1. A prior version of this model was previously loaded and parts of the models have not changed, specifically when no changes have been made to the names of the classes and attributes.

2. The model shares class and attribute names with another, previously loaded model.

In addition, the Roundtrip step can add GME namespace tags to components in your model that have been reused from other models. In this case, the GME annotations added to your model will point to the XSD for the reused components. Use the checkbox in the Roundtrip wizard to determine whether you want to include or exclude GME namespace annotations as part of the Roundtrip process. For more information on when it is appropriate to use the Roundtrip task to write GME namespace tags to your model, see [Chapter 9, Using GME Annotations](#), on page 109.

Where possible, the XMI Roundtrip task maps new or updated UML attributes to existing caDSR CDEs rather than to EVS concepts.

Input To Step: The original UML model in XMI format, exported from either EA (saved with an .xmi file extension) or ArgoUML (saved with a .uml file extension) and reviewed through the Review Unannotated XMI File step.

Example: myModel.xmi for an EA export; myModel.uml for an ArgoUML export.

Output From Step: A partially annotated XMI file with the term “Roundtrip” appended to the beginning of the filename (roundtrip_\${filename}.xmi. or roundtrip_\${filename}.uml).

The file output from this step differs from a file output from the Semantic Connector in that the Roundtrip XMI file is annotated with caDSR public IDs rather than EVS concepts.

Running XMI Roundtrip

This section provides the instructions necessary for launching and completing the XMI Roundtrip step of the SIW.

To Perform XMI Roundtrip:

1. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.

- From the SIW Welcome screen, select option **2. Perform XMI Roundtrip (Model Owner)** and click **Next**. The Project Search screen appears.

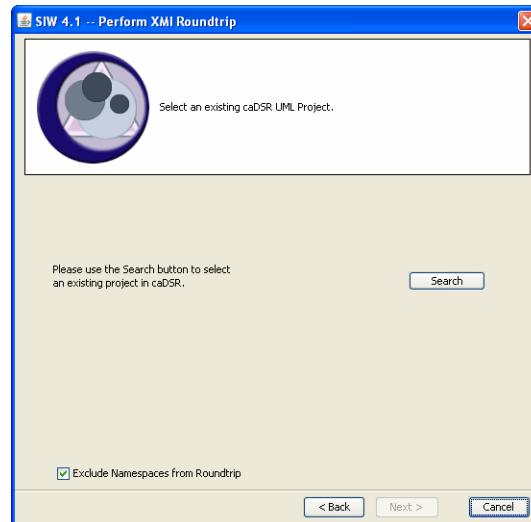


Figure 5.1 Search for an existing Project to use for Roundtrip

- In the project search screen, use the **Exclude Namespaces from Roundtrip** checkbox to determine whether you want the Roundtrip step to generate GME namespace tags for the reused components in your model. This box is checked by default.
- Click **Search**. This opens a Search for Classification Schemes window (Figure 5.2) that allows you to search caDSR for an existing project and version to use for the Roundtrip processing.

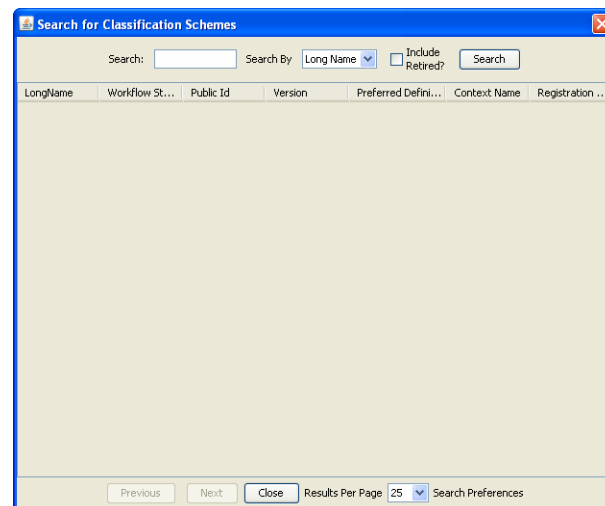


Figure 5.2 Search for Classification Schemes (project) to use for Roundtrip processing

5. Enter a classification scheme name in the Search text box and click **Search**. You may use the asterisk (*) as a wildcard along with a portion of a name, to return a list of projects to choose from.

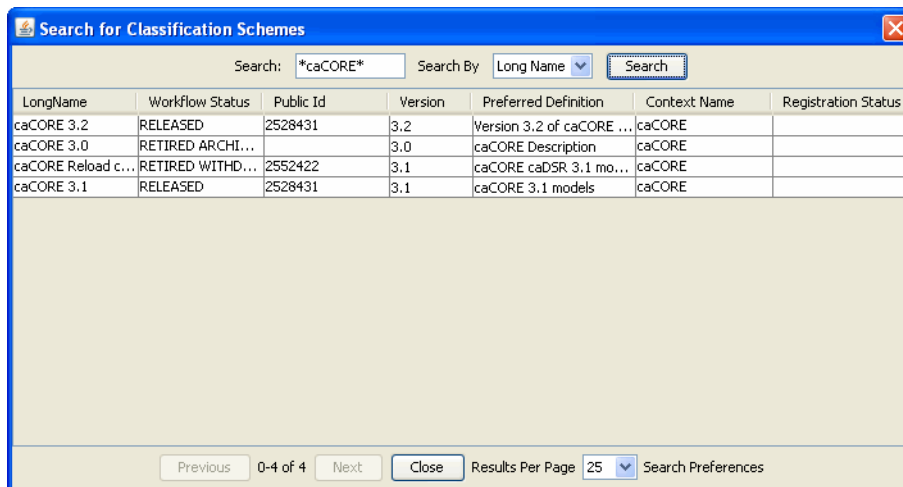


Figure 5.3 Search for Classification Schemes - result set

6. From the result set, find the project *and* version of the project you want to use for processing your new file through Roundtrip, and double-click on that entry. The select project window returns, showing the details of the selected project.

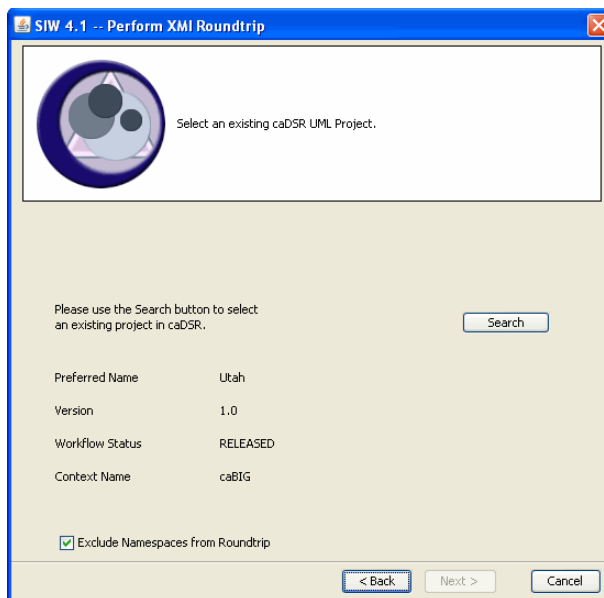


Figure 5.4 Project selection screen showing selected Roundtrip project details

7. Click **Next**.
8. In the Select file to parse window, identify the file you want to process. You have the following options:
 - If the file you want to review appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename;

- **Type** the full path and filename into the text box;
- Click **Browse** to browse for and select the file exported from EA or ArgoUML.

Note: The Browse function defaults to selecting (showing) only files with an .xmi file extension. If you are using a .uml file, select **UML Files** from the **Files of Type** drop-down list at the bottom of the Open dialog box.

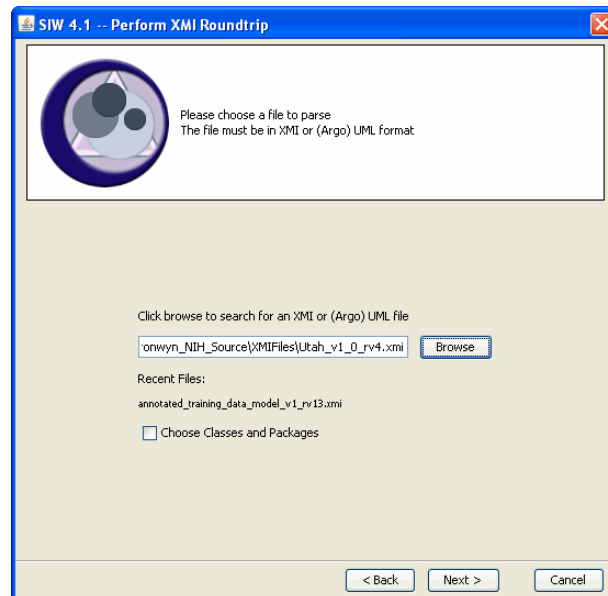


Figure 5.5 Select the file to process through XMI Roundtrip

9. If you want to process only portions of the file, enable the **Choose Classes and Packages** checkbox. Otherwise the SIW will submit the entire file through the Roundtrip process.

10. Click **Next**. If you enabled the **Choose Classes and Packages** checkbox, the package selection screen appears, listing all of the packages in the file. All packages are checked by default.

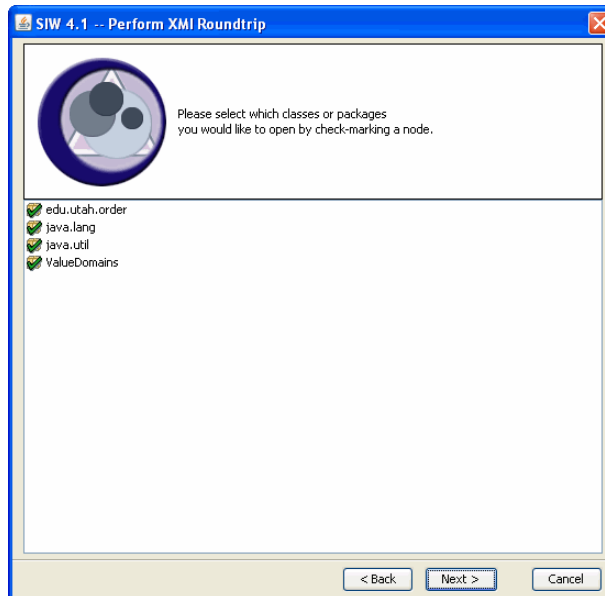


Figure 5.6 Package selection screen - lists the packages resident in the file

11. Click a **checkmark** to open a list of all items contained within the package, and then click to remove the checkmarks next to any items you do not want submitted through Roundtrip. Then Click **Next**.

The Roundtrip step begins to process your XML file against the project you selected.

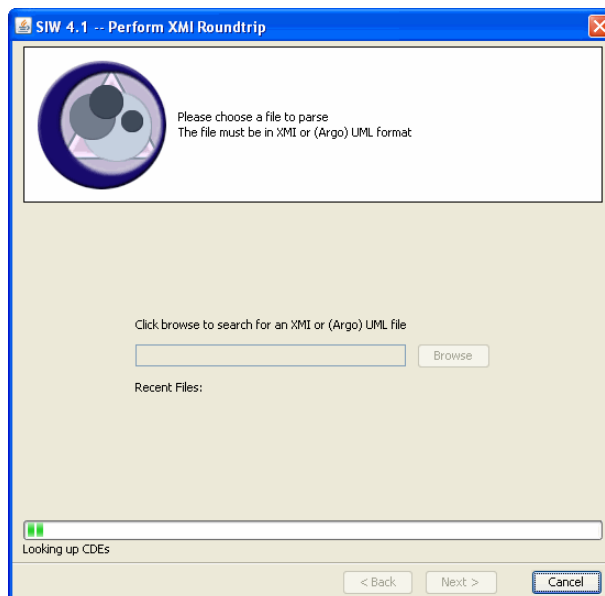


Figure 5.7 SIW performs the Roundtrip process on the selected file against the selected project

For each UML attribute in the XMI file being processed, the SIW attempts to find a CDE with an Alternate Name in the selected Project/Version that is similar to the fully

qualified attribute name in the model. If a match is found, the attribute in the XMI file is annotated with the matching CDE public ID/Version from caDSR. Note that mapping an attribute to a CDE also maps the class to a caDSR OC and the datatype for the attribute to a caDSR value domain.

When the Roundtrip process is complete, the SIW saves a new version of the XMI file into the same directory as the source file, appended with the term “Roundtrip” on the front of the filename. For example, if the processed file's path/name was `c:\myXMIfiles\Version2\myModel.xmi` the newly saved file will be `c:\myXMIfiles\Version2\Roundtrip_myModel.xmi`.

Next Steps - Review Annotated File and Semantic Connector

After performing the Roundtrip step, the model owner should review the automated mapping using the **Review Annotated XMI File** option. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91.

If there are UML elements in the file that are not mapped, the model owner will want to process the XMI file through the Semantic Connector to map those elements to EVS concepts. For more information, see [Chapter 6, Using the Semantic Connector](#), on page 67.

CHAPTER 6

USING THE SEMANTIC CONNECTOR

This chapter provides detailed information on the purpose of the Semantic Connector step of the SIW along with the procedures necessary for processing your XMI file through the Semantic Connector.

If you are new to the caCORE process, this is the automated step of semantic annotation you are looking for. The Semantic Connector automatically maps the elements in your model to EVS concepts, based on the element names. This step is designed to be used either with a newly exported XMI file or with one that has been processed through Roundtrip.

These automated mappings will need to be edited later, however the Semantic Connector saves significant time by automating the search/map part of the process.

Topics in this chapter include:

- *What the Semantic Connector Does* on page 67
- *Running the Semantic Connector* on page 69
- *Next Steps - Review Annotated File and Curate XMI File* on page 71

What the Semantic Connector Does

The Semantic Connector option of the SIW performs an EVS search against each element in the XMI file (or selected packages), and attaches one or more EVS concepts to each element. The search is performed using the element names. If the elements in the file are named properly using the “camel case” convention, the SIW can separate the parts of each element based on the capitalization in the element name, and perform an EVS search on each part. The Semantic Connector then maps the parts of each element to EVS concepts as appropriate.

For example, the XMI file contains an attribute called “initialDiagnosisDate.” Because the element name uses camel case, the Semantic Connector recognizes three parts to

the element: initial, diagnosis and date. Each of those terms is then searched against the NCI Thesaurus, and three matching EVS concepts are returned.

Because “date” appears last in the series, the Semantic Connector identifies that as being the main idea of the element, and therefore maps the Primary Concept as “date.” The Qualifier Concept #1 is mapped as “diagnosis” and Qualifier Concept #2 is mapped as “initial.” The element is now annotated with a primary and two qualifier concepts.

If only, however, it were that simple. Since the Semantic Connector is an automated process, it actually maps *all* of the matches it finds for terms to the elements. Meaning that the element called “Id” in your model might have several concepts mapped to it, including “identifier”, “Idaho”, and “Indonesia” because all of those match the term “id”. So that part of the curation process after the automated concept mapping includes removing the extraneous concept mapping, and replacing mapped concepts with perhaps better choices.

When the Semantic Connector step is finished, the SIW automatically creates the Semantic Connector Report and saves it as a new file to the same directory where the source file resides. The model owner should review this file before submitting it to the EVS curators for the Curate XMI File process.

Although the Semantic Connector will add concepts to anything not marked as “reviewed” (even those items already mapped to a CDE), those concepts are ignored during validation and loading, in favor of the mapped CDEs.

Input To Step: The original UML model in XMI format, exported from either EA (saved with an .xmi file extension) or ArgoUML (saved with a .uml file extension) and reviewed through the Review Unannotated XMI File step. Or the input could be an XMI file that has been processed through the XMI Roundtrip step.

Example: `myModel.xmi` for an EA export; `myModel.uml` for an ArgoUML export; or `roundtrip_myModel.xmi` or `roundtrip_myModel.uml` for a Roundtrip processed file.

Output From Step: The Semantic Connector Report, saved as an XMI file with the term “FirstPass” appended to the front of the file name (e.g., `FirstPass_myModel.xmi` or `FirstPass_Roundtrip_myModel.xmi`).

After performing this step, the model owner can review the automated mapping using the **Review Annotated XMI File** option. This is strongly recommended so that you can see how the Semantic Connector mapping works, even if you are not comfortable with changing any of the initial mapping. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91.

The file output from this step is ultimately sent to the EVS curation team as the input for the Curate XMI File step of the SIW.

Running the Semantic Connector

This section provides the procedures necessary to process your XMI file through the Semantic Connector step of the SIW.

To run the Semantic Connector:

1. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.
2. From the SIW Welcome screen, select option **3. Run Semantic Connector (Model Owner)** and click **Next**.
3. In the Select file to parse window, identify the file you want to process. You have the following options:
 - If the file you want to review appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename.
 - **Type** the full path and filename into the text box.
 - Click **Browse** to browse for and select the file exported from EA or ArgoUML.

Note: The Browse function defaults to selecting (showing) only files with an .xmi file extension. If you are using a .uml file, select **UML Files** from the **Files of Type** drop-down list at the bottom of the Open dialog box.

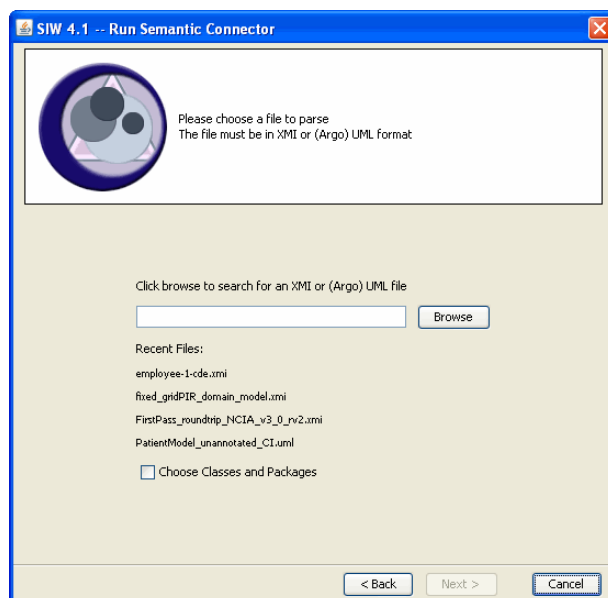


Figure 6.1 Select the file to process through the Semantic Connector

4. If you want to process only portions of the file, enable the **Choose Classes and Packages** checkbox. Otherwise the SIW will submit the entire file through the Semantic Connector.

- Click **Next**. If you enabled the **Choose Classes and Packages** checkbox, the package selection screen appears, listing all of the packages in the file. All packages are checked by default.

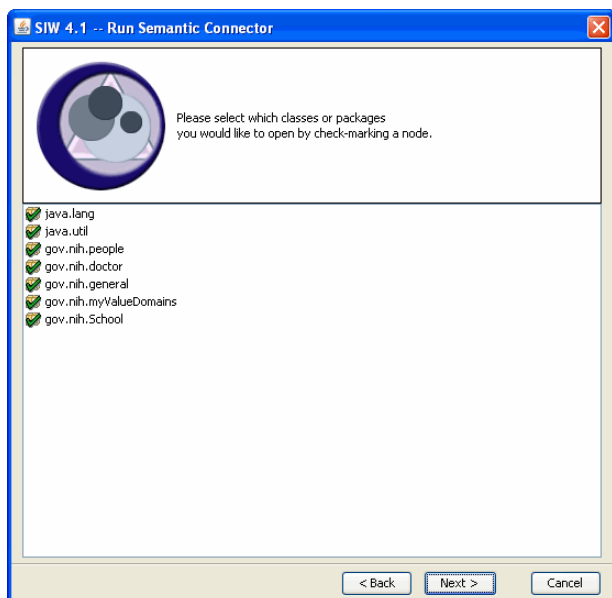


Figure 6.2 Package selection screen - lists the packages resident in the file

- Click a **checkmark** to open a list of all items contained within the package, and then click to remove the checkmarks next to any items you do not want to process through the Semantic Connector. Then click **Next**.

The Semantic Connector begins to process the XMI file against the NCI Thesaurus.

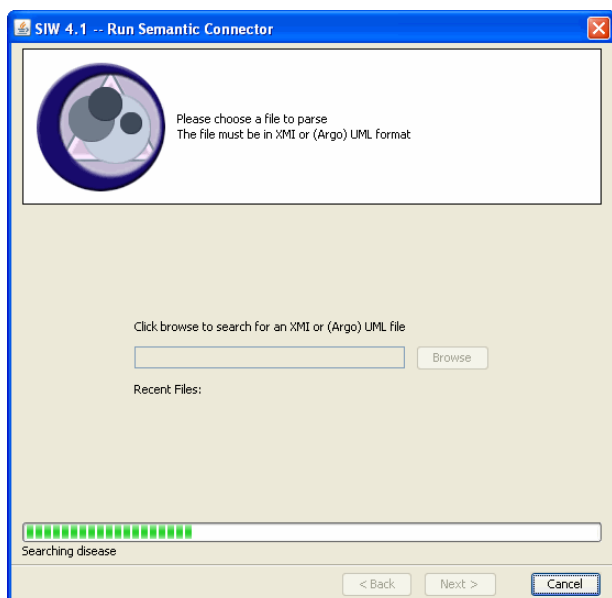


Figure 6.3 SIW performs the Roundtrip process on the selected file

If the process is successful, the Semantic Connector maps the elements to EVS concepts and inserts them into the XMI file. When the Semantic Connector is done, the SIW saves a new version of the XMI file, which is referred to as the Semantic

Connector Report. A final screen appears confirming completion of the process and identifying the location of the newly generated file. Note the location and click **Finish**.

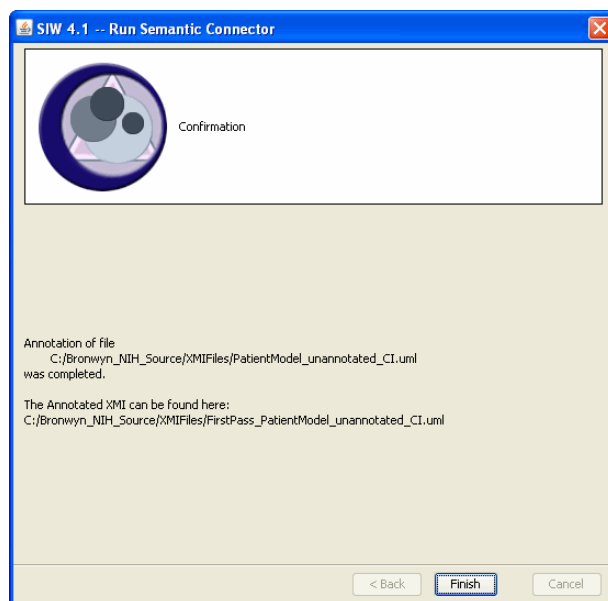


Figure 6.4 Semantic Connector process complete - displays location of new XMI file

This new file is saved into the same directory as the source file and is appended with the term “FirstPass” on the front of the filename. For example, if the processed file’s path/name was `c:\myXMIfiles\myModel.xmi` the newly saved file will be `c:\myXMIfiles\FirstPass_myModel.xmi`. If the example file had been submitted through the Roundtrip process first, the newly saved file would be named `c:\myXMIfiles\FirstPass_Roundtrip_myModel.xmi`. If your source was exported from ArgoUML, the extension on the filename would continue to be `.uml`.

Next Steps - Review Annotated File and Curate XMI File

After performing this step, the model owner is strongly encouraged to review the Semantic Connector Report (new XMI file) using the **Review Annotated XMI File** option. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91. If you are the kind of model owner who wants to take a first pass at editing the concept mapping, use the Review Annotated XMI File step to perform this function.

After reviewing or making concept mapping changes, the file should be submitted to the EVS curation team to perform the Curate XMI File step of the SIW process. The EVS curation team reviews the file, removes any remaining extraneous mapped concepts, inserts new concepts, and reorders the mapped concepts to match the UML class and attribute entities and the intent identified for the element by the owner (through the element description tags).

Note: If the Semantic Connector process was not successful, an error message may appear, or the process may simply not complete. The most likely cause for this is that the XMI file is not in the expected format. This issue should be corrected before re-running the Semantic Connector and may require re-export of the XMI file from EA.

CHAPTER 7

CURATING AN XMI FILE

This chapter provides detailed information on how the EVS Curation team uses the SIW to curate your XMI file. While the information contained in this chapter is directed at the CBIIT staff who uses this option of the SIW, the procedures contained here also apply in large part to the procedures available to model owners through the Review Annotated XMI File step.

Topics in this chapter include:

- *What is Curating an XMI File?* on page 73
- *Opening an XMI File for Curation* on page 74
- *Adding EVS Concepts to an Element in Curate Mode* on page 76
- *Searching EVS for Concept Information* on page 78
- *Editing Concept Information in Curate Mode* on page 81
- *Applying Changes to All Similar Nodes* on page 86
- *Validating XMI File Concept Information Against EVS* on page 87
- *Verifying the Curated XMI File* on page 89

What is Curating an XMI File?

The **Curate XMI File** step is performed by the EVS concept curation team at CBIIT. During this step, the EVS team adds and removes concepts from the XMI file, and indicates recommended semantic mappings for the UML model. The EVS team will use existing EVS concepts where possible, or create new EVS concepts to reflect new items being introduced by the model.

EVS curators have the authority to change any concept field and to use concepts that are not in the NCI Thesaurus. Model owners, on the other hand, should only change concept information by picking an EVS concept from the Thesaurus or entering one

supplied by EVS. New EVS concepts cannot be verified using the **Search EVS** feature, so new concepts should only be used by EVS curators during the Curate XMI File step. However if the EVS curation team supplies the model owner with a new EVS concept to use, they can be added during the Review Annotated XMI File step.

The end result of the curation process is an Annotated XMI file that is returned to the model owner who can then review the annotated file and either accept or change the mappings performed by the curators. Frequently, if the model owner makes changes to the file, they will return it to the EVS curation team for re-review and curation.

Input: The Semantic Connector Report, which is the file that the SIW saved after completing the Run Semantic Connector step. The file should be appended with “FirstPass” at the front of the filename.

Example: `FirstPass_myModel.xml` or `FirstPass_Roundtrip_myModel.xml` (if the file was processed through the Roundtrip step before the Semantic Connector).

Output: An curated XMI file. The file should be appended with “annotated” at the front of the filename. For example: `Annotated_FirstPass_myModel.xml` or `Annotated_FirstPass_Roundtrip_myModel.xml`.

The curated XMI file is the input to the Review Annotated Model step. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91.

Note: The procedural information in this section is directed specifically at EVS curators, though model owners are *strongly* encouraged to review this section in order to understand the process of curation. Be advised however, where the text in this section refers to “you” or “the user” performing a task, it is referring to the EVS curator.

Opening an XMI File for Curation

This section provides procedures for opening an XMI file for Curation. Subsequent sections of this chapter provide details on how to curate the elements in the file itself.

To Curate an XMI File:

1. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.
2. From the SIW Welcome screen, select option **4. Curate XMI File (Vocabulary Reviewer)** and click **Next**.
3. In the Select file to parse window, identify the file you want to process. You have the following options:
 - If the file you want to review appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename.
 - **Type** the full path and filename into the text box.

- ° Click **Browse** to browse for and select the file exported from EA or ArgoUML.

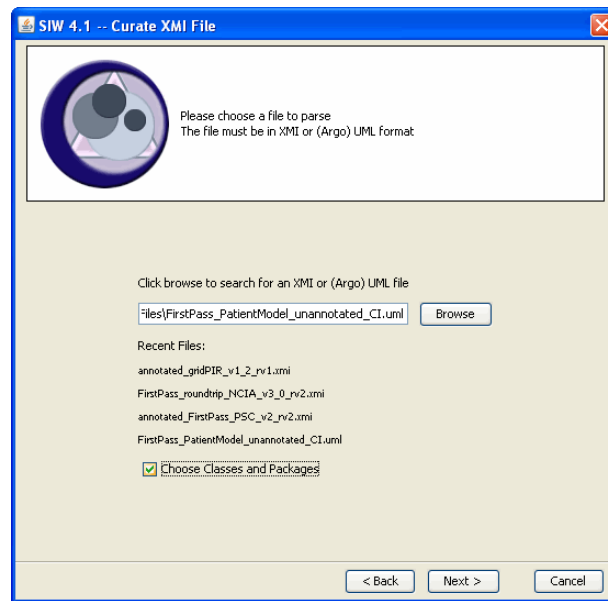


Figure 7.1 Select the XMI file to curate

4. If you want to process only portions of the file, enable the **Choose Classes and Packages** checkbox. Otherwise the SIW will open the entire file for curation.
5. Click **Next**. If you enabled the **Choose Classes and Packages** checkbox, the package selection screen appears, listing all of the packages in the file. All packages are checked by default.

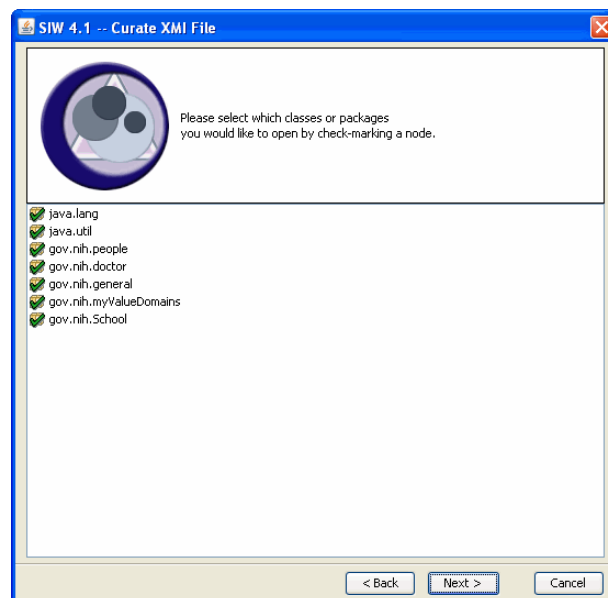


Figure 7.2 Package selection screen - lists the packages resident in the file

6. Click a **checkmark** to open a list of all items contained within the package, and then click to remove the checkmarks next to any items you do not want to open for curation. Then click **Next**.

When the SIW is finished parsing the file, the SIW viewer appears.

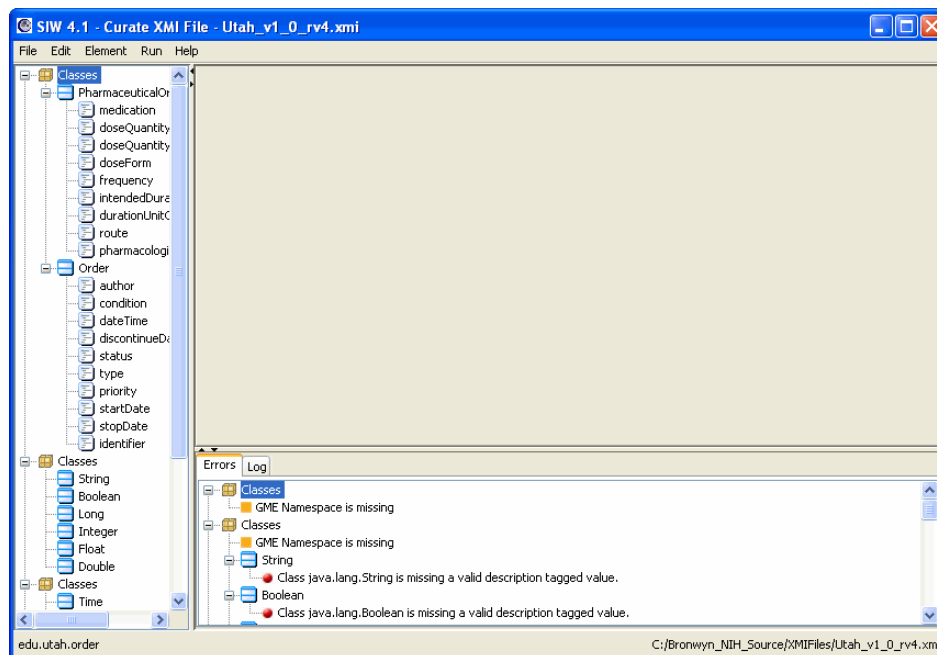


Figure 7.3 SIW Viewer - showing an XMI file ready for curation

Any element not already mapped to a caDSR CDE or OC must be mapped to one or more EVS concepts. The Semantic Connector process did some of the concept mapping automatically, however it likely mapped multiple concepts for more common elements (such as “id”) or did not find any matching concepts for some elements. Furthermore, additional concepts may need to be added to an element, some mapped concepts may need to be removed, or the semantic order of the concepts may need to change.

The SIW viewer allows you to add, remove, edit and re-order the EVS concepts for the elements in the XMI file. The sections that follow provide information on how to perform these tasks.

If necessary, see [Using the SIW Viewer](#) on page 35 for a basic overview on navigating the SIW viewer and on using the **Previous**, **Next**, **Add**, **Apply** and **Search EVS** buttons.

Adding EVS Concepts to an Element in Curate Mode

If an element does not have an EVS concept mapped to it, you must add one. If an element has an EVS concept mapped to it, you may add one or more Qualifier concepts to the existing concept(s).

The first concept added to an element is called the “Primary” concept. Subsequent mapped concepts are called “Qualifier” concepts. Concepts are added and then listed in semantic order, meaning that the Primary concept is always added first and must reflect the basic point of the element (typically the last item in the element name).

For example, an element is named “initialDiagnosisDate.” The item itself reflects a date, while the rest of the information reveals details about the date. In this case, the Primary concept for the element should be mapped to an EVS concept named “date.”

If the element already has a Primary concept mapped, it may require further mapping of Qualifier concepts. In the example above, once the Primary concept is mapped, you would add a Qualifier concept mapped to an EVS concept named “diagnosis” and then add a second Qualifier concept mapped to an EVS concept named “initial.”

The procedures for adding concepts to an element are the same, regardless of whether you are adding Primary or a Qualifier concepts. Furthermore, adding concepts only applies to elements that are not already mapped to caDSR CDEs or OCs.

To add a concept:

1. In the SIW viewer, select an element from the navigation tree.
2. In the detail view, click **Add**. A set of empty concept fields appears, along with a **Search EVS** button.

Note: If you have changed your viewer Preferences to Display Primary Concept First, the new concept fields will appear below any existing mapped concepts, meaning you may need to scroll down in the detail view to see the new fields. See [Setting Viewer Preferences](#) on page 44 for more information on setting your viewer preferences.

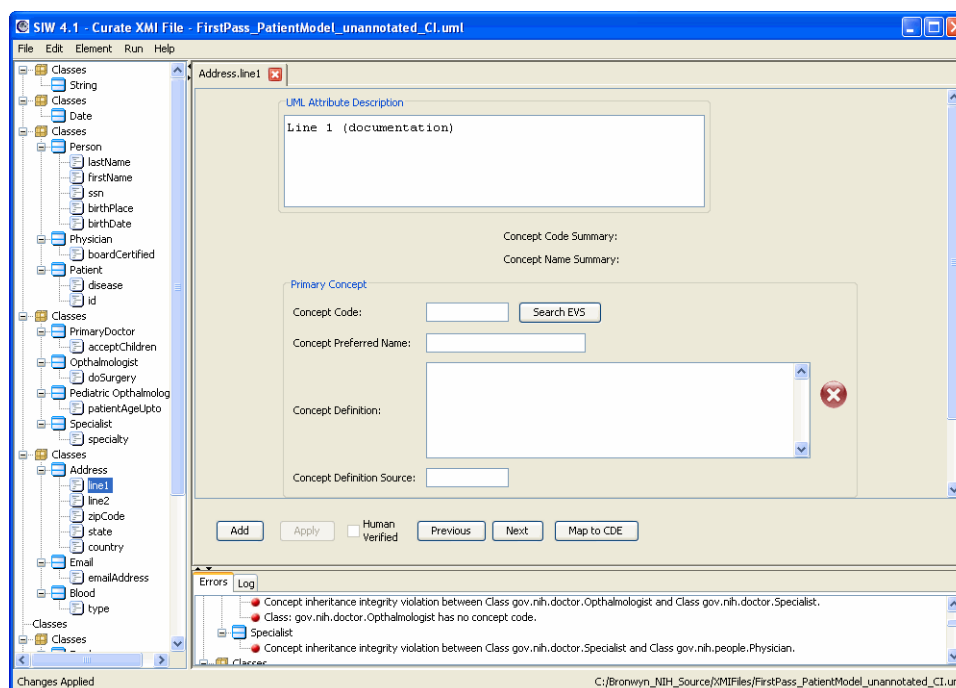


Figure 7.4 SIW Viewer in Curate XMI mode, showing new blank concept fields.

3. Enter the appropriate information into all four text fields. You have the following options:
 - Click **Search EVS** to perform a search for the appropriate concept to map. **Double-click on a search result item** to populate the concept fields with

the needed information. For more details on using the EVS search to populate the concept fields, see [Searching EVS for Concept Information](#), below.

- Manually type the information into the text boxes,
- 4. When the fields are complete, click **Apply**. This applies the new concept to the the element. (You will still have to save the file after all of your changes are made. See [Saving Changes to a File](#) on page 48.
- 5. If necessary, after applying your changes, you can click **Add** again and complete the above steps to add another concept to this element.

Searching EVS for Concept Information

Whether adding or editing concept information, the detail view of the SIW viewer provides a **Search EVS** button that allows you to search the NCI Thesaurus for existing concepts and if a match is found, to automatically map the concept to the element.

The **Search EVS** button appears next to each mapped concept and next to the empty concept information fields when you select to Add a new concept.

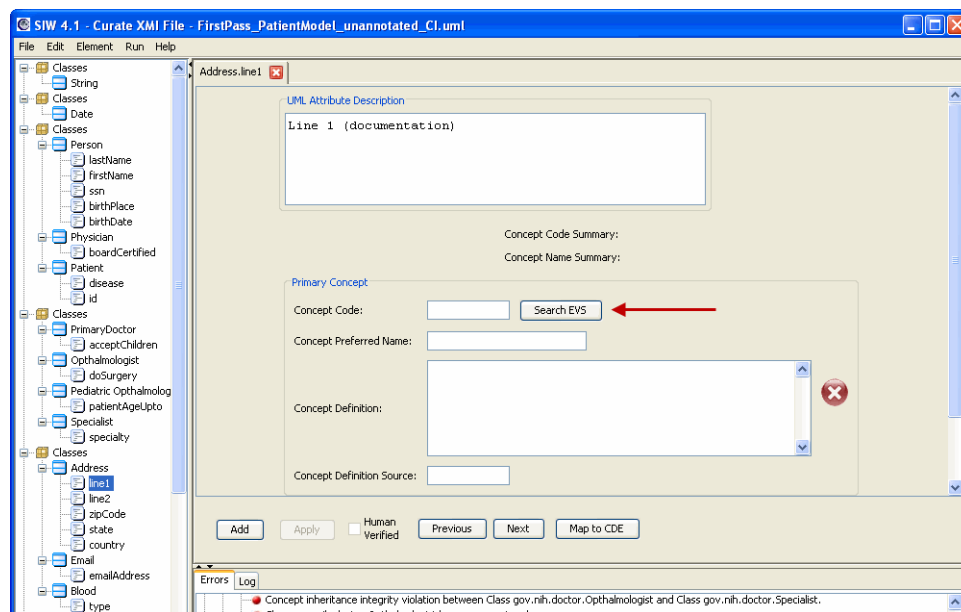


Figure 7.5 The Search EVS button appears next to concept information fields

By default, if the Concept Name field is populated when you click **Search EVS**, the EVS search automatically performs a synonym search of the NCI Thesaurus for the concept name listed. If you have changed your viewer Preferences to not automatically perform an EVS search, then you will need to manually enter search criteria. See [Setting Viewer Preferences](#) on page 44 for more information.

To search EVS/NCI Thesaurus for concept information:

1. In the SIW viewer, click **Search EVS**, located to the right of the Concept Code field. The Search Thesaurus dialog box appears in one of two states:

- If the Concept Name field was populated when you clicked **Search EVS** (and your Viewer Preferences are set to automatically search EVS), the concept name appears in the Search text box, a search is automatically performed, and the results appear in the Search Thesaurus dialog box.

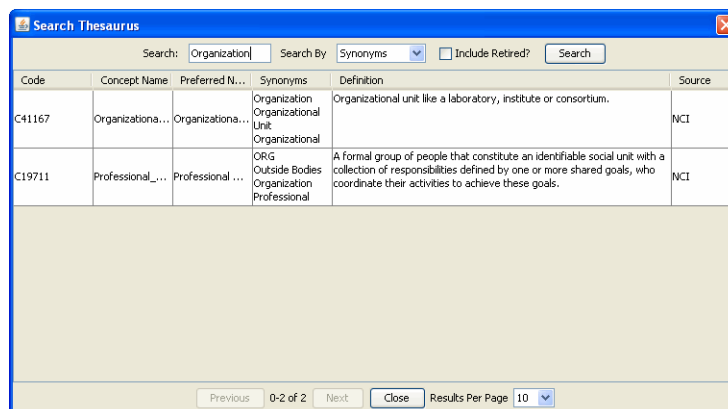


Figure 7.6 Search Thesaurus dialog box with automatic search results

- If the Concept Name field was blank when you clicked **Search EVS** (for example, if you are adding a new concept), the Search Thesaurus dialog box appears but is blank.

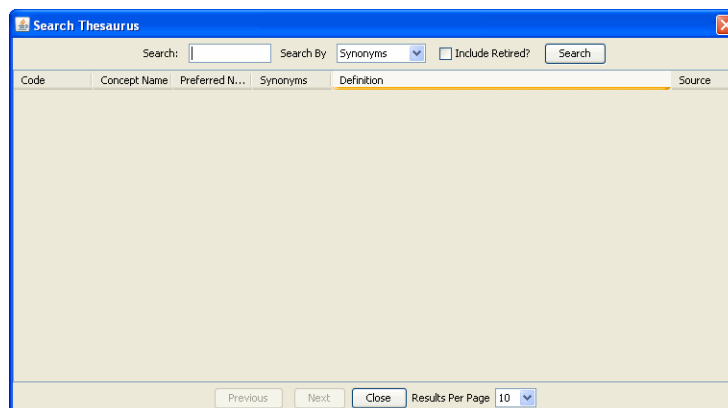


Figure 7.7 Blank Search Thesaurus dialog box - enter criteria for search

The Search Thesaurus dialog box contains the following items:

- **Search text box:** Use this field to enter your search criteria. You may use an asterisk (*) as a wildcard. The search is not case-specific.
- **Search By drop-down list:** Use this option to select whether you want to search on the Synonym field or Concept Code field in EVS.
- **Include Retired checkbox:** Use this to select whether to include Retired concepts in the search results. Retired concepts are excluded by default, as active concepts are preferred for mapping.
- **Search button:** Click to execute the search based on the criteria you entered.

- **Close button:** Click to close the search dialog box without mapping any concepts.
 - **Previous and Next buttons:** These buttons only become active if your search results cover more than one page. Use these to page through the result set. Between these buttons is an indicator that lets you know which results are currently displayed along with the total number of items returned.
 - **Results Per Page drop-down list:** Use this option to change how many results appear on each page.
2. If an automatic search was run and the concept you want to map appears in the results, **double-click** the item. The information for the new concept replaces the information that had been in the concept information fields in the SIW. Otherwise, continue with the steps below to search for concepts.
 3. Enter your search criteria into the **Search** text box. You have the following options:
 - Type in the term you are looking for;
 - Type in part of the term you are looking for along with an asterisk (*) as a wildcard character. This method will return more results but also typically take longer because the result set is more inclusive;
 - Type in the concept code you want to find. You may NOT use a wildcard to search for concept code. Concept code searches return exact matches only.
 4. In the **Search By** drop-down list, select the NCI Thesaurus field to search in: **Synonym** or **Concept Code**.

 Each concept in the NCI Thesaurus has at least one synonym associated with it, and many concepts may share a synonym. For example, a search for “ID” will return multiple results because there are several concepts that have ID listed as a synonym (e.g., Identifier, Idaho, Indonesia). Conversely a search for “Genes” will return “Gene” as its only result because “Genes” only exists as a synonym for “Gene” in the NCI Thesaurus.
 5. Check the **Include Retired?** check box if you want the results to include Retired items. Retired concepts are excluded by default.
 6. Click **Search**. The results appear below the search options, listed in alphabetical order by concept name. If there are multiple pages of results, the

Previous and **Next** buttons are enabled, allowing you to scroll through the pages.

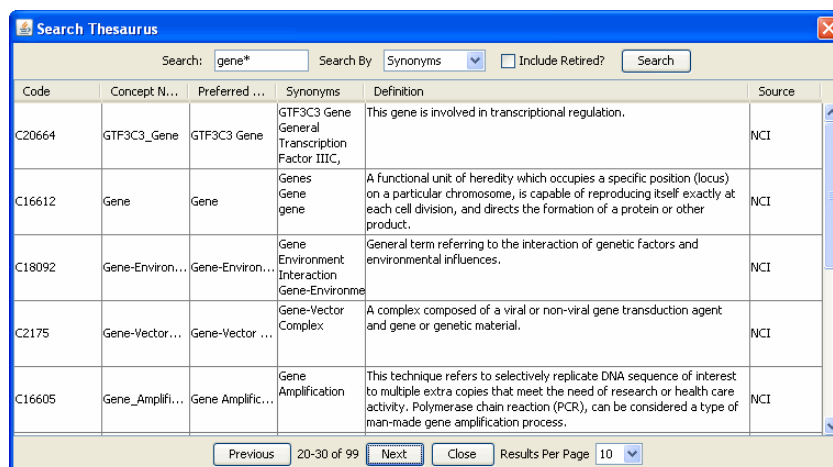


Figure 7.8 Search Thesaurus (EVS) dialog box with search results

If needed, you can resize the Search Thesaurus dialog box as well as resizing the columns within the dialog box, to better view the result set details. If you cannot sufficiently resize the box to see the information you need, hold your mouse cursor over each field to display the full text of the field in the mouse-over tool-tip that appears.

7. If the concept you want to map appears in the result set, **double-click** anywhere in the row for that concept. This closes the Search dialog box and enters the information for the selected concept into the concept information fields in the SIW viewer.
8. If the search returns no results (the Search dialog box remains empty) or the concept you want to map does not appear in the result set, alter your search criteria to be more inclusive by using the wildcard or by removing terms from the Search text box if multiple terms appear.

After making changes to the concept information, you can click **Apply** to apply those changes to that element, or select **Edit > Apply to All** to apply your changes to every instance of that element in the model. For more information, see [Applying Changes to All Similar Nodes](#) on page 86.

Note: The search result list is limited to 100 terms. If your search returns 100 terms, there may be more concepts available that match your criteria. Alter your search results to be more restrictive or specific to the concept information you are looking for.

Editing Concept Information in Curate Mode

Besides adding concepts to elements, EVS curators can also edit the existing concept information for an element. Editing concept information consists of replacing mapped concepts with different concepts, changing the semantic order of mapped concepts, editing the text in the concept information fields, and where necessary, removing concepts from an element's mapping.

Note: If you accidentally make changes to concept information that you did not intend to make or don't want to keep, click **Previous** or **Next** before clicking **Apply**. The SIW prompts you regarding whether or not to save the change. Click **No** and the element's concept information is returned to its original state.

Replacing Mapped Concepts in Curate Mode

Replacing mapped concepts uses the same steps as adding concepts except that the concept information fields are already populated, and if you use the **Search EVS** button to search the NCI Thesaurus, the existing concept name will be used to launch the initial search for concepts.

Some mapped concepts may need to be replaced because the Semantic Connector did not parse the element name properly and performed a search on a term that is not applicable to the element, or returned a result that simply isn't in line with the intent of the element. In this case, one or more of the existing mapped concepts may need to be replaced with more accurate concepts from EVS.

To replace a mapped concept:

1. Select an element to view, and if necessary scroll through the detail view to find the concept you need to replace.
2. Click **Search EVS** located to the right of the Concept Code field. The Search Thesaurus dialog box appears and should contain search results based on the concept name of the concept you are replacing.
3. Since the automated search is probably what mapped the incorrect concept in the first place, change the text in the Search field to return results that better reflect the element concept you are trying to map.
4. Click **Search**.
5. Review the search results. If the concept you want to replace the existing concept with appears in the list, **double-click** anywhere in that row. The new concept information replaces the old concept information.

If the search results do not return any concepts you want to use, revise your search criteria and search again, or consider creating a new concept for the model. For information on creating a new concept for the model, see [Editing Concept Information While Changing the Concept Code](#) on page 85.

After making changes to the concept information, you can click **Apply** to apply those changes to that element, or select **Edit > Apply to All** to apply your changes to every instance of that element in the model. For more information, see [Applying Changes to All Similar Nodes](#) on page 86.

Changing the Order of Mapped Concepts in Curate Mode

As stated earlier in this section, every element must have a Primary Concept that reflects the basic point or meaning of the item, and then may have Qualifier Concepts that reflect additional details about the element. This is often true of attributes, which can reflect multiple details about the class element they belong to.

The order in which the concepts are mapped is reflected in the order given to those concepts in the SIW. This means that the first concept mapped is always the Primary Concept, while subsequently mapped concepts become Qualifier concepts with numbers that reflect when they were added (respectively). The order of these Qualifier concepts must reflect the intended semantic meaning of the element to which they are mapped.

The Semantic Connector uses the names of elements to map those elements to EVS concepts, parsing the names into individual parts based on the camel case naming convention. However, while curating the file you may discover that the concepts for an element are not mapped in an order that reflects the purpose for the element. In this case you will want to change the semantic order of the mapped concepts.

Once an element has more than one concept mapped to it, up and/or down arrow buttons appear to the right of the Concept Definition field in the detail view of the SIW viewer.

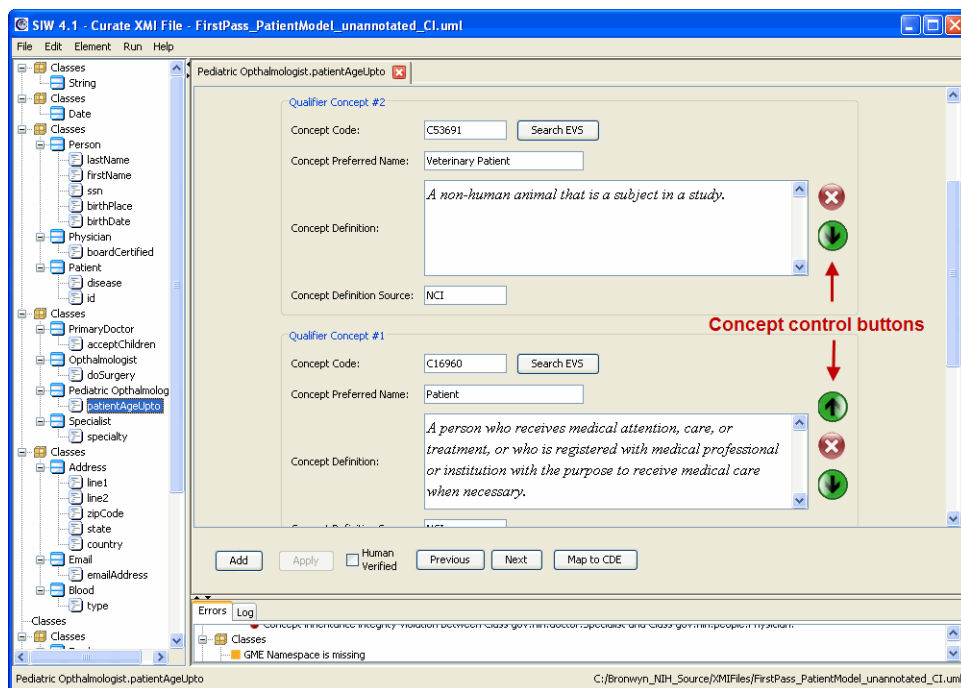


Figure 7.9 An attribute element with multiple concepts and the order control buttons.

The arrow buttons move the concept information either up or down (as indicated by the arrow selected) one position in semantic order for that item. The X button removes the concept from the element.

For example, an attribute named “initialDiagnosisDate” has a primary concept of Date, with two qualifying concepts: “Initial” and “Diagnosis”. But for whatever reason, the Qualifier concept #2 is “Diagnosis” and the Qualifier concept #1 is “Initial.” You believe these need to be switched in order to be mapped in proper semantic order.

In this case, clicking the “down” arrow next to Qualifier Concept #2 (“Diagnosis”) causes this concept information to switch places with Qualifier Concept #1 (“Initial”). This changes the semantics of the concept information, though the element itself is still mapped to those EVS concepts. (Be sure to click **Apply** to apply your change.)

The ability to rearrange the semantic order for concepts is most useful if you have to add or replace mapped concepts for the element. In that case, you can simply use the **Search EVS** button to find and add/replace concept information in any order you like, and then use the arrows to put the mapped concepts into the proper order, based on the attribute name and the UML element description provided by the model owner.

To change the semantic position of a concept:

1. Select an element from the navigation tree, and if necessary scroll through the detail view to find the concept you need to move.
2. Click the **Up Arrow** or **Down Arrow** located to the right of the Concept Definition field.

Notice that the concept is not simply moved; the concept switches places with the concept located next to it. This means that you will probably have to repeat these steps multiple times to get the concepts into the proper semantic order, moving concepts up and down in the list as necessary.

After making changes to the concept information, you can click **Apply** to apply those changes to that element, or select **Edit > Apply to All** to apply your changes to every instance of that element in the model. For more information, see [Applying Changes to All Similar Nodes](#) on page 86.

Deleting Mapped Concepts in Curate Mode

The Semantic Connector will typically map multiple concepts to an element when it finds multiple matching names for an element. In this case you will need to remove the mapped concepts that do not apply to that element.

For example, if the model contains a class or attribute with a name “id” it is quite likely that the Semantic Connector will map several different concepts to that element, including “identifier,” “Idaho” and “Indonesia” because the Semantic Connector cannot know what the intent of the element is. Therefore it selects and maps all matches it finds.

To remove a concept from the element:

- Click the **X** located next to the Concept Description field of the concept you want to remove. The concept is deleted.

After making changes to the concept information, you can click **Apply** to apply those changes to that element, or select **Edit > Apply to All** to apply your changes to every instance of that element in the model. For more information, see [Applying Changes to All Similar Nodes](#) on page 86.

Note: If you accidentally delete a concept that you did not intend to remove, click **Previous** or **Next** before clicking **Apply**. The SIW prompts you regarding whether or not to save the change. Click **No** and the concept information is returned to its original state. Please note that this “undo” action does NOT apply if you select **Apply to All**.

Editing Concept Information Without Changing the Concept Code

Within a model, two concepts with the same concept code must have the same concept name, definition and definition source. This means that if you change any of the properties of a concept without also changing the concept code, the SIW applies the property changes you made to *all* other elements within the model that use the same concept code.

For example, two classes “Person” and “Animal” have the “Name” attribute in common, because both elements have names. The “Name” attribute is annotated with the following concept information:

- `conceptCode: C25192`
- `conceptPreferredName: Name`
- `conceptDefinition: A word or group of words indicating the identity of a person usually consisting of ...`
- `conceptDefinitionSource: NCI Thesaurus`

You change the concept definition to state, “A term consisting of a word or group of words that indicates the specific identity of the item to which the term is applied.” However, you do not change the concept code. When you click **Apply**, the change you made is also made to *every instance of that concept in the model*.

The SIW does this for two reasons: first, concept code is a unique identifier, and everywhere a term in the model is identified with that code, it must reflect the same information. Second, if some aspect of a concept (like definition) within EVS changes for a particular term, that change needs to be applied across all systems (which is the point of semantic integration). Automating this type of change makes curation of XMI files a much more efficient process.

Editing Concept Information While Changing the Concept Code

Since the concept code is a unique identifier, if you change some aspect of the concept information for an element, but you don’t want that change to cascade to all instances of that concept, you must also change the concept code. Changing concept information while also changing the concept code essentially creates a new EVS concept. This is useful if the model contains an element that while is similarly named to other elements, does not map neatly to any existing EVS concepts and therefore requires its own description and definition.

If you make changes to the concept information and change the concept code to a new one, only the element you are editing is modified (as long as you are using a code that is *not* used elsewhere in the model). Those changes do not cascade to other elements.

For example, the model has a class named “Animal” that contains an attribute called “Breed.” The Semantic Connector mapped this to an EVS concept whose definition reflects the noun form of the term “breed” rather than the verb form. This model intends for the attribute to reflect the act of breeding instead of a type of animal.

During curation, you (the curator) can tell the intent of the attribute because of the UML definition provided by the model owner (see why these are important?). So you change the definition for the concept and also change the concept code so that your changes are only applied to this element. Once entered into EVS, there will be two EVS

concepts called “Breed” with two different concept codes and two different concept definitions.

Caution: If you do change the concept code for a mapped concept, be sure you are using a new code and *not* an existing one. If you change the code to one that is in use in the model, your changes will cascade to all elements mapped to the concept with that code.

You may find that you need to apply your concept changes to all similar elements in the model. For information on this feature, see [Applying Changes to All Similar Nodes](#) below.

Applying Changes to All Similar Nodes

In some cases, an attribute is reused several times across the model. A common example is the attribute “id.” Typically, this attribute represents the same type of information everywhere it is used across the model. As noted in an earlier example, the term “id” may apply to a list of different types of information, and therefore the Semantic Connector may have mapped several different concepts to this element. These extra concept mappings will need to be removed from every instance of the “id” attribute.

The **Apply to All** command, located in the Edit menu, allows you to apply the changes made to an element to all similar items in the model.

Apply to All is used in place of **Apply**. If you click **Apply** first, the **Apply to All** option is grayed out and no longer available.

Using the example of the “id” attribute, in the model, “id” occurs 27 times, and 25 of those times, refers to “Identifier.” All 27 instances of “id” have been mapped to the concept “Identifier” but they have also been mapped to “Idaho” and “Indonesia” by the Semantic Connector. You can remove these unnecessary concepts from one of the “id” attributes in the model, then select **Edit > Apply to All**. This removes those concepts from all instances of the attribute “id.”

You can then go back and manually update or replace the concept information for the two instances of “id” that refer to something other than “Identifier.”

The **Apply to All** feature is also useful if you are adding concepts to an element or reordering the concepts in an element and need to apply those same changes to all or most of the other instances of the element in the model. For information on adding concepts, see [Adding EVS Concepts to an Element in Curate Mode](#) on page 76. For information on editing concept information, see [Editing Concept Information in Curate Mode](#) on page 81.

Caution: If you use **Apply to All** you will no longer have the ability to “undo” your changes by selecting a different element in the model and then selecting not to apply the changes. Once **Apply to All** is used, all similar elements are changed to reflect the changes made in the currently active element. If you need to undo those changes, you can *exit the SIW without saving the changes to the file*, and then reopen the file. You will, however, lose all of your changes since you last saved the file.

Remember that applying changes is not the same as saving the file. You must still use the **File > Save** or **File > Save As** commands to save your changes to the file. For more information, see [Saving Changes to a File](#) on page 48.

Validating XMI File Concept Information Against EVS

The **Validate Concepts** option, located under the Run menu, performs a comparison of the Concept Code, Concept Name and Concept Definition fields for each mapped element in the currently open XMI file to the “official” information located in EVS. This allows you to easily see where there are differences or where there is concept information in the model that is not resident in EVS.

To Validate the XMI concepts against EVS concepts:

1. With the annotated XMI file open in Review Annotated XMI File mode, select **Run > Validate Concepts**.
2. A confirmation dialog box appears, informing you that the validation process may take some time to complete, and asking if you want to continue. Click **Yes**.
3. An empty Validate Concepts window appears, with a progress bar at the bottom informing you of the status of the validation process.

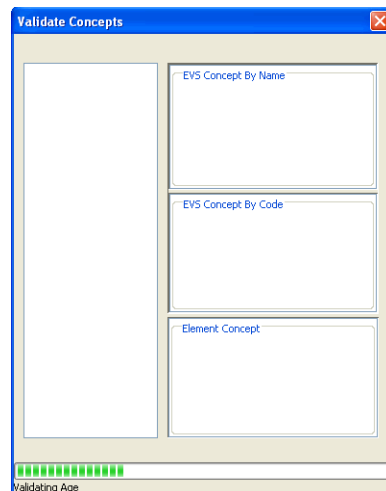


Figure 7.10 Validate Concepts window showing progress of the validation

When the validation process completes, the progress bar spans the bottom of the window, and a status of **Done** appears below the progress bar. In addition, the left side of the window will list all elements in the XMI file that contain concept information that differs from what appears in EVS.

If the left side of the window is empty, there are no disparities between the XMI file and EVS.

4. Select one of the elements from the list on the left side of the Validate Concepts window. This populates the other areas of the window with concept name, concept code and concept definition information as follows:

- **EVS Concept by Name** - The area located on the top-right of the Validate Concepts window shows the entry in EVS for the concept name that matches the concept name used for the selected element in the XMI file.

If this area is empty, it means that no match was found in EVS for the concept name listed for the selected element.
- **EVS Concept by Code** - The area located in the middle-right of the Validate Concepts window shows the entry in EVS for the concept code that matches the concept code used for the selected element in the XMI file.

If this area is empty, it means that no match was found in EVS for the concept code listed for the selected element.
- **Element Concept** - The area located on the bottom-right of the Validate Concepts window shows the concept information resident in the XMI file that corresponds with the EVS concept(s) appearing in the upper two boxes. This allows you to directly compare what is in the XMI file with the corresponding information in EVS.

In addition to these informational areas, the top of the Validate Concepts window shows the type of element and type of disparate concept currently selected (e.g., Class Primary Concept or Attribute Qualifier Concept #2).

The Validate Concepts window highlights the fields where there are differences between the information in EVS and in the XMI file.

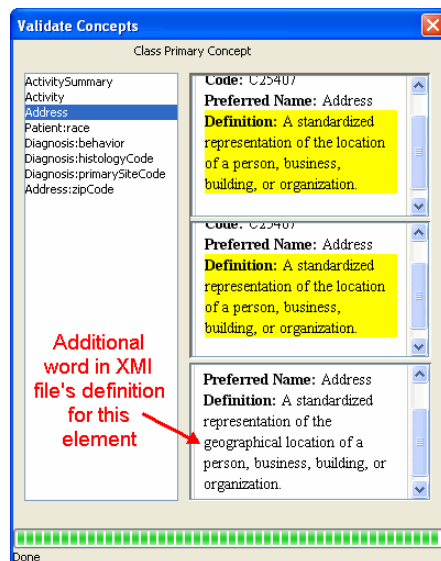


Figure 7.11 Validate Concepts window showing a difference in concept definition

You can make changes to the information in the Element Concept section of the Validate Concepts window, which makes those changes to the element in the XMI file.

In addition, when you select an element in the Validate Concepts window, the same item is also selected in the navigation tree of the main SIW viewer window. This allows you to review all of the other concept information for that element if needed.

Verifying the Curated XMI File

Once you have reviewed and updated the concept information in the file to your satisfaction, you can use the Human Verified checkbox to mark that curation for this element is complete. As is logical, the Human Verified checkbox is not available for any element that does not contain concept information or which is not already mapped to a CDE or OC.

Once an element has been checked as Human Verified, a checkmark also appears next to the element in the navigation tree. This makes it very easy for you to identify which elements still need to be reviewed and verified.

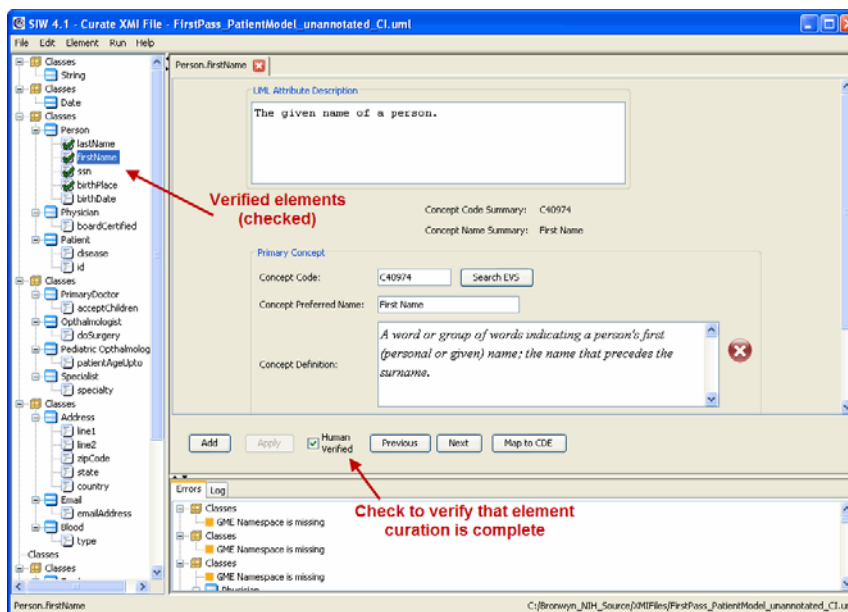


Figure 7.12 SIW Viewer in Curate XMI File mode, with human verified elements

After the file is fully curated, you must mark **all** elements as **Human Verified**, to mark the elements in the navigation tree are marked with a checkmark. This includes the elements that you either mapped or verified the mapping of concept information for, and the elements that are already mapped to CDEs or OCs.

To verify an element:

1. Select an element from the navigation tree.
2. Review the information in the detail view to verify that all of the necessary information is mapped to the element.
3. Click the **Human Verified** checkbox to enable it. One of the following occurs, depending on the element you are verifying:
 - If you have selected a class element and there are un-verified attributes associated with that class, the next item in the tree is automatically selected for you to verify.
 - If you have selected an attribute, a checkmark appears on the attribute in the navigation tree and the next item in the tree is automatically selected.

- If you have selected the last attribute in the class to be verified (all others have already been verified), and the class has already been verified, a checkmark appears on the attribute you are verifying and on the class element, and the next item in the tree is automatically selected.

The automatic movement of the SIW to the next element during element verification is simply a navigational aid, so that you do not have to continually click **Next** to move to the next item. Once an element is verified, the SIW assumes that you are finished with that item.

Once you have reviewed all items in the file, and cleared all of the errors listed in the Errors tab (which may require multiple launches of the SIW and the XMI file to update the Errors tab), you must save the file using the **File > Save As** command. Curated and verified files should be saved with the term “annotated” appended to the front of the filename. For example, if the original file name was `FirstPass_myModel.xmi` then you would save the file as `Annotated_FirstPass_myModel.xmi`. If the original file had a `.uml` extension (exported from ArgoUML) then the curated file must also be saved with a `.uml` extension.

For further instructions on saving a file, see [Saving Changes to a File](#) on page 48.

Next Step - Review the Annotated XMI File

Once the curation team is finished curating and marking the model elements as verified, they return the XMI file to the model owner for review and verification. The model owner can simply check and verify the curated elements, or may choose to make changes to the curation done by the curation team.

Either way, the next step in the process is to open and review the curated file in the Review Annotated XMI File option of the SIW. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91.

CHAPTER 8

REVIEWING AN ANNOTATED FILE

This chapter provides detailed information on how you can use the SIW to perform or edit concept mappings in your XMI file, or simply review and approve the mappings done by the EVS curation team.

Topics in this chapter include:

- *Purpose for Reviewing an Annotated XMI File*, below
- *Opening an Annotated XMI File for Review* on page 93
- *Changing Concept Information in Review Mode* on page 95
- *Re-Verifying the Reviewed XMI File* on page 105

There are effectively two “parts” to reviewing an annotated XMI file: making changes to the annotations, and marking the annotations as “model owner verified.” Whether you are making a first pass at doing your own concept mapping, or simply looking at what the EVS team has done, marking the elements as verified lets the CBIIT staff know that you have looked at each of the checked elements in the file.

While this chapter contains procedural information for using this SIW option, you will find references to [Chapter 7, Curating An XMI File](#), on page 73 that point you to further details on these functions. If you plan to use this SIW option to edit the concept mapping in your XMI file (either before handing it off to the EVS team, or as an iterative step with the EVS curator), you are strongly encouraged to review the information in Chapter 7.

Purpose for Reviewing an Annotated XMI File

The **Review Annotated XMI File** step is always performed by the model owner after the EVS curation team finishes annotating the file and returns the annotated file. It is also sometimes used immediately after the Semantic Connector, if the model owner wishes to perform their own concept mapping and editing prior to handing the file off to the EVS curation team.

Using this option of the SIW, the model owner can make changes or corrections to the concepts mapped to the model elements, and/or simply verify the mappings contained in the annotated XMI file. Users can search EVS for concepts to change the concept mapping for a class or attribute, or users may choose to map UML attributes to existing caDSR value domains or data elements where valid, applicable caDSR items already exist.

The main difference between working with a file in the Review Annotated XMI File mode as opposed to the Review Unannotated XMI File or Curate XMI File mode is that the Review mode shows *all* of the items from the UML model that will be uploaded into caDSR. The Review mode also shows concept errors and other problems with the file that were not relevant to an unannotated file.

When opening a file in the Review Annotated XMI File step, the SIW performs a number of validation checks to ensure that the XMI file can be correctly transformed into caDSR metadata. For each UML class and attribute in the file, the SIW checks for the presence of at least one of each of the following:

- Concept code
- Concept name
- Concept definition
- Concept definition source
- Valid datatype

Furthermore, if the file has already passed the Curate XMI File step, each element was checked as Human Verified by the EVS curator. When the annotated file is opened in Review Annotated XMI File mode, those checkmarks are reset so that the model owner can review each element in the file and when satisfied that the element is complete, check the element as Model Owner Verified.

If any of the elements in the file are missing any of the required mapping items, the SIW lists an error in the Errors tab, allowing the model owner to review the error and make changes to the element mapping as appropriate.

Note: All errors must be corrected before a model can be loaded to the caDSR.

Reminder about the Errors tab—The SIW generates the Errors tab information when the file is opened; it cannot update the error listing dynamically. This means that as you make changes, you will need to save the file, exit the SIW, then reopen the saved file in the Review Annotated XMI File mode to see an updated error listing.

Input: The Annotated XMI file as curated by the EVS concept curation team. The file is returned to the model owner with the term “Annotated” added to the beginning of the file name.

Example: `Annotated_FirstPass_myModel.xmi` or `Annotated_FirstPass_Roundtrip_myModel.xmi` for EA generated models. For ArgoUML-generated models, the file names would end with a `.uml` extension.

Output: An Approved or Fixed Annotated XMI file. If you are using this SIW option to simply review EVS concept team mappings, when you save the file, you should

append it with the term “approved” at the front of the filename. For example:

Approved_Annotated_FirstPass_myModel.xml or

Approved_annotated_FirstPass_Roundtrip_myModel.xml. For ArgoUML generated models, the file names would end with a .uml extension.

If you are using this step to make changes to the concept mapping and will be returning the file to the EVS curation team, you may choose to save the file with the term “fixed” appended to the front, rather than “approved.” In addition, you may also want to employ the use of revision numbering in the file name, so that you can delineate between the different versions of processed files (e.g., fixed_r2_annotated_FirstPass_myModel.xml).

Ultimately when you have completed your final review and approval of a fully annotated file, the end result of the Review XMI file step is a file that contains no errors, and where all of the elements in the file have been checked as “Model Owner Verified.” Once you have reviewed the concept mappings for each element in the file and mapped elements to caDSR CDEs where appropriate, you can then finally verify all of the elements in the file and save the file as an “Approved Annotated” version of the XMI file. The approved annotated XMI file is the file that will be passed on to CBIT for logging into CVS and loading to caDSR.

Opening an Annotated XMI File for Review

This section provides procedures for opening an XMI file for review and if necessary editing. Subsequent sections of this chapter provide details on how to make changes to the annotations of the elements in the file.

To Review an Annotated XMI file:

1. Be sure you have saved the annotated (curated) XMI file you received from the EVS curators.
2. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.
3. From the SIW Welcome screen, select option **5. Review Annotated XMI File (Model Owner)** and click **Next**. The file selection screen appears.

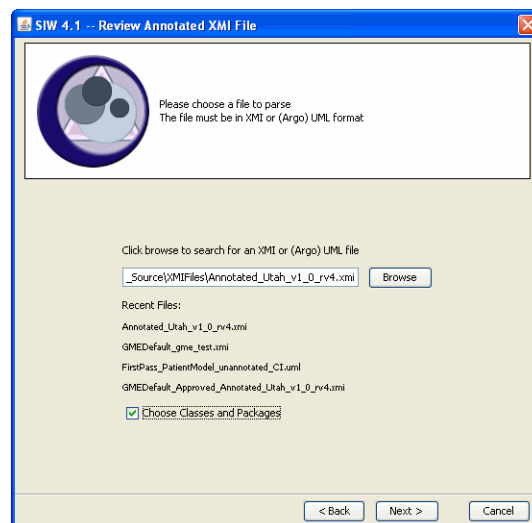


Figure 8.1 Select a file for SIW to parse and display for review

If you have used the SIW before, the last five files used for this option appear in a Recent Files list located below the filename text box.

4. Identify the file you want to review. You have the following options:
 - If the file you want to review appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename.
 - **Type** the full path and filename into the text box.
 - Click **Browse** to browse for and select the file exported from EA or ArgoUML.

Note: The Browse function defaults to selecting (showing) only files with an .xmi file extension. If you are using a .uml file, select **UML Files** from the **Files of Type** drop-down list at the bottom of the Open dialog box.

5. If you want to review only portions of the file, enable the **Choose Classes and Packages** checkbox. Otherwise the SIW will parse the entire file for review.
6. Click **Next**. If you enabled the **Choose Classes and Packages** checkbox, the package selection screen appears, listing all of the packages in the file.

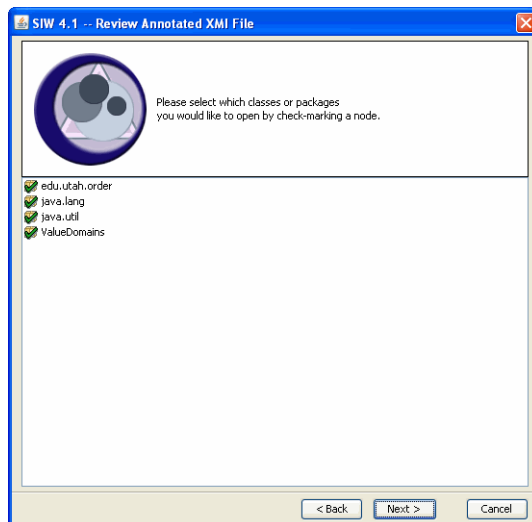


Figure 8.2 Package selection screen - lists the packages resident in the file

7. All packages listed are checked by default. Click a **checkmark** to open a list of all items contained within the package, and then click the checkmarks next to any items you do not want the SIW to parse for review. Then click **Next**.

The SIW displays the file selection screen with a progress bar at the bottom showing the current action and overall progress. How long this process will take is determined by the size of the XMI file or packages you are using as well as the number of errors or warnings the SIW must generate.

When the SIW is finished parsing the annotated file, the SIW viewer appears showing the selected file (or selected packages/classes).

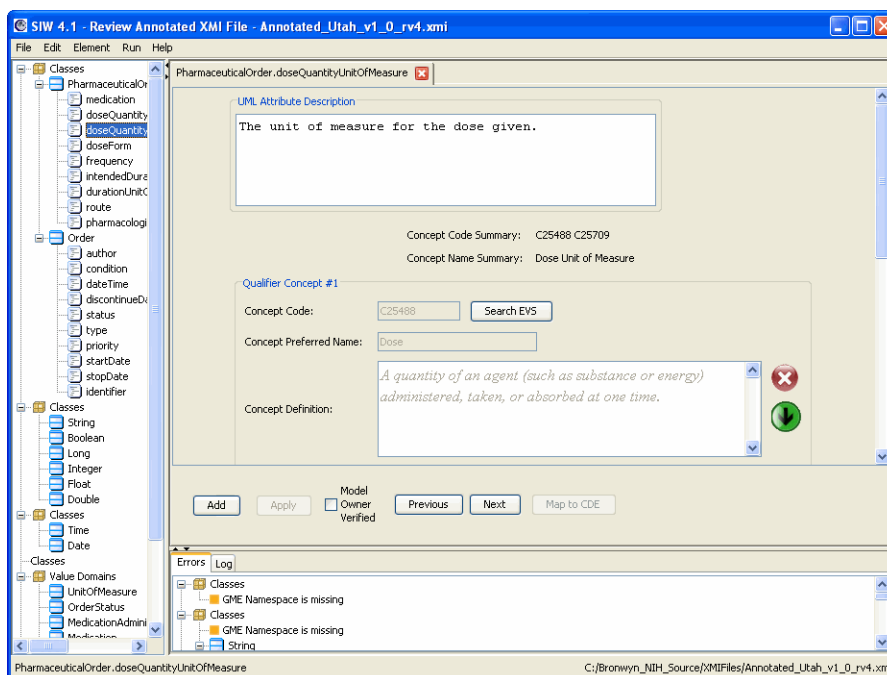


Figure 8.3 SIW Viewer - showing an Annotated XML file with errors

Use the SIW viewer to review the mappings for each element, and where listed, the errors encountered in the file, to determine what changes need to be made in the file.

If necessary, see [Using the SIW Viewer](#) on page 35 for a basic overview on navigating the SIW viewer and on using the **Previous**, **Next**, **Add**, **Apply** and **Search EVS** buttons.

You are also *strongly* encouraged to review [Editing Concept Information in Curate Mode](#) on page 81. This section provides detailed information on making changes to the concept information of your file, including searching EVS, replacing mapped concepts, reordering concepts, and removing concepts for an element.

Changing Concept Information in Review Mode

While reviewing the annotated file, you may find that you need to make changes to the concept information mapped to some of the elements. The information in this section provides basic procedures for how to add, replace, reorder and remove concept information for an element. For more detailed information on these features, see [Adding EVS Concepts to an Element in Curate Mode](#) on page 76 and [Editing Concept Information in Curate Mode](#) on page 81.

If you do make changes to the concept information in your model, you may want to validate the concepts in your XML file against those in EVS. The **Validate Concepts** option, located under the **Run** menu, performs a comparison of the concept information in your XML file against the concept information in EVS. For more information on using this feature, see [Validating XML File Concept Information Against EVS](#) on page 87.

Note: If you accidentally make changes to concept information that you did not intend to make or don't want to keep, click **Previous** or **Next** *before* clicking **Apply**. The SIW prompts you regarding whether or not to save the change. Click **No** and the element's concept information is returned to its original state.

Adding EVS Concepts to an Element in Review Mode

Whether you are working with a curated file or with an XMI file before curation, you may find that some elements require the addition of concepts.

To add a concept:

1. In the SIW viewer, select an element from the navigation tree.
2. In the detail view, click **Add**. A set of empty concept fields appears, along with a **Search EVS** button.

Note: If you have changed your viewer Preferences to Display Primary Concept First, the new concept fields will appear below the existing mapped concepts, meaning you may need to scroll down in the detail view to see the new fields. See [Setting Viewer Preferences](#) on page 44 for more information.

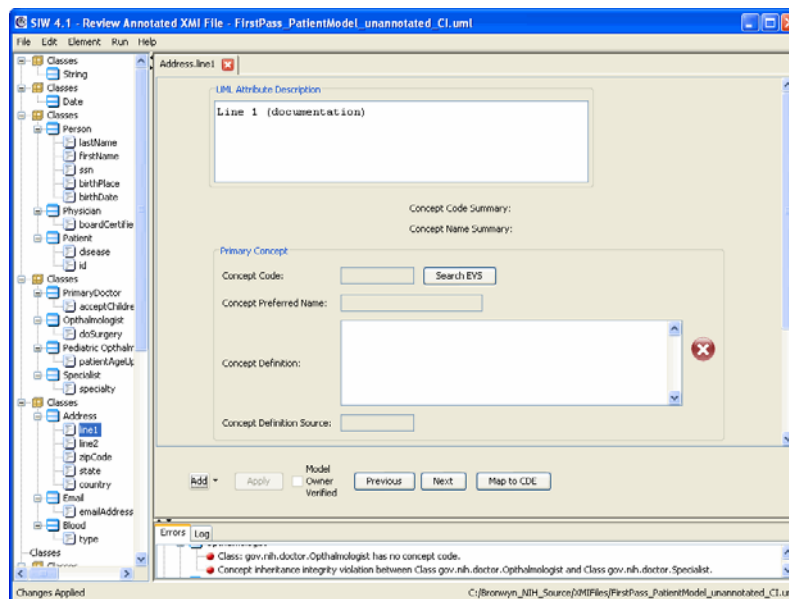


Figure 8.4 Review Annotated XMI mode, showing new blank concept fields.

3. Click **Search EVS** to perform a search for the appropriate concept to map.
4. Enter the item you want to search for into the Search text box, select whether you are searching for a **Synonym** or **Concept Code**, then click **Search**.
5. When the concept you want to add appears in the search result set, **double-click on the item** to populate the concept fields with the information. For more details on using the EVS Search to populate the concept fields, see [Searching EVS for Concept Information](#) on page 78.

6. When the concept fields are complete, click **Apply**. This applies the new concept to the element. (You will still have to save the file after all of your changes are made. See [Saving Changes to a File](#) on page 48.
7. If necessary, after applying your changes, you can click **Add** again and complete the above steps to add another concept to this element.

Replacing Mapped Concepts in Review Mode

Some mapped concepts may need to be replaced because the concept information mapped to them is not exactly what you intended for the item. In this case, you can search EVS for a better match and replace the existing concept.

Replacing mapped concepts is the same as adding concepts to an element, except that the concept information fields are already populated. Unless you have changed your default viewer Preferences, clicking **Search EVS** automatically launches a search of the NCI Thesaurus using the existing concept name of the item. See [Setting Viewer Preferences](#) on page 44 for more information.

Note: If you cannot find a matching concept in EVS, contact the EVS curation team to work with them on finding an existing concept or creating a new one.

To replace a mapped concept:

1. Select an element to view, and if necessary scroll through the detail view to find the concept you want to replace.
2. Click **Search EVS**, located to the right of the Concept Code field of the concept you want to replace. The Search Thesaurus dialog box appears and should contain search results based on the text in the Concept Name field of the concept you are replacing.
3. Since the automated search may be how the incorrect concept was mapped in the first place, you will probably want to change the text in the Search field to return results that better reflect the element concept you are trying to map.
4. Click **Search**.
5. Review the search results. If the concept you want to use appears in the list, **double-click** anywhere in that row. The new concept information replaces the old concept information.

If the search results do not return any concepts you want to use, revise your search criteria (using a wildcard [*] if necessary) and search again. If you cannot find a matching concept in EVS, contact the EVS curation team to work with them on finding an existing concept or creating a new one.

After making changes to the concept information, click **Apply** to apply the changes to that element. If you attempt to navigate away from the element without applying your changes, the SIW prompts you to apply or discard your changes.

Changing the Order of Mapped Concepts in Review Mode

Every element must have a Primary Concept that reflects the basic point or meaning of the item, and then may have Qualifier Concepts that reflect additional details about the element. This is often true of attributes, which can reflect multiple details about the class element to which they belong.

The order in which the concepts are mapped reflects the order given to those concepts in the SIW. This means that the first concept mapped is always the Primary Concept, while subsequently mapped concepts become Qualifier concepts with numbers that reflect when they were added (respectively). The order of these Qualifier concepts must reflect the intentional semantic meaning of the element to which they are mapped.

However, you may discover that the concepts for an element are not mapped in an order that reflects your purpose for the element. In this case you will want to change the semantic order of the mapped concepts.

For more information on the semantic ordering of mapped concepts, see [Changing the Order of Mapped Concepts in Curate Mode](#) on page 82.

Once an element has more than one concept mapped to it, up and/or down arrow buttons appear to the right of the Concept Definition field in the detail view of the SIW viewer.

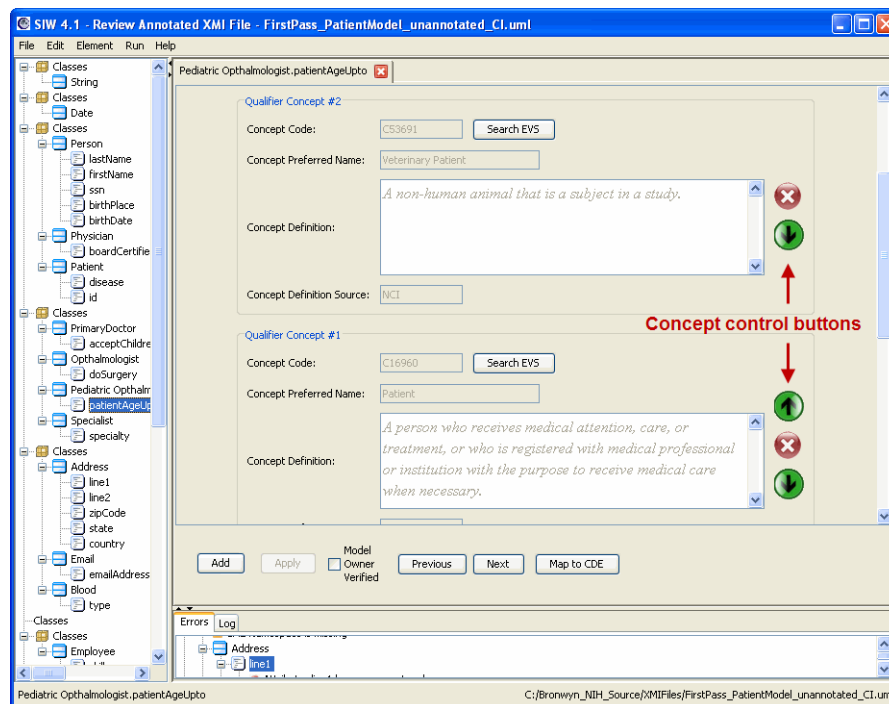


Figure 8.5 An attribute element with multiple concepts and the order control buttons.

The arrow buttons move the concept information either up or down (as indicated by the arrow selected) one position in semantic order for that item. The X button removes the concept from the element.

The ability to rearrange the semantic order for concepts is most useful if you have to add or replace mapped concepts for the element. In that case, you can use the **Search EVS** button to find and add/replace concept information in any order you like, and then

use the arrows to put the mapped concepts into the proper order, based on the intended purpose of the element in the model.

To change the semantic position of a concept:

1. Select an element to view, and if necessary scroll through the detail view to find the concept you need to move.
2. Click the **Up Arrow** or **Down Arrow** located to the right of the Concept Definition field.

Notice that the concept is not simply moved; the concept switches places with the concept located next to it. This means that you will probably have to repeat these steps multiple times to get the concepts into the proper semantic order, moving concepts up and down in the list as necessary.

When finished, look at the **Summary Concept Name** listed for the element, and be sure it reflects your intended meaning or use of the element within the model.

After making changes to the concept information, click **Apply** to apply the changes to that element.

Deleting Mapped Concepts in Review Mode

If you have added unnecessary or inaccurate concepts to an element, or find that the curation process left concepts that should not be mapped to an element, you can delete them. However you may only remove concepts if there are multiple concepts mapped to an element (because each element must have a Primary concept). If your element contains only one concept, you must add a new concept before you can delete the existing one. See [Adding EVS Concepts to an Element in Review Mode](#) on page 96.

To remove a concept from an element:

- Click the **X** located next to the Concept Description field of the concept you want to remove. The concept is deleted.

After making changes to the concept information, click **Apply** to apply those changes to the element.

Note: If you accidentally delete a concept that you did not intend to remove, click **Previous** or **Next** before clicking **Apply**. The SIW prompts you regarding whether or not to save the change. Click **No** and the concept information is returned to its original state. Please note that this “undo” action does NOT apply if you select to **Apply to All**. In that instance you must exit the SIW without saving your changes and re-open the file. You will lose all of your change since the last time you saved the file.

Mapping UML Attributes to caDSR CDEs

Each UML attribute in your model can either be mapped to one or more EVS concepts or mapped to a caDSR CDE. Mapping an attribute to a CDE automatically maps the class element to the caDSR OC of the CDE and the datatype for the attribute to the value domain of the CDE. For more information on the relationship between UML model elements and caDSR CDEs, see [Terms You Should Know](#) on page 16 and [Using Pre-mapped CDEs \(Common Data Elements\)](#) on page 131.

While you are reviewing your annotated file, you may find that you can map some of the attributes in your model to CDEs. This mapping provides direct connectors between the items in your model and caDSR metadata, and improves the semantic integration of your model with other caCORE-compatible systems.

When viewing an attribute in Review Annotated XML File mode, for any attribute not already mapped to a CDE, the detail view provides a **Map to CDE** button. Clicking **Map to CDE** displays a different detail view including a **Search Data Element** button that allows you to search the caDSR for CDEs to map to the selected attribute.

When you click **Map to CDE**, notice that it turns into a **Map to Concepts** button. This allows you to toggle between the CDE mapping and EVS concept mapping views for the selected attribute. Once you have applied to mapping of the attribute to a CDE however, the concept information no longer appears, in favor of the CDE mapping.

Note: The **Map to CDE** button is not available if the attribute uses a LVD. This is because mapping an attribute to a CDE automatically maps the caDSR value domain as well. For more information on creating and using LVDs for your model instead of caDSR value domains, see [Using Local Value Domains](#) on page 23.

Mapping a UML Attribute to an Existing Common Data Element

Any attribute in the UML model can be mapped to an existing caDSR CDE. In addition, you can replace a CDE mapping if it is incorrect or needs to be updated to use a newer version of the CDE.

To map an attribute to a CDE:

1. Select an attribute in the navigation tree. If the attribute has not already been mapped to a CDE, concept information appears in the detail view along with a **Map to CDE** button. If the element is already mapped to a CDE, skip to Step 3

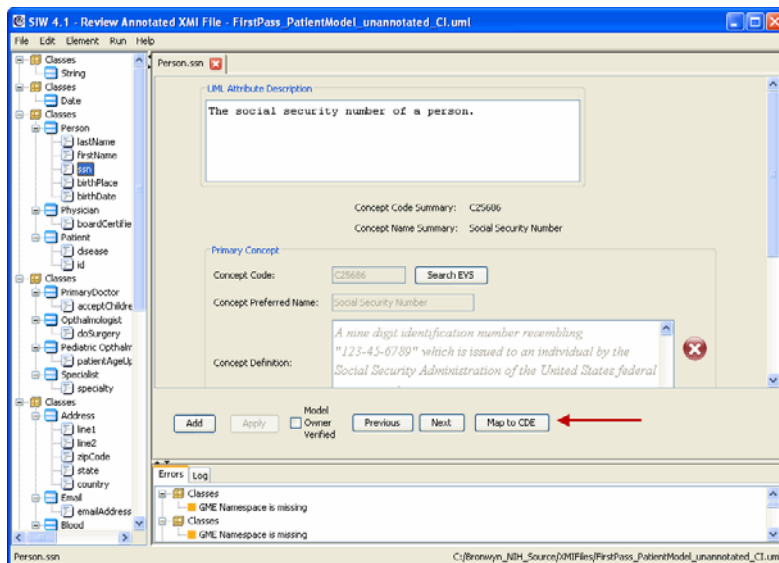


Figure 8.6 SIW Viewer Detail View showing Map to CDE button

2. Click **Map to CDE**. The detail view changes to show empty data element fields. In addition two new buttons appear: **Search Data Element** and **Clear**. Also notice that the **Map to CDE** button has changed to **Map to Concepts**.

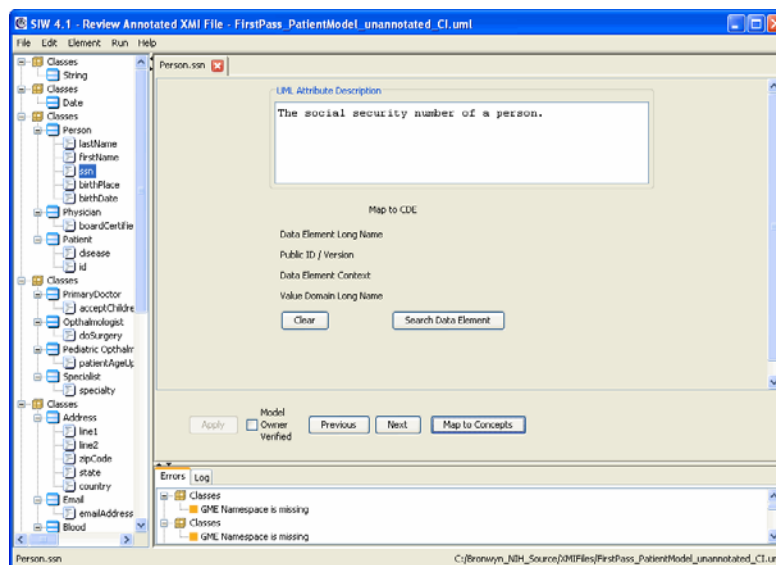


Figure 8.7 Detail View after the Map to CDE button is clicked

3. Click **Search for Data Element**. This opens the Search for Data Element dialog box, allowing you to search the caDSR for CDEs to map your attribute.

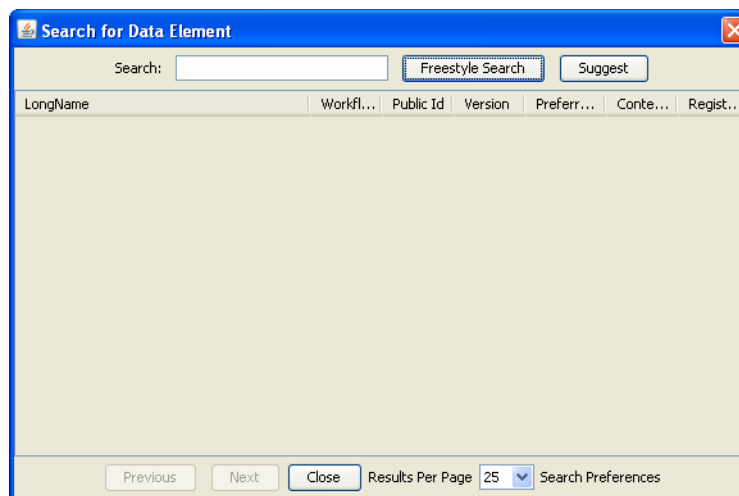


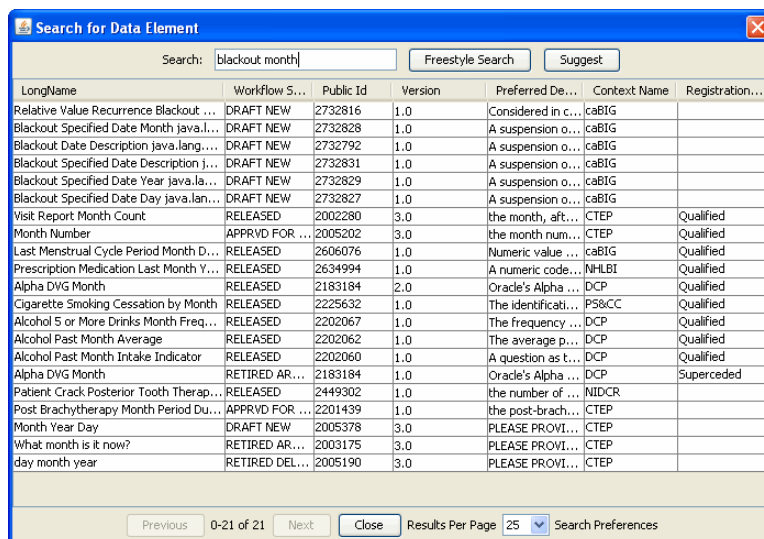
Figure 8.8 Search for Data Element dialog box

Note: If you have conducted a previous search for a data element during the current session, the Search for Data Element dialog box will be populated with the previous search information/results.

4. In the Search for Data Element dialog box you have two choices:
 - Enter a term in the Search field and click **Freestyle Search**. This search takes advantage of the Freestyle search engine which searches a variety of fields in the caDSR for the term entered (<http://freestyle.nci.nih.gov/freestyle/do/search>). You may use the asterisk (*) as a wildcard character.

- ° Click **Suggest**. This searches the caDSR for any element with an alternate name that matches the attribute name you are attempting to map. **Suggest** uses only the attribute name ignoring any information in the Search text box.

The results appear below the search options.



The screenshot shows a dialog box titled "Search for Data Element" with a search bar containing "blackout month" and buttons for "Freestyle Search" and "Suggest". Below the search bar is a table with the following columns: LongName, Workflow S..., Public Id, Version, Preferred De..., Context Name, and Registration... The table lists various data elements, including "Relative Value Recurrence Blackout...", "Blackout Specified Date Month java.lang...", "Blackout Date Description java.lang...", "Blackout Specified Date Description j...", "Blackout Specified Date Year java.la...", "Blackout Specified Date Day java.lan...", "Visit Report Month Count", "Month Number", "Last Menstrual Cycle Period Month D...", "Prescription Medication Last Month Y...", "Alpha DVG Month", "Cigarette Smoking Cessation by Month", "Alcohol 5 or More Drinks Month Freq...", "Alcohol Past Month Average", "Alcohol Past Month Intake Indicator", "Alpha DVG Month", "Patient Crack Posterior Tooth Therap...", "Post Brachytherapy Month Period Du...", "Month Year Day", "What month is it now?", and "day month year". The status of each element is indicated in the "Registration..." column, with values like "Qualified", "Superceded", and "CTEP". At the bottom of the dialog box, there are buttons for "Previous", "Next", "Close", and a "Results Per Page" dropdown set to "25".

LongName	Workflow S...	Public Id	Version	Preferred De...	Context Name	Registration...
Relative Value Recurrence Blackout ...	DRAFT NEW	2732816	1.0	Considered in c...	caBIG	
Blackout Specified Date Month java.l...	DRAFT NEW	2732828	1.0	A suspension o...	caBIG	
Blackout Date Description java.lang...	DRAFT NEW	2732792	1.0	A suspension o...	caBIG	
Blackout Specified Date Description j...	DRAFT NEW	2732831	1.0	A suspension o...	caBIG	
Blackout Specified Date Year java.la...	DRAFT NEW	2732829	1.0	A suspension o...	caBIG	
Blackout Specified Date Day java.lan...	DRAFT NEW	2732827	1.0	A suspension o...	caBIG	
Visit Report Month Count	RELEASED	2002280	3.0	the month, aft...	CTEP	Qualified
Month Number	APPRVD FOR ...	2005202	3.0	the month num...	CTEP	Qualified
Last Menstrual Cycle Period Month D...	RELEASED	2606076	1.0	Numeric value ...	caBIG	Qualified
Prescription Medication Last Month Y...	RELEASED	2634994	1.0	A numeric code...	NHLBI	Qualified
Alpha DVG Month	RELEASED	2183184	2.0	Oracle's Alpha ...	DCP	Qualified
Cigarette Smoking Cessation by Month	RELEASED	2225632	1.0	The identificati...	PS&CC	Qualified
Alcohol 5 or More Drinks Month Freq...	RELEASED	2202067	1.0	The frequency ...	DCP	Qualified
Alcohol Past Month Average	RELEASED	2202062	1.0	The average p...	DCP	Qualified
Alcohol Past Month Intake Indicator	RELEASED	2202060	1.0	A question as t...	DCP	Qualified
Alpha DVG Month	RETIRED AR...	2183184	1.0	Oracle's Alpha ...	DCP	Superceded
Patient Crack Posterior Tooth Therap...	RELEASED	2449302	1.0	the number of ...	NIDCR	
Post Brachytherapy Month Period Du...	APPRVD FOR ...	2201439	1.0	the post-brach...	CTEP	
Month Year Day	DRAFT NEW	2005378	3.0	PLEASE PROVI...	CTEP	
What month is it now?	RETIRED AR...	2003175	3.0	PLEASE PROVI...	CTEP	
day month year	RETIRED DEL...	2005190	3.0	PLEASE PROVI...	CTEP	

Figure 8.9 Search for Data Element dialog box with search results

If there are multiple pages of results, the **Previous** and **Next** buttons are enabled, allowing you to scroll through the results pages. In addition, you can resize both the search window and the columns within the results table, if necessary, to see more information in each field.

5. If the search returns no results (the Search dialog box remains empty), the element you want to map does not appear in the result set, or the search returns too many results, alter your search criteria and/or use the wildcard (*) to return different results. You may also choose to click **Close** to close the Search dialog box without mapping the attribute.
6. If the CDE you want to map appears in the result set, **double-click** anywhere in the row for the item. This closes the Search dialog box and enters the information for the selected CDE into the fields in the SIW viewer. It also maps (or verifies) the OC to the class associated with the attribute and the value domain to the datatype for the attribute.

Note: It is recommended that only data elements with a status of “Released” be used to map to model attributes.

You may receive an invalid selection error informing you that the DE you selected is mapped to a different OC than the current class element. This means that the selected element cannot be used for this mapping, and you must select a DE containing an OC that corresponds to the OC already mapped for the class element.

Another error you may encounter is that mapping the attribute to the selected DE causes a duplicate mapping. A duplicate mapping occurs when a different attribute in your model has already been mapped to the selected CDE.

If you select a valid CDE, the data element information appears in the detail view for the attribute.

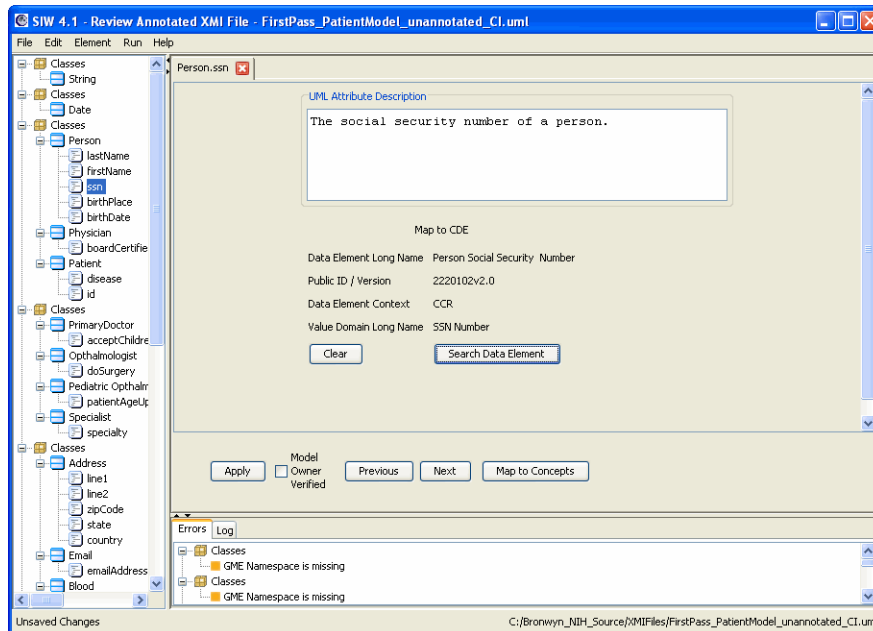


Figure 8.10 Viewer showing an attribute mapped to a CDE (but not yet applied)

Once you have selected the CDE to map to the attribute, you have the following options:

- Click **Map to Concepts** to return to the concept information view. You may toggle between the concept and the CDE view for the attribute until the CDE mapping is applied. Once an attribute is mapped to a CDE, the concept information no longer appears.
- Click **Clear** to undo the choice of the CDE and then click **Apply**. This removes the CDE information mapping from the attribute.
- Click **Apply** to map the CDE to the attribute.

When an attribute is mapped to a CDE, the class that the attribute belongs to is mapped to the OC for the CDE selected, and the datatype for the attribute is mapped to the value domain for the CDE. If this is the first attribute for the class being mapped to a CDE, the SIW pops up a note informing you of the OC-to-class mapping.

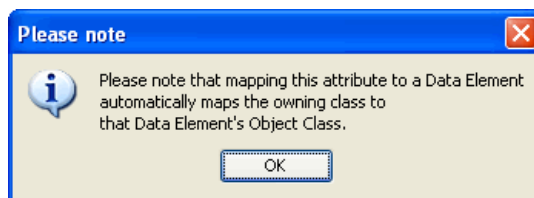


Figure 8.11 Pop-up dialog box informing you of the mapping of the class to the DE's Object Class

Like the CDE mapping for the attribute, the OC mapping for the class replaces the concept information displayed for the class.

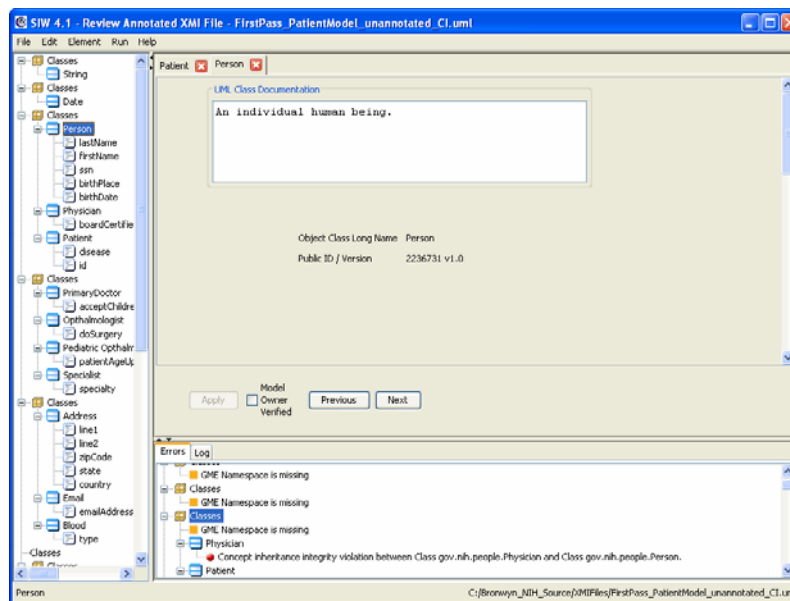


Figure 8.12 Viewer showing a class element mapped to a caDSR Object Class

Mapping a UML Attribute to a Value Domain

Mapping an attribute to a CDE automatically maps the datatype for the attribute to the value domain for the CDE. If an attribute is not mapped to a CDE, you can still map the attribute's datatype to a value domain from caDSR or to a local value domain. In the event the attribute is already mapped to a value domain, you can also clear the value domain mapping.

The drop-down list for mapping an attribute to a value domain appears below the concept information for the attribute in the detail view.

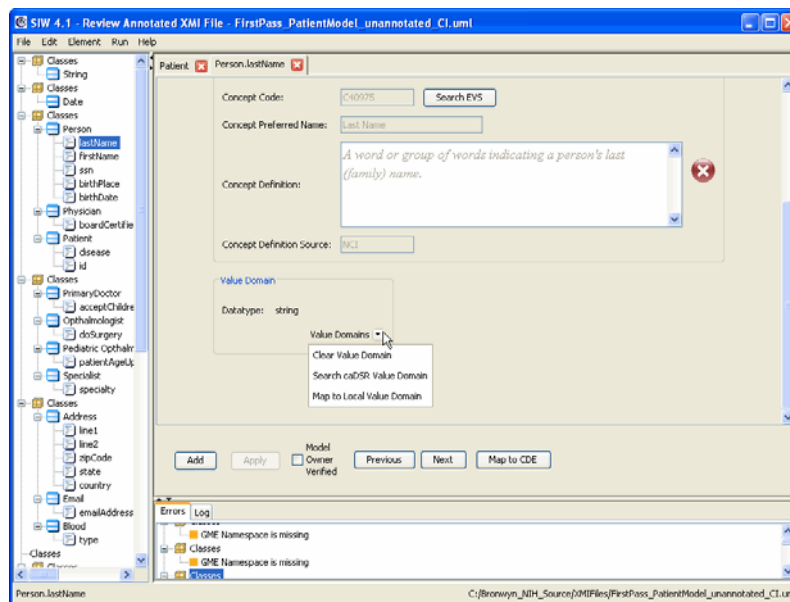


Figure 8.13 The Value Domain information box is located below the concept information

Clicking the **Search Value Domain** option from the list opens the **Search for Value Domain** dialog box.

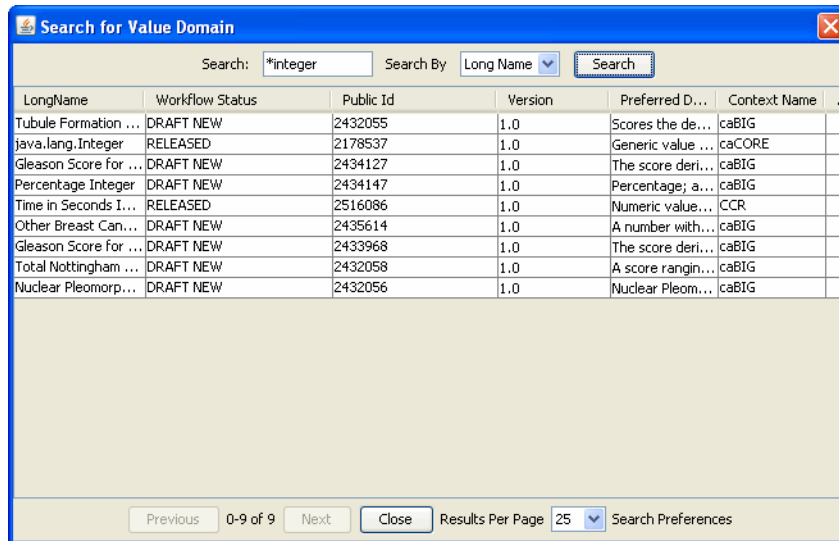


Figure 8.14 Search for Value Domain from caDSR to map to the selected attribute

The Search for Value Domain dialog box functions similarly to other search functions in the SIW, allowing for the use of an asterisk (*) as a wildcard, and for resizing the dialog box and columns as necessary to better view the information.

By default, the search looks for a LongName that matches the information entered into the **Search** text box. You can also select to search by **Public ID**.

As with the other search functions, **double-clicking** an item in the result set maps the selected value domain to the selected attribute in the model.

After making changes to the value domain mapping, click **Apply** to apply those changes to the element.

Re-Verifying the Reviewed XMI File

Once you are finished reviewing the concept information in the file and where possible mapping elements to caDSR CDEs, and the XMI file is annotated to your satisfaction, you can use the **Model Owner Verified** checkbox to mark that the annotation for each element is complete.

Once an element has been checked as Model Owner Verified, a checkmark also appears next to the element in the navigation tree. This makes it very easy for you to identify which elements still need to be reviewed and verified.

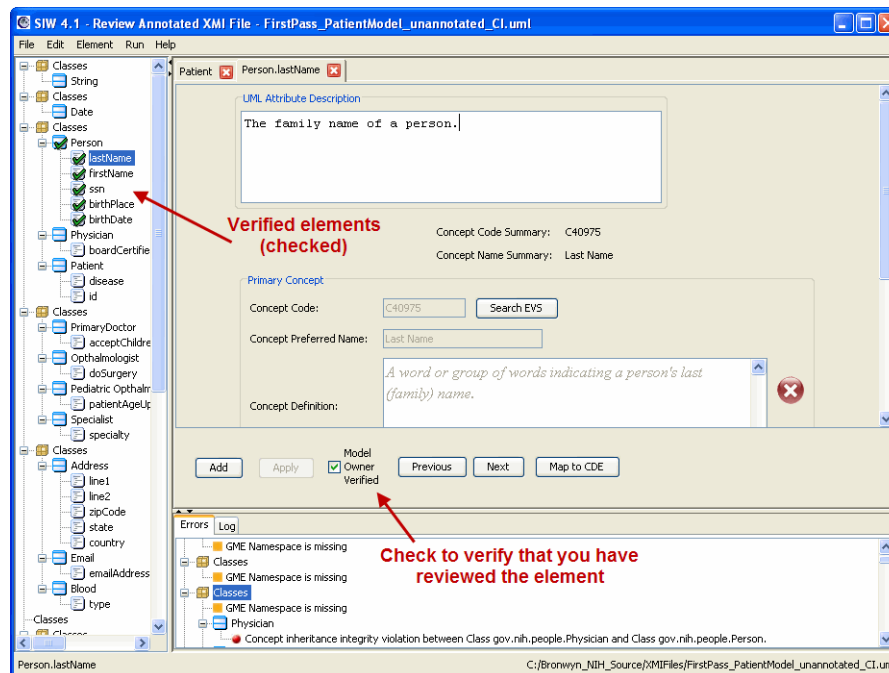


Figure 8.15 SIW Viewer in Review Annotated XMI File mode, with model owner verified elements

Once you are satisfied with the annotations for all of the elements in the file, you must mark all elements as **Model Owner Verified**. This includes the elements that you mapped to CDEs and those you simply verified the concept mapping for.

To verify an element:

1. Select an element from the navigation tree.
2. Review the information in the detail view to verify that the element is mapped to your satisfaction (either to one or more concepts or to an OC or CDE).
3. Click the **Model Owner Verified** checkbox to enable it. One of the following occurs, depending on the element you are verifying:
 - If you have selected a class element and there are un-verified attributes associated with that class, the next item in the tree is automatically selected for you to verify.
 - If you have selected an attribute, a checkmark appears on the attribute in the navigation tree and the next item in the tree is automatically selected.
 - If you have selected the last attribute in the class to be verified (all others have already been verified), and the class has already been verified, a checkmark appears on the attribute you are verifying and on the class element, and the next item in the tree is automatically selected.

The automatic movement to the next element during element verification is simply a navigational aid, so that you do not have to continually click **Next** to move to the next item. Once an element is verified, the SIW assumes that you are finished with that item.

Once you have reviewed all items in the file, and cleared all of the errors listed in the Errors tab (which may require multiple launches of the SIW and the XMI file to update the Errors tab), you must save the file using the **File > Save As** command. Reviewed and verified files should be saved with the term “approved” or “fixed” appended to the front of the filename, depending on whether you are returning the file to the EVS Curation team for further curation, or you believe the file to be ready for loading. For example, if the original file name was `Annotated_FirstPass_myModel.xmi`, then you might save the reviewed file as

`Approved_Annotated_FirstPass_myModel.xmi` or

`Fixed_rv1_Annotated_FirstPass_myModel.xmi`. If the original file had a `.uml` extension (exported from ArgoUML) then the reviewed file must also be saved with a `.uml` extension.

For further instructions on saving a file, see [Saving Changes to a File](#) on page 48.

An approved annotated XMI file is used as input to the UML Loader by CBIIT staff. You can use the approved annotated XMI file as the input to the next version of your model, or to update the current version of the model with the SIW-generated annotations. See [Updating UML Model Definitions with SIW-Generated Changes](#) on page 49 for more information.

CHAPTER 9

USING GME ANNOTATIONS

Because your model does not have to contain GME tagged values in order to be considered semantically annotated, GME annotations are treated separately from the other features of the SIW.

This chapter provides information on the purpose for using GME annotations in your model, how to use the SIW to generate those annotations, along with a few scenarios that may help you determine when to apply which step of the SIW to your model. This applies specifically to using the Roundtrip step along with the Generate Default GME tags step, as both of those steps can provide GME tagged values for your model.

Regardless, the information in this chapter should provide you with enough information to let you use the SIW to generate the appropriate GME tagged values for the elements in your model, along with the appropriate semantic annotations.

Topics in this chapter include:

- [*Overview of GME Annotations*](#) on page 109
- [*Sample Scenarios for Using GME Options*](#) on page 111
- [*Generating Default GME Tags*](#) on page 113
- [*Clearing GME Tags*](#) on page 117

Overview of GME Annotations

Before going too far, you need to understand a little about two terms: XSD and GME tagged values.

What is XSD? XSD stands for **X**ML **S**chema **D**efinition. XSD is an XML language used to express a set of rules to which XML documents must conform in order to be considered “valid”. So when someone refers to an XSD, they mean an XML schema document that describes or defines a specific schema.

What are GME tagged values? GME tagged values are intended to capture the mapping between the UML model elements and the XML schemas that define the XML that will be used to exchange data represented by your model.

XSDs are important for caGrid. A caBIG UML model indexed in caGrid is known as a “Grid Data Service”. Grid services exchange data using XML. The format of the XML exchanged is described in an XSD document. The XSD document is stored in a caGrid component called GME (Global Metadata Exchange). caGrid allows data services to be created from any model registered in caDSR. Registering GME metadata (along with the other metadata used by your model) allows for the automatic retrieval of XML schema definitions.

So if you are loading your model to caGrid, having the proper GME annotations in your model is important. The GME information resides in your model as tagged values. This guide uses the term “GME annotation” to refer to both the GME tagged values as well as the process of including them in your model.

As stated at the beginning of this chapter, inclusion of the GME annotations are not required for semantic integration of your model. The SIW will run just fine with or without the GME tags, and the UML Loader will still register models that do not have GME tags. In the caDSR, GME tags are registered as Alternate Names in your model, so to caDSR, it is just one more alternate name being attached to various administered components of your model. The importance of having the GME annotations in caDSR is so that caGrid can access this information directly from caDSR rather than having to go through the step of finding the appropriate schema.

GME Namespaces are represented as URIs. If you use the SIW to create *default* GME tags, the URI given is automatically derived by the SIW. The important thing about this URI is that it gives the caGrid enough information to look up the relevant portion of the identified XSD for any component in a model. The GME namespace URI is saved in caDSR as an “Alternate Name” for the model element. The alternate name added into caDSR for the element will have a type that specifies that it is a GME namespace tag.

This means that when the UML Loader loads the model to caDSR, the following alternate names are added for the model components containing GME tags:

- GME_XMLElement
 - 1 per Object Class
- GME_XMLLocReference
 - 1 per CDE
- GME_XMLNamespace
 - 1 per Project
 - 1 per Package
 - 1 per Object Class
 - 1 per CDE
- GME_SourceXMLLocRef
 - 1 per association

- GME_TargetXMLLocRef
 - 0 or 1 per association

When you view CDE information in the CDE Browser, you can see these alternate names and their types in the Alternate Name information section of the Data Element Details tab. These GME Alternate Names appear as URLs, as described above.

Since we support the addition of GME annotations to models and to the information contained in caDSR for a model, there must be some way to get this information into caDSR via the UML Model Loader. The way this is accomplished is through the use of tags and their corresponding tagged values in the XMI file.

The GME tagged values can be added either through the SIW (as described in this chapter) or using caAdapter.

The difference between using SIW or caAdapter for generating the GME tagged values is as follows: use caAdapter if you want to specify the URL of the XSD yourself; use SIW if you want default URLs generated for you.

To see the names of the GME tags and to learn a little more about the details of these values, refer to the XMI Tag Reference wiki page: <https://wiki.nci.nih.gov/x/SYI8>.

Additional information regarding the use and loading of GME tagged values is also available on the GME Design wiki page: <https://wiki.nci.nih.gov/x/klAl>.

Sample Scenarios for Using GME Options

There are many different possible scenarios for what you may need to accomplish with respect to adding or editing the GME annotations in your UML model. Which situation you are in will determine the order in which you perform the steps of the SIW.

There are some times when you want to reuse your existing GMEs and some times when you want to create new ones. There may be even be times when you want to do both in the same model. Without any specifics, the basic order of operations would be:

1. Clear the GME tags.
2. Run the Roundtrip step, including GMEs (optional and only if you are already running Roundtrip against an external project).
3. Run Default GME creation.

The other steps of the SIW do not particularly matter from a GME viewpoint because those modes add semantics, not GME information.

Since, however, there is no way to point out every possible scenario, the bottom line is that the more you understand about GME and how it works, the better decisions you will make about how to integrate GME tags into your model and how to use the SIW (or caAdapter) to do so.

That being said, the following sections provide generalized scenarios to address what the overall SIW process would be for GME annotation along with semantic integration of your model.

Scenario One: Annotating a New Model With New Classes

In this scenario you are working with a new model that contains classes that are also new and are not used in any other models. Since this is a brand new model, no GME tags exist yet.

Since there are no classes coming from other models, you do not need to run the Roundtrip step, so the GME tags will not be generated that way. To generate the GME tags for your model in this case, all you must do is run the Generate Default GME Tags step. This can be done at any time in the process.

Alternately, you may be building a new model but using an existing (but also not registered) XSD. However since it is a new model, it has never been registered in caDSR. In this case you would use caAdapter to map the classes to the URIs.

Scenario Two: Annotating a New Model With Reused Classes

In this scenario, you are working with a new model but your model contains classes from other loaded models. Since this is a brand new model, no GME tags exist yet.

Since, however, there are classes in the model from other loaded models, you do need to run the Roundtrip step. This gives you the following choices:

- If you also want to reuse the same XSD for the classes you are reusing, you must **uncheck** the **Exclude Namespaces from Roundtrip** checkbox (an option in the SIW Roundtrip mode wizard). This tells the SIW to add GME Namespace tags to the classes in your model that identify the XSD for those already-loaded classes you are reusing.
- If you do not want to reuse the XSD for the classes you are reusing, you should run the Generate Default GME tags step first, and then run the Roundtrip step, being sure to **check** the **Exclude Namespaces from Roundtrip** checkbox. Doing things in this order provides the GME tags your model needs and then provides the information necessary from Roundtrip for the already-loaded classes you are reusing. Remember that the SIW will not overwrite existing GME tags, so you should generate your default tags first.

Using the Roundtrip step to enter GME tags for the reused classes adds GME tags to the packages, classes, and attributes that are found in the existing model. The tags added in this way identify the URL of the existing external XSD for the existing model, but only for the items that are reused in your model. All other components remain without GME annotations.

Regardless of whether you want to reuse the external XSD or not, you will want to run the Generate Default GME Tags step, to either add default GME tags to your entire model or just to the components that were not tagged as a function of the Roundtrip step.

Scenario Three: Annotating a New Version of an Existing Model

In this scenario you are working with a new version of a model for which a previous version was annotated and loaded to caDSR. This means that GME tags exist for all of the model components that have not changed since the last version, but you have a new model that may also contain new components.

In this case you will want to update the GME tags because part of the tag URL includes the model version number. The old GME URIs contain the old version number and will therefore be out of date.

The first step will be to run the GME Cleanup step, to remove all of the old GME annotations (those with the old version number).

The second step will be to run the Roundtrip step, being sure to **check the Exclude Namespaces from Roundtrip**, in order to reuse the CDEs that exist for the components that have not changed since the last version, but *without* applying the old version's GME tags.

Finally, you can then use the Generate Default GME Tags step to generate updated GME annotations with URIs that contain the new model version number. If the previous version used namespaces other than the default ones, you can also use caAdapter to add namespaces to your updated model.

Scenario Four: Making Updates to an Annotated Model

In this scenario you are working with a model that has already been annotated, but you have had to make changes to the model for some reason. Perhaps there was a problem loading the model the caDSR Sandbox, or there was a problem that arose during a Silver-Level Compatibility review.

Whatever the reason, the model already contains the GME tags from the earlier annotations steps. If you are making just a few changes, the easiest path is to manually delete the GME tags from the packages, classes, and/or attributes that you are changing. This can be done directly in EA or ArgoUML by deleting the actual GME tag and its value as part of making the other necessary changes to your model.

Then, after you export the new XMI file, you can open the SIW and run the Generate Default GME Tags step. This process will generate new GME tags only for those components whose tags you removed.

The bottom line is that the more you understand what the GME is, how it is used by caGrid, and how the URIs are created to point to the correct XSD, the easier it will be for you to determine when to create new GME tags versus reusing existing GME annotations.

Generating Default GME Tags

The SIW provides you with the ability to automatically add GME Namespace tagged values to your model. Once generated, these GME tags accurately reflect the schema used for your Grid service. In addition, the UML Model Loader recognizes these GME tags and records those tagged values as alternate names in caDSR.

When you perform this step, you can provide a project name, version, and context name for generating the GME tags. Default tags are only generated if they are missing from the input file, except for the root level GME_XMLNamespace tag, which is always regenerated.

When the Generate Default GME Tags step is finished, the SIW automatically creates a new XMI file with the tags and saves it as a new file to the same directory where the

source file resides. The model owner should review this file before submitting it through the curation process or for loading to caDSR via the UML Model Loader.

When you should run this step is to some degree determined by whether you are working with a brand new model with brand new components, or you are reusing components from another model, or you are updating a new version of an existing model. See the above section, [Sample Scenarios for Using GME Options](#) on page 111 to determine whether to run this step instead of or in conjunction with the Roundtrip step of the SIW.

Input To Step: The UML model in XMI format, exported from either EA (saved with an .xmi file extension) or ArgoUML (saved with a .uml file extension).

Example: `Annotated_myModel.xmi` for an EA export;
`Annotated_myModel.uml` for an ArgoUML export; of an annotated processed file.

Output From Step: The XMI file containing GME tags is saved with the term “GMEDefault_” appended to the front of the file name (e.g., `GMEDefault_Annotated_myModel.xmi`. or `GMEDefault_GMECleanup_Annotated_myModel.xmi`).

After performing this step, you may want to review the GME tags that have been added to the file using the **Review Annotated XMI File** option. See [Chapter 8, Reviewing An Annotated File](#), on page 91.

In the SIW Viewer, you can review the GME tags for:

- Package (XMLNamespace)
- Classes (XMLElement)
- Attributes (XMLLocReference)
- Associations (SourceXMLRefLoc and TargetXMLRefLoc)

For each of these, the SIW provides a read-only text field allowing you to review this information, as shown in the figure below. If you do not see these tags in the viewer window, check your viewer preferences to be sure the Show GME tags option is checked. See [Setting Viewer Preferences](#) on page 44 for more information.

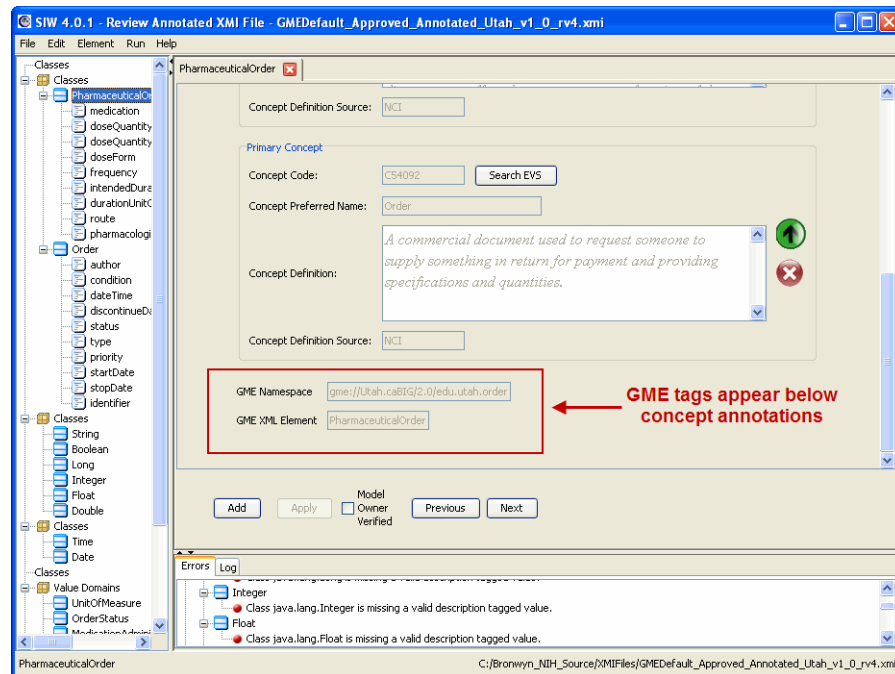


Figure 9.1 SIW Viewer showing GME tags added to the model

If you want to modify the values, you must do so using either the modeling tool or caAdapter.

To generate default GME tags:

1. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.
2. From the SIW Welcome screen, select option **6. Generate Default GME Tags (Model Owner)** and click **Next**.
3. In the Select file to parse window, identify the file you want to process. You have the following options:
 - If the file you want to review appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename.
 - **Type** the full path and filename into the text box.

- ° Click **Browse** to browse for and select the file exported from EA or ArgoUML.

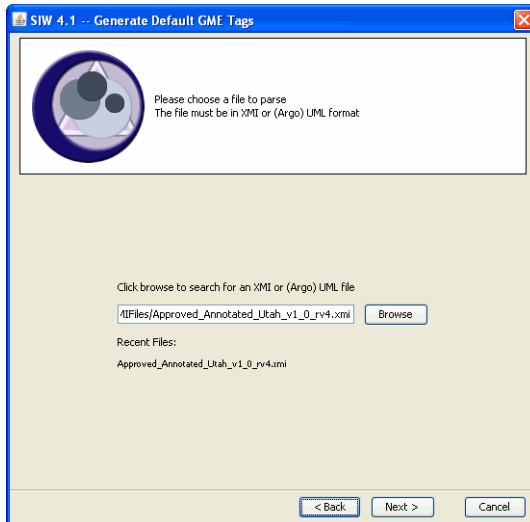


Figure 9.2 Select the file to generate GME tags into

4. Click **Next**.
5. Select the **Project Name**, **Project Version**, and **Context** that correspond to the GME namespace tags you want to generate for the items in your model. These should be the same as those you specify in your submission document, as this is the context and project name that will appear in the Browser tree for your model.
6. Use the **Select Package** drop-down list to identify the package in your model for which you want to generate these tags. If you want to tag your entire model, select **Logical Model** from the list.

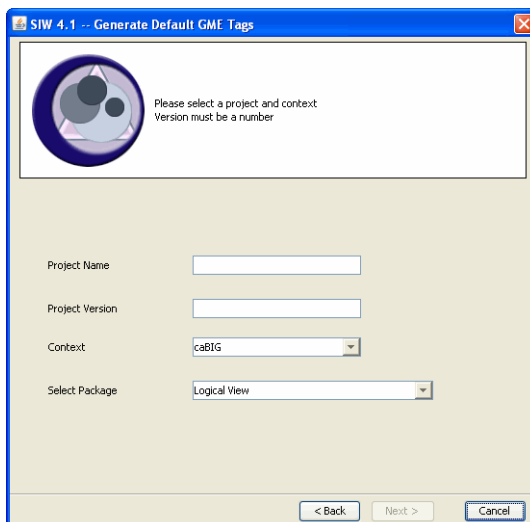


Figure 9.3 Select the Project, Version and Context for the GME tags.

7. Click **Next**.

The SIW processes the XMI file using the project and version values you entered, and generates GME tags for the items in the model.

If the process is successful, the SIW creates GME tags for the items in your model and inserts those tags into the XMI file. When finished, a final screen appears confirming completion of the process and identifying the location of the newly generated file. Note the location and click **Finish**.

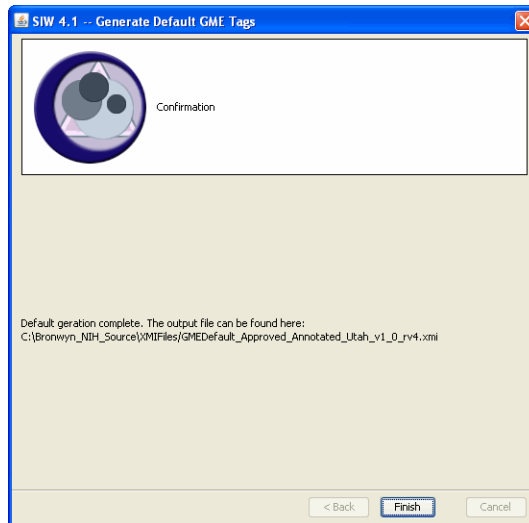


Figure 9.4 GME tagging process complete - displays location of new XMI file

This new file is saved into the same directory as the source file and is appended with the term “GMEDefault” on the front of the filename. For example, if the processed file’s path/name was `c:\myXMIfiles\myModel.xmi` the newly saved file will be `c:\myXMIfiles\GMEDefault_myModel.xmi`. If your source was exported from ArgoUML, the extension on the filename would continue to be `.uml`.

After performing this step, the model owner should review the new XMI file using the **Review Annotated XMI File** option. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91.

Be advised that you may have to adjust your SIW Viewer preferences in order to see your GME tags in the viewer. See [Setting Viewer Preferences](#) on page 44 and [Show GME Tags](#) on page 48 for more information.

If there are GME tags missing from the file, the SIW generates a warning and lists it with the item in the Errors tab. If a GME tag is invalid, the SIW generates an error for the tag. Errors must be fixed before a file can be considered fully annotated and submitted for loading.

Clearing GME Tags

The SIW provides you with the ability to automatically remove GME tagged values from your model. This automated process is useful if you need to replace existing GME tags in your model. Before generating the new tags, you must remove the old ones.

When the Cleanup GME step is finished, the SIW automatically creates a new XMI file without the GME tags and saves it as a new file to the same directory where the source file resides.

When (during the SIW process) you should run this step is to some degree determined by whether you are working with a brand new model with brand new components, or you are reusing components from another model, or you are updating a new version of an existing model. See the above section, [Sample Scenarios for Using GME Options](#) on page 111 to determine when to run this step in conjunction with the Roundtrip and the Create Default GME Tags steps of the SIW.

Input To Step: The UML model in XMI format, exported from either EA (saved with an .xmi file extension) or ArgoUML (saved with a .uml file extension).

Example: `Annotated_myModel.xmi` for an EA export;
`Annotated_myModel.uml` for an ArgoUML export; of an annotated processed file.

Output From Step: The XMI file whose GME tags have been removed is saved with the term “GMECleanup_” appended to the front of the file name (e.g., `GMECleanup_Annotated_myModel.xmi`. or `GMECleanup_Annotated_FirstPass_myModel.xmi`).

After performing this step, the model owner should review the XMI file using the **Review Annotated XMI File** option, to see that the tags have been removed properly. See [Chapter 8, Reviewing An Annotated File](#), on page 91.

To clear GME tags:

1. Open the SIW URL: <http://cadsrsiw.nci.nih.gov>.
2. From the SIW Welcome screen, select option **7. GME Cleanup (Model Owner)** and click **Next**.
3. In the Select file to parse window, identify the file you want to process. You have the following options:
 - If the file you want appears in the **Recent Files** list, click on the file name to populate the filename field with the path/filename.
 - **Type** the full path and filename into the text box.

- Click **Browse** to browse for and select the file exported from EA or ArgoUML.

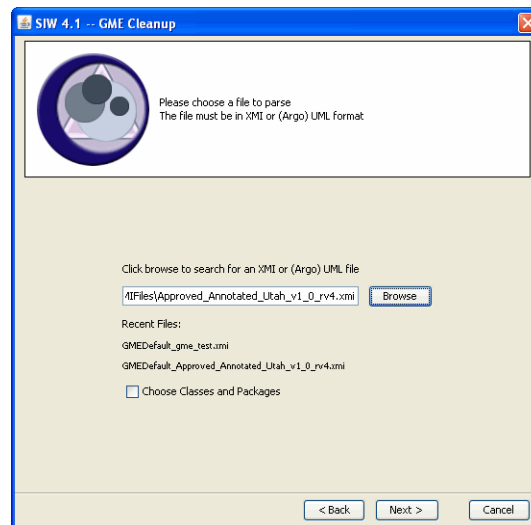


Figure 9.5 Select the file to generate GME tags into

- Click **Next**.

The SIW processes the XMI file and removes the GME tags from all items in the model.

If the process is successful, the SIW removes the GME tags from the XMI file. When finished, a final screen appears confirming completion of the process and identifying the location of the newly generated file. Note the location and click **Finish**.

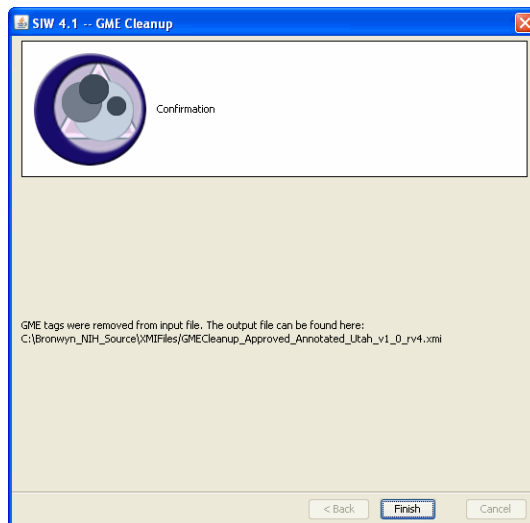


Figure 9.6 GME tag removal process complete - displays location of new XMI file

This new file is saved into the same directory as the source file and is appended with the term “GMECleanup_” on the front of the filename. For example, if the processed file’s path/name was `c:\myXMIfiles\Annotated_myModel.xmi` the newly saved file will be `c:\myXMIfiles\GMECleanup_Annotated_myModel.xmi`. If your source was exported from ArgoUML, the extension on the filename would continue to be `.uml`.

After performing this step, the model owner can review the new XMI file using the **Review Annotated XMI File** option to be sure the tags have been removed. For more information, see [Chapter 8, Reviewing An Annotated File](#), on page 91.

Be advised that you may have to adjust your SIW Viewer preferences in order to see your GME tags in the viewer. See [Setting Viewer Preferences](#) on page 44 and [Show GME Tags](#) on page 48 for more information.

When there are GME tags missing from the file, the SIW generates a warning and lists it with the item in the Errors tab. If appropriate, you can now run the file through the Generate Default GME Tags step to generate new tags for the file. See [Generating Default GME Tags](#) on page 113.

UNDERSTANDING THE UML MODEL LOADER

The UML Model Loader is a Java application that transforms UML domain models into caDSR metadata, reusing existing caDSR administered components or creating new ones as needed. This process is also referred to as registering the UML model in caDSR.

This chapter describes how the UML Loader transforms the UML model data from an annotated XMI file into caDSR metadata while also mapping the registered metadata back to the UML model that uses it.

Metadata registration into the caDSR is performed by a team at CBIIT, working with model owners to be sure that the information from their model is registered properly. The information in this chapter is designed to help model owners understand how their annotation work translates into caDSR content so that they can accurately predict the outcome of the loading of their model into caDSR.

Beyond the basic mapping of UML model elements to caDSR registered objects, this chapter also contains basic information about how association and inheritance relationships translate into caDSR content.

Topics in this chapter include:

- *Overview of UML Loader Process* on page 122
- *Relating (Classifying) Metadata Items to UML Models* on page 124
- *Using Pre-mapped CDEs (Common Data Elements)* on page 131
- *Creating and Updating Concepts in caDSR* on page 131
- *Mapping a UML Class to an Object Class* on page 132
- *Mapping a UML Attribute to a Property* on page 133

- [Mapping a UML Class to a Value Domain](#) on page 134
- [Mapping Data Element Concepts](#) on page 138
- [Mapping Data Elements](#) on page 139
- [Mapping UML Associations to Object Class Relationships](#) on page 140
- [Mapping UML Inheritance](#) on page 141
- [Reviewing and Verifying Registered Metadata](#) on page 143

Note: Before continuing, you are encouraged to review [Terms You Should Know](#) on page 16 regarding the definitions of data elements and data element concepts, how they are formed, and how UML model elements relate to caDSR components.

Overview of UML Loader Process

Once the UML object model is annotated with EVS concepts and the annotated version of the model has been approved, the model owner sends the annotated XMI file to the CBIIT caDSR team for processing. The EVS concept annotations contained in the file are the basis for determining whether each of the UML elements can be represented using existing caDSR components or if new ones must be created.

These EVS concept annotations function as the semantic bridge between the elements in the class model and the components in caDSR. This is why the UML Loader will not load XMI files that do not contain the mandatory EVS concept tags for all UML classes and attributes.

UML model element to caDSR component mapping is summarized in [Table 10.1](#). Specifically, the UML Loader transforms UML model attributes and classes, including inheritance and association links, into caDSR metadata. UML class operations and other types of UML elements are *not* transformed. In addition, the UML Loader does not transform UML data models.

UML Model Element	caDSR Administered Component	Additional Information
Mapped EVS Concept (concept tagged value)	caDSR Concept (Concept Class). caDSR maintains its own concept data, corresponding to the information in EVS/NCI Thesaurus	The UML model concept information is correlated to caDSR concept information. If the model contains a concept not resident in caDSR, a new caDSR Concept is created for use.
Class	Object Class (OC)	Using the concept codes mapped to the class element, an existing caDSR OC is used or a new one is created.
Attribute	Property	Using the concept codes mapped to the class element, an existing caDSR Property is used or a new one is created.

Table 10.1 Transformation Mappings from UML Model Elements to caDSR Components

UML Model Element	caDSR Administered Component	Additional Information
Datatype (Attribute Datatype)	Value Domain	<p>This is the datatype associated with the attribute in the model. Typically the datatypes used are common Java datatypes and as such already exist in caDSR. However, some models define their own local datatypes (as classes in the model) and associate those with the attributes in the model using tagged values.</p> <p>Locally defined datatypes are referred to as “local value domains” or “LVD.”</p>
Class-Attribute combination	<p>Data Element Concept (DEC)</p> <p>A DEC is the combination of an OC and a Property.</p>	<p>After the OC and the Property are mapped (to the class and attribute), they are combined to form a DEC. If the combination already exists in caDSR the existing DEC is used. If the combination is new, a new DEC is created.</p> <p>One DEC is formed for every class-attribute combination in the model. So if the model has a class element “Gene” that has four attributes, four DEC’s are formed.</p>
Class-Attribute-Datatype combination (The datatype is the datatype identified for the attribute.)	<p>Data Element (DE) or Common Data Element (CDE).*</p> <p>A DE or CDE is the combination of a DEC with a value domain.</p> <p>*The terms DE and CDE are used interchangeably.</p>	<p>After the DEC is formed, the value domain is determined and added to the DEC to form the DE.</p> <p>If the combination already exists in caDSR the existing DE is used. If the combination is new, a new DE is created.</p> <p>One DE is formed for every class-attribute-datatype combination in the model.</p> <p>So if the model has a class element “Gene” with four attributes, there will be four DEC’s formed. Each of those DEC’s will be combined with the value domain corresponding to the datatype for each attribute, forming four DE’s.</p>

Table 10.1 Transformation Mappings from UML Model Elements to caDSR Components

So for every element in the UML model, a corresponding metadata component (also referred to as an administered component) in caDSR is either mapped or created (and then mapped). The “mapping” of caDSR components involves adding information about the project and the model to the component information in caDSR. This process of relating caDSR metadata to the models that use them is described in more detail in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

Submitting a UML Model to caDSR

When you have completed the semantic integration process and are ready to submit your UML model (via fully annotated and approved XMI file) to CBIIT for registering, contact the CBIIT help desk at: ncicb@pop.nci.nih.gov.

You will either be provided with the submission form and instructions, or pointed to the location of the form on GForge: http://gforge.nci.nih.gov/docman/index.php?group_id=64&selected_doc_group_id=184&language_id=1

You may want to follow the link to the submission form before you are ready to submit the file, in order to review the pre-requisite instructions included with the form.

Your model will initially be loaded to the caDSR Sandbox environment, allowing you to review the loading of your model and work with the caDSR curators to make any small changes needed. For more information on reviewing your model's elements after they are loaded, see *Reviewing and Verifying Registered Metadata* on page 143.

Relating (Classifying) Metadata Items to UML Models

Whenever caDSR metadata items are created or reused for a UML model, certain information about the model is appended to the item in the caDSR. This is done for two reasons:

1. It allows the information to be cross-referenced by Project Name and/or model information (class or attribute name);
2. It provides detailed information about how the Project/Model uses that particular caDSR component.

The relationship mapping is done by “Classifying” each component and by assigning an “Alternate Name” and “Alternate Definition” to each component.

The following table identifies the relationship between the information added to each component in caDSR and the UML model item that provides that information. Keep in mind that the more common caDSR components (“gene name” or “patient”) is likely to have several of each of these associated with it, because each time a model uses the component, the relationship information is added to the component in caDSR.

caDSR Component Information	UML Model Information	Used to...
Classification Scheme (CS) Identifies the project in caDSR.	Project Name Provided by the model owner in the form submitted with the model to NCICB.	All caDSR components derived from UML models are “classified” during model load. Classification identifies which projects use that caDSR component in their model. Classification also provides a method of categorization for caDSR items. You can search caDSR for a particular component or you can look for all components in a CS.

Table 10.2 Relationship between caDSR component information and UML model information.

caDSR Component Information	UML Model Information	Used to...
Classification Scheme Item (CSI) Identifies the sub-project, project alias or package name in caDSR	Sub-Project, Project Alias or Package Name Provided by the model owner or detected by the UML Loader (packages only).	The CSI is used as an additional classification or identification mechanism for when projects are divided into sub-projects or packages. Also like CS, CSIs provide an additional level of categorization for caDSR components.
Object Class Alternate Name	Class Name	Provides the exact name of the class element as used in the project's registered model, which has been mapped to this particular caDSR OC. The CS identifies the project.
Object Class Alternate Definition	Class description - Text is pulled from the description tagged value for the class element in the model.	Relates the identified project's intended use or definition of the class mapped to this particular caDSR OC. The CS identifies the project.
Data Element Alternate Name	Two are created: Class Name & Attribute Name (connected with a colon); Package Name & Attribute Name (referred to as a "fully qualified attribute.")	One provides the project's specific name for the class/attribute combination used to form this particular caDSR DE; the other provides the fully qualified attribute information for the attribute used to form the DE. The CS identifies the project; the fully qualified attribute name identifies the package or sub-project of the model containing the attribute.
Data Element Alternate Definition	Attribute description - Text is pulled from the description tagged value for the attribute element in the model.	Relates the project's intended use or definition for the attribute used to form this particular caDSR DE. The CS identifies the project.

Table 10.2 Relationship between caDSR component information and UML model information.

Note: Alternate Names and Alternate Definitions are also created for Property and Data Element Concept components, however these are not visible through the CDE Browser and they reiterate the Alternate Name/Definition information for Object Classes and Data Elements. They can be viewed through the CDE Admin Tool if necessary.

The sections that follow provide more detailed explanations for how and why caDSR metadata is classified, how and why alternate names and definitions are added to caDSR elements, and where you can view that information.

Classifying UML Model Metadata (Classification Schemes and Classification Scheme Items)

A *Classification* is a grouping of items related to a specific project or model. A *Classification Scheme* identifies the project and version associated with the registered model. A *Classification Scheme Item* identifies any sub-projects, packages, or aliases included in the project (since UML class models are commonly organized into packages). The UML Loader uses one classification scheme (CS) and at least one classification scheme item (CSI) for each UML domain model. Unless otherwise specified, all CS are created in the caBIG context.

The CS and CSI(s) for the domain model act as both a cross-referencing mechanism for viewing only a specific model and as an identifier for which projects are using what caDSR components in their models. Furthermore, classification schemes are used to identify whether or not that grouping of components is ready for general access through the CDE Browser. A CS will not appear in the CDE Browser until the workflow status is set to “Released.” This allows model owners and curators to review and edit their caDSR content before allowing full public visibility.

The CS for the project is created using the information provided on the form submitted with the XMI file to CBIIT. This information includes Project Abbreviated Name, Project Long Name, Project Definition, and Version.

The CSI(s) for the project are created in one of two ways:

1. The UML Loader can be configured to create a CSI corresponding to each package in the UML model, using the package names from the XMI file and associating each CSI with the CS.
2. The UML Loader can be configured to ignore the separate packages in the UML model and instead have only one CSI specified for the entire model. The UML Loader then creates only one CSI associated with the CS.

When viewing caDSR components through the CDE Browser, the classification information can be viewed in two ways: in the tree view of the main CDE Browser page, and in the Classifications tab for a selected data element.

The CDE Browser tree view lists all Contexts that contain caDSR metadata. Expanding a Context (click the + sign to the left of the folder) displays the components of the

context, including a node for classifications. Expanding the Classifications node provides a list of all classifications that have been loaded to that context.

The screenshot shows the CDE Browser interface. On the left is a tree-view of contexts. The 'caBIG Context' is highlighted. Below it, the 'Classifications' node is expanded, showing a list of classification schemes and their associated items. Red arrows point to the 'caBIG Context' and the 'Classification Scheme (CS) and associated Classification Scheme Item (CSI)'.

Long Name	Preferred Question Text	Owned By	Used By Context	Registration Status	Workflow Status	Public ID	Version
Engineered Gene Name	java.lang.String	caCORE	caBIG	Qualified	RELEASED	2224376	3.0
Gene Alias Name	java.lang.String	caCORE	caBIG	Qualified	RELEASED	2223776	3.0
Gene Expression Reporter Name	java.lang.String	caBIG		Qualified	RELEASED	2529113	1.0
Gene Ontology Name	java.lang.String	caCORE	caBIG	Qualified	RELEASED	2529575	1.0
Gene Ontology Term Name	java.lang.String	caBIG		Qualified	RELEASED	2433417	1.0
Gene Ontology Term Name	java.lang.String	caBIG		Qualified	RELEASED	2557994	1.0
Gene Product Name		caBIG		Qualified	RELEASED	2638667	1.0

Figure 10.1 CDE Browser tree-view with node items labeled

Selecting a CS or CSI in the tree-view allows you to search or browse for only the metadata elements used by that project/sub-project.

If you select to view a data element, one of the available tabs is called Classifications and lists all of the CS and CSI that are mapped to that data element.

The screenshot shows the CDE Browser interface with the 'Classifications Tab' selected. The 'Selected Data Element' section displays the following information:

Public ID:	2529575
Version:	1.0
Long Name:	Gene Ontology Name java.lang.String
Short Name:	2529521v1.0 2178533v1.0
Preferred Question Text:	
Definition:	The Gene Ontology (GO) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases. The goal of the Gene Ontology project is to produce a controlled vocabulary that can be applied to all organisms even as knowledge of gene and protein roles in cells is accumulating and changing. GO provides three structured networks of defined terms, molecular function, biological process, and cellular component, to describe gene product attributes. (from http://www.geneontology.org/). The words or language units by which a thing is known. Generic value domain for a java datatype that is a class that represents character strings.
Workflow Status:	RELEASED

Below the 'Selected Data Element' section is the 'Classifications' tab, which displays a table of Classification Scheme (CS) and Classification Scheme Item (CSI) information.

CS* Long Name	CS* Definition	CS* Public ID/Version	CSI* Name	CSI* Type	CSI* Public ID/Version
caBIO	caCORE 3.1 models	2528431v3.1	caBIO	UML_PACKAGE_ALIAS	2736727v1.0
gov.nih.nci.cabio.domain	caCORE 3.1 models	2528431v3.1	gov.nih.nci.cabio.domain	UML_PACKAGE_NAME	2736156v1.0
caBIO	Version 3.2 of caCORE models	2528431v3.2	caBIO	UML_PACKAGE_ALIAS	2736727v1.0
gov.nih.nci.cabio.domain	Version 3.2 of caCORE models	2528431v3.2	gov.nih.nci.cabio.domain	UML_PACKAGE_NAME	2736156v1.0
gov.nih.nci.cabio.domain	caBIO model	2714661v4.0	gov.nih.nci.cabio.domain	UML_PACKAGE_NAME	2736156v1.0

Figure 10.2 CDE Browser showing the Classification Tab for a Data Element

The classifications listing shows every CSI that uses the selected data element. This means that the CS may be listed several times, once for each CSI it contains that uses that DE.

CS and CSI information is also listed with each Alternate Name and Definition in the main Data Element tab. These are discussed in more detail in the section below.

Alternate Names and Alternate Definitions

Alternate Names and Alternate Definitions are used to provide specific meaning for how or why a particular model used a particular piece of metadata. This is done by providing both the exact name of the element(s) involved (class name and/or attribute name) and the element description resident in the model.

Because this information is specific to a particular project's model, Alternate Names and Definitions are always listed with the CS and CSI for the model.

Alternate Names and Alternate Definitions are created for all caDSR components, however the CDE Browser only displays this information for Object Classes and for data elements. The caDSR Admin Tool shows all associated alternate information.

Creating Alternate Names and Alternate Definitions for Object Classes

Each time a UML model class element is mapped to a caDSR OC, two Alternate Names are created for that OC:

- A **UML Class** type Alternate Name: Provides the exact class name of the element in the model. The project is identified by the CS/CSI listed with the alternate name. The **Type** column shows **UML Class**.
- A **UML Qualified Class** type Alternate Name: Provides the exact package name and class name combination for the class element. The project is identified by the CS/CSI listed with the alternate name. The **Type** column shows **UML Qualified Class**. For example, the class name "Gene" in the domain "gov.nih.nci.cabio.domain" would become an Alternate Name of **gov.nih.nci.cabio.domain.Gene** with a Type of **UML Qualified Class**.

Alternate Names

Name	Type	Context
edu.wustl.geneconnect.domain.Gene	UML Qualified Class	caBIG
Classifications		
CS* Long Name	CS* Definition	CS* Public ID
GeneConnect	GeneConnect is the mapping service that will facilitate interoperability by interlinking VCDE approved genomic identifiers. To interlink all of these identifiers, database annotations will be used to construct pair wise connections, and then all-to-all relationships will be calculated by traversing all possible combinations of edges in the graph using every node as the starting point. Next, using reciprocity calculations performed over the data sources, the set of non-commutable results will be determined with each record given a confidence score based on the ratio of commutable results to all possible combinations for set of the genomic identifiers under question.	2544880
		1.0
		edu.wustl.geneconnect.domain
		UML_PACKAGE_NAME

Name	Type	Context
Gene	UML Class	caCORE
Classifications		
CS* Long Name	CS* Definition	CS* Public ID
Genomic Identifiers	The lack of a common genomic identifier in biomedical databases results in an interoperability problem in the caBIGTM environment. The question of how to deal with the problem of multiple identifiers (e.g. RefSeq ID, GenBank ID, Entrez Gene ID, Ensembl ID, UniProt ID) all pointing to the same object needed to be addressed at the metadata level. The purpose of this UML model is to create CDEs for the most common Genomic Identifiers for use by different projects within the ICR workspace.	
		1.0
		edu.upenn.med
		UML_PACKAGE_NAME
caFE Server 1.1	Function Express Server is a tool that will annotate probes on microarrays using publicly available biomedical databases and will automatically update these annotations on a regular basis. This data can be viewed using a web-based query interface or may be accessed programmatically using the provided Application Programmers Interface (API). Function Express Server can be used to create and populate a database that will store interrelated gene annotation data. The web pages allow the user to display a gene literature network in a tabular format, to search for genes on various annotations like Entrez Gene, UniGene, Gene Ontology term, etc.	2514015
		1.1
		edu.wustl.fe
		UML_PACKAGE_NAME
caArray	Version 2.0 caArray Model	2726654
		2.0
		gov.nih.nci.caarray.domain.array
		UML_PACKAGE_NAME
caCORE 3.0	caCORE Description	3.0
		Cancer Bioinformatics Infrastructure Objects (caBIO)
		UML_PACKAGE_ALIAS

Figure 10.3 CDE Browser - Object Class View showing Alternate Names for the OC.

For common classes, such as Gene, many different projects are likely to have used the same class name for the item. In this case, the UML Class type Alternate Name will list all projects, packages, and versions where a class element with that name exists.

A single Alternate Definition is created for each class element mapped to a caDSR OC. This definition is given the type **UML Class** and comes from the element description tagged value(s) associated with the class element in the model. As with all alternate information, the project is identified by the CS/CSI listed with the definition.

Alternate Definitions

Definition		Type	Context		
Contains gene genomic identifiers. ← UML element description for this class element (Gene) contained in this project's model		UML Class	caBIG		
Classifications		CS* Public ID	CS* Version	CSI* Name	CSI* Type
CS* Long Name	CS* Definition				
GeneConnect	GeneConnect is the mapping service that will facilitate interoperability by interlinking VCDE approved genomic identifiers. To interlink all of these identifiers, database annotations will be used to construct pairwise connections, and then all-to-all relationships will be calculated by traversing all possible combinations of edges in the graph using every node as the starting point. Next, using reciprocity calculations performed over the data sources, the set of non-commutable results will be determined with each record given a confidence score based on the ratio of commutable results to all possible combinations for set of the genomic identifiers under question.	2544880	1.0	edu.wustl.geneconnect.domain	UML_PACKAGE_NAME

Definition		Type	Context		
A hereditary unit consisting of a sequence of DNA that occupies a specific location on a chromosome and determines a particular characteristic in an organism. The functional and physical unit of heredity passed from parent to offspring. Genes are pieces ← UML element description for this class element (Gene) contained in this project's model (different CSIs contain the same class)		UML Class	caBIG		
Classifications		CS* Public ID	CS* Version	CSI* Name	CSI* Type
CS* Long Name	CS* Definition				
Genomic Identifiers	The lack of a common genomic identifier in biomedical databases results in an interoperability problem in the caBIGTM environment. The question of how to deal with the problem of multiple identifiers (e.g. RefSeq ID, GenBank ID, Entrez Gene ID, Ensembl ID, UniProt ID) all pointing to the same object needed to be addressed at the metadata level. The purpose of this UML model is to create CDEs for the most common Genomic Identifiers for use by different projects within the ICR workspace.		1.0	edu.upenn.med	UML_PACKAGE_NAME
Genomic Identifiers	The lack of a common genomic identifier in biomedical databases results in an interoperability problem in the caBIGTM environment. The question of how to deal with the problem of multiple identifiers (e.g. RefSeq ID, GenBank ID, Entrez Gene ID, Ensembl ID, UniProt ID) all pointing to the same object needed to be addressed at the metadata level. The purpose of this UML model is to create CDEs for the most common Genomic Identifiers for use by different projects within the ICR workspace.		1.0	UPenn	UML_PACKAGE_ALIAS

Figure 10.4 CDE Browser - Object Class View showing Alternate Definitions for the OC.

Many Alternate Definitions will have only one CS/CSI listed, while others will have several. If the class is used either by multiple versions of a project and/or by multiple packages within a model, each of those uses is listed because the class element is defined the same way throughout the model; all of the CS/CSI listed use the exact same element description within the model.

Creating an Alternate Names and Alternate Definitions for Data Elements

Alternate Names and Alternate Definitions for data elements are created using a combination of the class and attribute element details from each model.

Each time UML model elements are mapped to form a caDSR DE, two Alternate Names are created for that DE:

- A **UML Class:UML Attr** type Alternate Name: Provides the exact class/attribute name combination of the mapped elements in the model, separated by a colon. The **Type** column shows **UML Class:UML Attr**.

For example, the class element “Gene” having an attribute “name” would become an Alternate Name of **Gene:name** with a Type of **UML Class:UML Attribute**.

- A **UML Qualified Attr** type Alternate Name: Provides the exact package name, class name and attribute name combination for the mapped elements in the model, separated by periods. The **Type** column shows **UML Qualified Attr**.

For example, the class element “Gene” having an attribute “name” in the domain “gov.nih.nci.cabio.domain” would become an Alternate Name of **gov.nih.nci.cabio.domain.Gene.name** with a Type of **UML Qualified Attr**.

A single Alternate Definition is created for each class/attribute combination mapped to a caDSR DE. This definition is given the type **UML Class:UML Attr** and comes from the element description tagged value(s) associated with the attribute element in the model.

The UML attribute description was chosen as opposed to the combination of the description of the class and attribute because, in practice, when defining an attribute for a class, the notion of the class is generally incorporated in the attribute's description.

For DEs each CS/CSI is listed with the associated alternate information, as opposed to listing the Alternate Names and Alternate Definitions separately as is the case for OCs.

Alternate Names and Definitions

CS* Long Name	CS* Definition	CSI* Name	CSI* Type
caBIO	caBIO model	gov.nih.nci.cabio.domain	UML_PACKAGE_NAME
Alternate Names			
Name	Type	Context	Language
gov.nih.nci.cabio.domain.Gene.fullName	UML Qualified Attr	caCORE	ENGLISH
Gene.fullName	UML Class:UML Attr	caCORE	ENGLISH
Alternate Definitions			
Name	Type	Context	
The title of the gene.	UML Class:UML Attr	caCORE	

CS* Long Name	CS* Definition	CSI* Name	CSI* Type
caFE Server	Function Express Server is a tool that will annotate probes on microarrays using publicly available biomedical databases and will automatically update these annotations on a regular basis. This data can be viewed using a web-based query interface or may be accessed programmatically using the provided Application Programmers Interface (API). Function Express Server can be used to create and populate a database that will store interrelated gene annotation data. The web pages allow the user to display a gene literature network in a tabular format, to search for genes on various annotations like Entrez Gene, UniGene, Gene Ontology term etc.	edu.wustl.fe	UML_PACKAGE_NAME
Alternate Names			
Name	Type	Context	Language
edu.wustl.fe.Gene.name	UML Qualified Attr	caBIG	
Gene.name	UML Class:UML Attr	caBIG	
Alternate Definitions			
Name	Type	Context	
A name for a gene determined by the Entrez Gene project at NCBI.	UML Class:UML Attr	caBIG	

CS* Long Name	CS* Definition	CSI* Name	CSI* Type
caCORE 3.0	caCORE Description	gov.nih.nci.cabio.domain	UML_PACKAGE_NAME
Alternate Names			
Name	Type	Context	Language
gov.nih.nci.cabio.domain.Gene.fullName	UML Qualified Attr	caCORE	ENGLISH
Gene.fullName	UML Class:UML Attr	caCORE	ENGLISH
Alternate Definitions			
Name	Type	Context	
The title of the gene.	UML Class:UML Attr	caCORE	

Figure 10.5 CDE Browser - Data Element view showing Alternate Names and Definitions

Creating Alternate Names and Alternate Definitions for a Property

Because the caDSR Property metadata represents the attribute element in a UML model, the Alternate Names and Alternate Definitions for Properties are also based on the attribute information provided in the model.

A single Alternate Name is created for each attribute element mapped to a caDSR Property. This name is given the type **UML Attribute** and comes from the exact name used for the attribute in the model. A single Alternate Definition is also created for the attribute element mapped to the Property. This definition is given the type **UML Attribute** and comes from the element description tagged value(s) associated with the attribute element in the class model.

While alternate information for caDSR Properties is not viewable in the CDE Browser, it does appear in the caDSR Admin Tool.

Creating Alternate Names and Alternate Definitions for Data Element Concepts

Because caDSR DEC is the class/attribute element combination in a UML model, the Alternate Names and Alternate Definitions for DEC are also based on a combination of the class/attribute information provided in the model.

A single Alternate Name is created for each class/attribute combination element associated with a caDSR DEC. This name is given the type **UML Class:Attribute** and comes from the exact names used for the elements in the model. A single Alternate Definition is also created for the class/attribute combination mapped to the DEC. This definition is also given the type **UML Class:Attribute** and comes from the element description tagged value(s) associated with both the class and attribute elements in the class model.

While alternate information for DEC is not viewable in the CDE Browser, it does appear in the caDSR Admin Tool.

Using Pre-mapped CDEs (Common Data Elements)

The SIW interface provides the ability to map UML model elements to existing caDSR CDEs if appropriate. Specifically, it is the attribute elements in the model that are mapped to CDEs. When this is done, the class element is automatically mapped to the OC for the CDE, and the datatype is mapped to the value domain for the CDE.

Because some of this mapping can be pre-entered, the UML Loader automatically checks each attribute in the model to see if it contains the following tagged values:

- CADSR_DE_ID
- CADSR_DE_VERSION

If both of these tagged values are present, the DE (CDE) and all of its composite parts (the OC, Property, value domain, and DEC) are reused and the caDSR components are classified using the CS/CSI information provided for the project/model.

In addition, if the context that owns the mapped CDE is different than the context to which the model is being loaded, the Alternate Name type shown for the component lists "Used_By" and no Alternate Definition appears. You must view the CDE within the context that owns it to see this information.

Creating and Updating Concepts in caDSR

Because trying to retrieve concept information directly from EVS could make the UML Loader cumbersome and slow, and because new concept information is often introduced well before EVS can be updated, the caDSR maintains its own repository of concept information that corresponds to the NCI Thesaurus information in EVS.

When the UML Loader begins to process a file, it checks to see if each concept corresponding to the specified concept code already exists in caDSR. If the concept code exists but the name or definition is different, the UML Loader first checks EVS to see if the name and definition match what is in the XMI file. If they are, the loader updates the concept in caDSR with the new information.

If the code does not exist, then a new concept is created in caDSR. The data used for creating the new concept is pulled from the tagged values located in the XMI file which

were provided through the annotation process. The specific mapping is shown in [Table 10.3](#), using the example of the concept for Gene.

Concept Attribute	Data	Example
Preferred Name	Derived from ConceptCode tagged value.	C16612
Long Name	Derived from ConceptPreferredName tagged value.	Gene
Preferred Definition	Derived from ConceptDefinition tagged value.	A functional unit of heredity which occupies a specific position (locus) on a particular chromosome, is capable of reproducing itself exactly at each cell division, and directs the formation of a protein or other product.
Version	1.0 (Default)	1.0
Workflow Status	RELEASED (Default)	Released
Context	Specified in the submission form with the model - the context to which the model is to be uploaded.	caBIG
Begin Date	Current Timestamp	01/23/2005

Table 10.3 Concept Attribute details

Definitions that come from sources other than NCI are captured as Alternate Definitions for a concept and are mapped as shown in [Table 10.4](#).

Alternate Definition	Data
Definition	Derived from ConceptDefinition tagged value.
Context	Specified in the submission form with the model - the context to which the model is to be uploaded.

Table 10.4 Alternate Definition details

Mapping a UML Class to an Object Class

Each class in the UML domain model is mapped to a caDSR Object Class (OC). The UML Loader performs this mapping based on the NCI concepts to which the UML class elements have been mapped through the SIW. These are included as tagged values in the annotated XML file for the UML model.

To map a UML class to a caDSR Object Class, the UML Loader retrieves the NCI concept codes for the UML class from the tagged values in the XML file and checks to see if a caDSR Object Class based on those values exists. If it exists, the UML model class is mapped to it, and the OC is classified and appended with the appropriate CS/CSI and alternate name/definition information as described in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

If there is no OC in caDSR based on the mapped NCI concept code values for the class, the UML Loader must create a new OC. [Table 10.5](#) illustrates the details of a new object class that UML Loader creates.

Object Class Attribute	Description	Example
Preferred Name	Derived from the concept codes of the underlying concept(s). Usually the concept identifier(s).	C40992
Long Name	Derived from the long name of underlying concept(s).	Homologous Protein
Preferred Definition	Derived from the preferred definition of underlying concept(s).	A protein similar in structure and evolutionary origin to a protein in another species.
Version	1.0 (default)	1.0
Workflow Status	RELEASED (default)	Released
Context	Specified in the submission form with the model - the context to which the model is to be uploaded. Default is caBIG.	caBIG
Begin Date	Current Timestamp	01/23/2005

Table 10.5 Object class attribute details

As with all caDSR components, the OC is classified and appended with the appropriate CS/CSI and alternate name/definition information from the UML model as described in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

Mapping a UML Attribute to a Property

Each attribute in the UML domain model is mapped to a caDSR Property. The UML Loader performs this mapping based on the NCI concepts to which the UML attribute elements have been mapped through the SIW. These are included as tagged values in the annotated XMI file for the UML model.

To map a UML attribute to a caDSR Property, the UML Loader retrieves the NCI concept codes for the UML attribute from the tagged values in the XMI file and checks to see if a caDSR Property based on those values exists. If it exists, the UML model attribute is mapped to it, and the Property is classified and appended with the appropriate CS/CSI and alternate name/definition information as described in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

If there is no Property in caDSR based on the mapped NCI concept code values for the attribute, the UML Loader must create a new one. [Table 10.6](#) illustrates the details of the new Property that the UML Loader creates.

Property Attribute	Description	Example
Preferred Name	Derived from the concept codes of the underlying concept(s). Usually the concept identifier(s).	C25552:C411167
Long Name	Derived from the long name of underlying concept(s).	Lead:Organization alUnit
Preferred Definition	Derived from the preferred definition of underlying concept(s).	Be in charge of.: Organizational unit like a laboratory, institute or consortium.
Version	1.0 (default)	1.0
Workflow Status	RELEASED (default)	Released
Context	Specified in the submission form with the model - the context to which the model is to be uploaded. Default is caBIG.	caBIG
Begin Date	Current Timestamp	01/23/2005

Table 10.6 Property attribute details

Mapping a UML Class to a Value Domain

Each class stereotyped as “CADSR Value Domain” or “enumeration” in the UML domain model is mapped to a caDSR value domain. The UML Loader performs this mapping using the Long Name of the value domain and the Context to which it is being loaded.

For each class with Stereotype “CADSR Value Domain,” the UML Loader looks for the following tagged Values:

- CADSR_ValueDomainDefinition
- CADSR_ValueDomainDatatype
- CADSR_ValueDomainType (E or N)
- CADSR_ConceptualDomainPublicID
- CADSR_ConceptualDomainVersion
- CADSR_RepresentationPublicID
- CADSR_RepresentationVersion

and optionally:

- ValueDomainConceptCode
- ValueDomainConceptPreferredName
- ValueDomainConceptDefinition

- ValueDomainConceptDefinitionSource
- Value DomainQualifierConceptCodeN
- ValueDomainQualifierConceptPreferredNameN
- ValueDomainQualifierConceptDefinitionN
- ValueDomainQualifierConceptDefinitionSourceN
- NCI_VD_UOM
- NCI_VD_DISPLAY_FORMAT
- NCI_VD_MIN_LENGTH
- NCI_VD_MAX_LENGTH
- NCI_VD_DECIMAL_PLACE
- NCI_VD_HIGH_VALUE
- NCI_VD_LOW_VALUE

The list of optional tags above are used to indicate a “top level” concept for a value domain. This is also referred to as a “referenced value domain.” For a complete (and regularly updated) list of the XMI tagged values used by the SIW and UML Loader, refer to the XMI Tag Reference wiki page at <https://wiki.nci.nih.gov/x/SYI8>.

If the value domain is non-enumerated (N), the reference to a top level concept indicates that the values are not explicitly listed as attributes of the value domain class, but that the data values for the attribute are constrained to children of the top level concept.

If the value domain is enumerated (E) the top level concept indicates that the permitted data values are listed explicitly as attributes of the value domain class and are children of the top level concept.

During the load process, the UML Loader creates a value domain for each class with Stereotype “CADSR Value Domain.” [Table 10.7](#) below lists the information used to create each value domain.

caDSR Field	Value Used
Long Name	UML Class Name
Preferred Name	Constructed from PublicID and version number.
Preferred Definition	The CADSR_ValueDomainDefinition tagged value.
Value Domain Type	The CADSR_ValueDomainType tagged value.
Context	From run time default values provided by model owner.
Version	1.0
Workflow Status	Specified as Default (e.g DRAFT NEW).
Conceptual Domain	The CADSR_ConceptualDomainPublicID and CADSR_ConceptualDomainVersion tagged values.

Table 10.7 caDSR fields for creating a Value Domain

caDSR Field	Value Used
Datatype	The CADSR_ValueDomainDatatype tagged value.
Begin Date	The date of the load.
Concept Derivation Rule	Optionally created from the list of Concepts in tagged values. Similar to Object Class Concept Derivation Rules.

Table 10.7 *caDSR fields for creating a Value Domain (Continued)*

Value Meanings

For each attribute associated with a “CADSR Value Domain” stereotyped class, the UML Loader looks for the following tagged values:

- ValueMeaningConceptCode
- ValueMeaningConceptPreferredName
- ValueMeaningConceptDefinition
- ValueMeaningConceptDefinitionSource
- ValueMeaningQualifierConceptCodeN
- ValueMeaningQualifierConceptPreferredNameN
- ValueMeaningQualifierConceptDefinitionN
- ValueMeaningQualifierConceptDefinitionSourceN
- NCI_VD_UOM
- NCI_VD_DISPLAY_FORMAT
- NCI_VD_MIN_LENGTH
- NCI_VD_MAX_LENGTH
- NCI_VD_DECIMAL_PLACE
- NCI_VD_HIGH_VALUE
- NCI_VD_LOW_VALUE

During the load process and after each value domain has been loaded, the UML Loader looks for a matching caDSR Value Meaning based on the Value Meaning Long Name or any mapped concepts, if present.

For Value Meanings that have concepts associated with them:

- If an existing Value Meaning is found with the same concepts in the same order (i.e., with the same Concept Derivation Rule), that VM is reused.
- If more than one Value Meaning is found with the same concepts, the first VM found is reused (the Loader can reuse any one of the matching VMs found).

For Value Meanings that do not have concepts associated with them:

- If an existing Value Meaning with the same Long Name is found, that VM is reused.
- If multiple Value Meanings are found with the same Long Name, the first VM found is reused and the following message is logged (where <VM Public ID> and <VM Version> are the public id and version of the VM being reused):

More than one VM found with the name [<VM Long Name>]. Using the VM [<VM Public ID>v<VM Version>]

If no matching VM concepts or Long Name are found, the UML Loader creates a new Value Meaning. [Table 10.8](#) below lists the information used to create each Value Meaning.

caDSR Field	Value Used
Value Meaning	Created from the concatenation of 'ValueMeaningConceptPreferredName' and qualifiers if present.
Value Meaning Description	Concatenation of 'ValueMeaningConceptDefinition[_n]' tagged values and qualifiers if present.
Begin Date	The date of the load
Concept Derivation Rule	Created from the list of Concept in tagged values. Similar to Object Class and Properties Concept Derivation Rules

Table 10.8 caDSR fields used for Value Meaning

Permissible Values

For each Value Meaning, UML Loader creates a new permissible value. [Table 10.9](#) below lists the information used to create these values.

caDSR Field	Value Used
Value	From the underlying attribute's name
Value Meaning	The existing Value Meaning name, or the Value Meaning created according to Table 10.8 .
Meaning Description	The Value Meaning Description associated with the existing Value Meaning, or the one created according to Table 10.8
Begin Date	The date of the load

Table 10.9 caDSR fields used for Permissible Values

Using a Value Domain Defined within the Model

In order for an attribute in the UML model to use a value domain defined within the model (also referred to as a *local value domain* or LVD), the attribute must already have a tagged value of type “CADSR Local Value Domain” associated with it. The value indicated in the tag is the name given to the LVD.

For example: The model contains a class that is stereotyped as “CADSR Value Domain” with the name “My Model Value Domain.” For a UML attribute to use this value domain, attribute must have a tagged value of: “CADSR Local Value Domain” / “My Model Value Domain” (name/value).

You are strongly encouraged to first try to use caDSR value domains (which can be mapped through the SIW). If you are using LVDs, you should attempt to map your LVDs to caDSR value domains to take advantage of value domain reuse. For more information on value domains and using local value domains, see [Value Domains](#) on page 21 and [Using Local Value Domains](#) on page 23.

Mapping Data Element Concepts

A Data Element Concept (DEC) is formed using the combination of a caDSR Object Class and a caDSR Property. Each class/attribute combination in the UML domain model is mapped to a caDSR DEC. The UML Loader performs this mapping based on the mapping it has already done for the model classes (to caDSR OCs) and the model attributes (caDSR Properties) as described in the previous sections.

The UML Loader checks to see if a caDSR DEC based on the OC and Property values for the class/attribute combination exists. If it exists, the UML model class/attribute combination is mapped to it, and the DEC is classified and appended with the appropriate CS/CSI and alternate name/definition information as described in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

If there is no DEC in caDSR based on the OC and Property values for the class/attribute combination, the UML Loader must create a new one. [Table 10.10](#) illustrates the details of the new DEC that the UML Loader creates.

Data Element Concept Attribute	Description	Example
Preferred Name	Derived from the Object Class Public ID and version and Property Public ID and version. A colon is used as the separator character between these values.	111111v1.0:222222v1.0
Long Name	Derived from the Object Class Long Name and Property Long Name. A space is used as the separator character between these two values.	Homologous Protein Alignment Length
Preferred Definition	Object Class Preferred Definition and Property Preferred Definition. A colon is used as the separator character between these two values.	A protein similar in structure and evolutionary origin to a protein in another species: The linear extent in space from one end to the other. Often used synonymously with distance.
Version	1.0 (default)	3.0
Workflow Status	RELEASED (default)	Draft New
Context	Specified in the submission form with the model - the context to which the model is to be uploaded. Default is caBIG.	caBIG

Table 10.10 Data Element Concept details

Data Element Concept Attribute	Description	Example
Begin Date	Current Timestamp	01/23/2005
Object Class Long Name	Object Class corresponding to the UML Class	Homologous Protein
Property Long Name	Property corresponding to the UML Attribute	Alignment Length

Table 10.10 Data Element Concept details

Mapping Data Elements

A Data Element (DE or CDE) is formed using the combination of a caDSR DEC and a value domain. Each class/attribute/datatype combination in the UML model is mapped to a caDSR DE. The UML Loader performs this mapping based on the mapping it has already done for the class/attribute combinations in the model (to caDSR DEC)s and a generic existing caDSR value domain that corresponds to the datatype of the attribute or a value domain defined in the model.

The UML Loader checks to see if a caDSR DE based on the DEC and value domain combination exists. If it exists, the UML model class/attribute/datatype combination is mapped to it, and the DE is classified and appended with the appropriate CS/CSI and alternate name/definition information as described in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

Note: If the Context into which the UML model is being loaded is different from the Context that owns the existing data element, a “USED_BY” designation is added to the existing data element to capture the use by another Context

If there is no DE in caDSR based on the DEC and value domain combination, the UML Loader must create a new one. [Table 10.11](#) illustrates the details of the new DE that the UML Loader creates.

Data Element Attribute	Description	Example
Preferred Name	Derived from the DEC Public ID and version and the Value Domain Public ID and version. A colon is the separator character.	3333333v1.0v:4444444v1.0
Long Name	Derived from the DEC Long Name and Value Domain Long Name. A space is the separator character.	Homologous Protein Alignment Length java.lang.Long

Table 10.11 Data Element details

Data Element Attribute	Description	Example
Preferred Definition	Derived from the DEC Preferred Definition and Value Domain Preferred Definition. The separator character is a colon.	A protein similar in structure and evolutionary origin to a protein in another species: The linear extent in space from one end to the other. Often used synonymously with distance. Value Domain for java language 'java.lang.Long' datatype.
Version	1.0 (default)	3.2
Workflow Status	RELEASED (default)	Draft New
Context	Specified in the submission form with the model - the context to which the model is to be uploaded. Default is caBIG.	caBIG
Begin Date	Current Timestamp	01/24/2005
Data Element Concept Long Name	The Data Element Concept corresponding to the UML Class and one of its attributes.	Homologous Protein Alignment Length
Value Domain	Corresponds to the datatype of the attribute. Supported datatypes include java classes and java primitives. Note: An updated file that defines the mapping between datatypes and value domains is located here: http://cadsrsiw.nci.nih.gov/datatype-mapping.xml .	java.lang.Long

Table 10.11 Data Element details (Continued)

Mapping UML Associations to Object Class Relationships

Each association in the UML domain model is mapped to an Object Class Relationship in caDSR.

Table 10.12 below provides details regarding the information used by the UML Loader to create new Object Class Relationships.

Object Class Relationship Attribute	Data
Preferred Name	Generated, equals source class corresponding Object Class public ID + version; target class corresponding Object Class and version.
Long Name	Derived from the role name of underlying association.

Table 10.12 New Object Class Relationship details

Object Class Relationship Attribute	Data
Preferred Definition	Derived from the type of association. Example of the derived value for preferred definition: Zero-to-Many Zero-to-One Many-to-One One-to-Many Many-to-Many Generalizes
Version	1.0
Workflow Status	Draft New – Specified as a parameter
Context	Specified as a parameter
Begin Date	Current Timestamp
Type	HAS_A
Source Low Cardinality	Derived from UML Association. The Source object is the class from which the link is drawn.
Source High Cardinality	Derived from UML Association. The Source object is the class from which the link is drawn.
Target Low Cardinality	Derived from UML Association. The Target object is the class to which the link is drawn.
Target High Cardinality	Derived from UML Association. The Target object is the class to which the link is drawn.
Direction	Navigability. Source-to-Target Target-to-Source Bidirectional

Table 10.12 New Object Class Relationship details

Mapping UML Inheritance

Each inheritance type association in the UML model is mapped to an Object Class Relationship in caDSR with the same attributes described for associations, except for Object Class Relationship Type, which for inheritance associations is “IS_A” instead of “HAS_A.”

Additionally, the child class of the association inherits all of the attributes of the parent class. DEC's are created based on the combinations of the child class and each of its parent's attributes. These combinations are mapped using the same process described in [Mapping Data Element Concepts](#) on page 138. Data elements are then created using the mapped DEC's and the associated datatypes. These combinations are mapped using the same process described in [Mapping Data Elements](#) on page 139.

In addition, all DEC and DEs are appended with the appropriate classification and alternate name/definition information as described in [Relating \(Classifying\) Metadata Items to UML Models](#) on page 124.

[Table 10.13](#) and [Table 10.14](#) below provide an overview of the information used to map the DEC and DEs for inheritance associations.

Data Element Concept Attribute	Data
Preferred Name	Child Object Class Public ID + Object Class Version: Parent Property Public ID + Property Version
Long Name	Child Object Class Long Name + Parent Property Long Name
Preferred Definition	Child Object Class Preferred Definition + Parent Property Preferred Definition
Version	1.0 (Specified as a parameter)
Workflow Status	Draft New (Specified as a parameter)
Context	Specified as a parameter
Begin Date	Current Timestamp
Object Class	Object Class corresponding to the Child UML Class
Property	Property corresponding to the Parent UML Attribute

Table 10.13 Inheritance Data Element Concept mapping

Data Element Attribute	Data
Preferred Name	DEC Public ID + DEC Version: Value Domain Public ID + Value Domain Version
Long Name	DEC Long Name + Value Domain Long Name
Preferred Definition	Derived from the underlying attribute description in the UML class diagram.
Version	1.0 – Specified as a parameter
Workflow Status	Draft New – Specified as a parameter
Context	Specified as a parameter
Begin Date	Current Timestamp
Data Element Concept	The Data Element Concept corresponding to the Child UML Class and the Parent Attribute.

Table 10.14 Inheritance Data Element mapping

Data Element Attribute	Data
Value Domain	<p>Corresponds to the datatype of the Parent Attribute:</p> <p>java.lang.String = Value Domain Name "java.lang.String"</p> <p>java.lang.Boolean = Value Domain "java.lang.Boolean"</p> <p>java.lang.Long = Value Domain "java.lang.Boolean"</p> <p>java.lang.Integer = Value Domain "java.lang.Boolean"</p> <p>java.lang.Float = Value Domain "java.lang.Float"</p> <p>java.util.Date = Value Domain "java.lang.Date"</p> <p>int = Value Domain "int"</p> <p>long = Value Domain "long"</p> <p>boolean = Value Domain "boolean"</p> <p>char = Value Domain "char"</p> <p>double = Value Domain "double"</p> <p>float = Value Domain "float"</p> <p>byte = Value Domain "byte"</p> <p>short = Value Domain "short"</p>

Table 10.14 Inheritance Data Element mapping

Reviewing and Verifying Registered Metadata

After your model is submitted for loading to caDSR, the model is first loaded to the caDSR "Sandbox." The Sandbox functions as a dry-run environment, so that model owners and CBIIT staff can be sure the model will load properly, and that the elements in the model are mapped to the appropriate caDSR metadata components. Once a successful load is verified, the UML model can be loaded to the caDSR Production database.

CBIIT provides two tools that allow you to search the caDSR and view loaded UML model elements: the UML Browser and the CDE Browser. Both of these tools are available for use on the Sandbox and Production environments. Model owners and CBIIT content curators will work primarily with the Sandbox environment until the model is ready for Production. The steps for using the browser tools are the same, regardless of environment; the only differences are the URLs used to access the different repositories.

To access the browser tools for each environment/repository:

- Sandbox UML Model Browser: <http://umlmodelbrowser-sandbox.nci.nih.gov>
- Sandbox CDE Browser: <https://cdebrowser-sandbox.nci.nih.gov>
- Production UML Model Browser: <http://umlmodelbrowser.nci.nih.gov>
- Production CDE Browser: <https://cdebrowser.nci.nih.gov>

Since this section is designed to provide information regarding review and verification of newly loaded models, the procedures direct you to use the Sandbox URLs.

Note: The information in this section is provided solely as an overview to using the caDSR browser tools. For detailed information on using the UML Model Browser or the CDE Browser, see the respective online help for those tools.

Using the UML Model Browser

Since model owners are typically more familiar with the elements in their UML models than with the CDEs in caDSR, many find the UML Model Browser easier to use to find and verify the loading and mapping of the elements from their model.

The UML Model Browser allows you to browse the caDSR repository for all elements in your project, or search the repository using the class and class element names from your model. You can also select which project, sub-project and/or package to search specifically for your model's elements.

The search result set provides links that open the CDE Browser, displaying the details of the caDSR administered components to which the elements are mapped.

To view registered UML model elements using the UML Model Browser:

1. Navigate to the following URL: <http://umlmodelbrowser-sandbox.nci.nih.gov>.

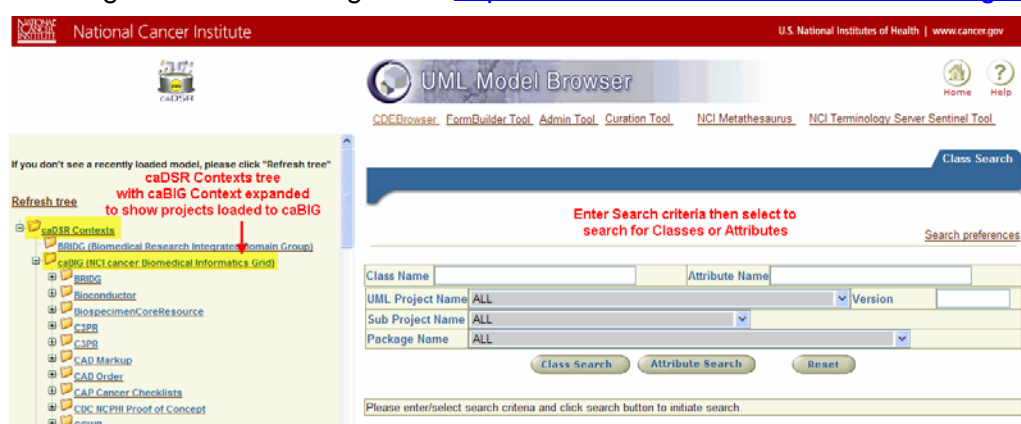


Figure 10.6 UML Model Browser showing expanded caBIG Context and Search area

2. Within the UML Model Browser window, you have the following options for browsing elements loaded to the caDSR:
 - To view all of the loaded elements for a context, click the context name from the navigation tree on the left side of the window;
 - To view all of the loaded elements for your project, click the plus sign (+) to the left of the context to which your model was loaded (caBIG unless otherwise specified in your submission) and click your project's name from the expanded list.
 - Use the right side of the window to enter search criteria, if desired select the Project Name, Sub Project Name, or Package Name you want to search, and then click either **Class Search** or **Attribute Search**.

The sections below provide more information on how to browse, search for, and view model elements using the UML Model Browser.

Viewing all registered elements for a Context or Project

When the UML Model Browser first appears, the navigation tree on the left shows all of the contexts in caDSR in “collapsed” mode. Each of the items listed in the left navigation tree is an active link. Selecting an item in the tree instructs the UML Browser to return all of the loaded model elements associated with the selected item.

You must click the plus sign to the left of a context to expand it and show all of the projects that have been loaded to that context. You must click the plus sign to the left of a project name to expand it and show all of the sub projects and packages associated with that project.

Expanding each node provides access to the active links for the items within the node, allowing you to view all of the elements for that node. This means you can view all loaded elements for a context, for a project, for a sub project or for a package, depending on the level of the node you select from the navigation tree.

Searching in the UML Model Browser

The UML Model Browser provides a straightforward search interface, allowing you to enter a class or attribute element name and then to select a project, sub project or package as well as version number to search for that class or attribute. However, it also allows you the flexibility to search for all attributes associated with a particular class, or for all classes containing a particular attribute.

To search for a class element:

1. Enter the element name in the **Class Name** text box. You may use an asterisk (*) as a wildcard if necessary.
2. If desired, use the drop-down lists to limit the search to a specific **UML Project**, **Sub Project** or **Package**, and identify a **Version** number if appropriate.
3. Click **Class Search**. The results appear below the search area.

The screenshot shows the UML Model Browser search interface. At the top, there is a header for the U.S. National Institutes of Health. Below it, there is a search area with the following fields:

- Class Name:** Gene
- Attribute Name:** (empty)
- UML Project Name:** ALL
- Sub Project Name:** ALL
- Package Name:** ALL
- Version:** (empty)

Below the search area, there are three buttons: **Class Search**, **Attribute Search**, and **Reset**. A red arrow points to the **Class Search** button with the text "Search across projects for class element 'Gene'".

Below the buttons, there is a tabbed interface with two tabs: **Classes** and **Attributes**. A red arrow points to the **Classes** tab with the text "Viewing Classes Tab".

Below the tabs, there is a table of search results. The table has the following columns: **Class Name**, **Project Name**, **Project Version**, **Project Workflow Status**, **Sub Project Name**, **Package Name**, **Class Details**, and **OC Version**. The table contains 10 rows of results for the class "Gene".

Red arrows point to the **Class Name** column header with the text "Select Class Name link to view Attributes for this project's use of the class" and to the **Class Details** column header with the text "Select Class Details link to open the CDE Browser and view the Object Class mapping for this project."

Class Name	Project Name	Project Version	Project Workflow Status	Sub Project Name	Package Name	Class Details	OC Version
Gene	caFE Server	2	RELEASED		edu.wustl.fe	2223326	1.0
Gene	Grid-enablement of Protein Information Resource (PIR) 1.2	1.2	RELEASED		edu.georgetown.pir.domain	2223326	1.0
Gene	Transcription Annotation Prioritization and Screening System	1	RELEASED	TrAPSS	edu.uiowa.clcg.trapss.domain	2223326	1.0
Gene	caArray	2	RELEASED	caArray 2.0	gov.nih.nci.caarray.domain.array	2223326	1.0
Gene	Seed	1	RELEASED		edu.uchicago.Seed.domain	2223326	1.0
Gene	caBioQ	4.1	RELEASED		gov.nih.nci.cabio.domain	2223326	1.0
Gene	Grid-enablement of Protein Information Resource (PIR) 1.1	1.1	RELEASED		edu.georgetown.pir.domain	2223326	1.0
Gene	Genomic Identifiers	1	RELEASED	UPenn	edu.upenn.med	2223326	1.0

Figure 10.7 UML Model Browser - Classes tab after search for class "Gene"

Conducting a **Class Search** against a **Class Name** returns all of the class elements and projects matching the criteria you entered. You can also enter an **Attribute Name** and a **Class Name** and click **Class Search**. In this instance, your results will be limited to only those occurrences of the class that contain the identified attribute.

From the Classes tab you have several options for viewing additional information:

- Click a **Class Name** link - this displays all of the attributes of the class for the project listed.
- Click a **Project Name** link - this displays detailed information about the project.
- Click a **Class Details** link - this opens the CDE Browser and shows the Object Class information for the OC to which this class is mapped for this project.

You may also select to view the Attributes tab. The UML Browser then performs a new search for all of the attributes in caDSR that are associated with the class name entered (for the identified projects/sub projects/packages). This is the same result you would get if you entered a **Class Name** and clicked the **Attribute Search** button.

To search for an attribute element:

1. Enter the element name in the **Attribute Name** text box. You may use an asterisk (*) as a wildcard if necessary.
2. If desired, use the drop-down lists to limit the search to a specific **UML Project**, **Sub Project** or **Package**, and identify a **Version** number if appropriate.
3. Click **Attribute Search**. The results appear below the search area.

The screenshot shows the UML Model Browser interface. At the top, there's a header with "U.S. National Institutes of Health | www.cancer.gov" and a "Search preferences" link. Below this is a search form with fields for "Class Name", "Attribute Name" (containing "diagnosis"), "UML Project Name" (set to "ALL"), "Sub Project Name" (set to "ALL"), and "Package Name" (set to "ALL"). There are buttons for "Class Search", "Attribute Search", and "Reset". A red arrow points from the "Attribute Search" button to the search results table below, with the text "Search across projects for attribute element 'diagnosis'".

Below the search form, there's a tabbed interface with "Classes" and "Attributes" tabs. The "Attributes" tab is selected. A red arrow points from the "Attributes" tab to the search results table, with the text "Viewing Attributes Tab".

The search results table has columns: "Attribute Name", "Version", "Context", "Data Type", "Definition", "DE Name", "DE Public ID", "Project Name", "Sub Project", and "Package Name". The table contains three rows of results for the attribute "diagnosis". A red arrow points from the "DE Name" column to the first row, with the text "Select links in DE Name column to open the CDE Browser and view data element details for each class/attribute/datatype combination."

Attribute Name	Version	Context	Data Type	Definition	DE Name	DE Public ID	Project Name	Sub Project	Package Name
diagnosis	1.0	caBIG	java.lang.String	Diagnosis represents the medical cause in narrative form associated with the patient's visit to the healthcare facility	Patient Health Care Visit Diagnosis java.lang.String	2720917	CDC NCPHI Proof of Concept		gov.cdc.bss.domain
diagnosis	1.0	caBIG	java.lang.String	Analysis of symptoms	Slide Diagnosis Diagnosis Study java.lang.String	2619980	caElmir		edu.ucdavis.caelmir.domain.eventRecords
diagnosis	1.0	caBIG	java.lang.String	Analysis of symptoms	Pathology Event Record Diagnosis Study Diagnosis Study java.lang.String	2619993	caElmir		edu.ucdavis.caelmir.domain.eventRecords

At the bottom of the table, there's a pagination control showing "1 - 3 of 3".

Figure 10.8 UML Model Browser - Attributes tab after search for attribute "diagnosis"

Conducting an **Attribute Search** against an **Attribute Name** returns all of the attribute elements and projects matching the criteria you entered. From the Attributes tab you can click a DE Name link for the element. This opens the CDE Browser and shows the data element information for the CDE to which this attribute (class/attribute/datatype combination) is mapped for the identified project.

You may also select to view the Classes tab. In this instance, the UML Browser performs a new search for all of the classes in caDSR that are associated with the attribute name entered (for the identified projects/sub projects/packages). This is the same result you would get if you entered an **Attribute Name** and clicked the **Class Search** button.

You may also choose to enter a **Class Name** with the **Attribute Name** and click **Attribute Search**. In this instance, your results will be limited to only those occurrences of the attribute that occur for the identified class.

Using the CDE Browser

The CDE Browser allows you to browse for or search the caDSR repository for the administered components that have been registered through and mapped to the elements from your (or any other) project. The main difference between the CDE Browser and the UML Model Browser is that the CDE Browser searches for and returns full data element information rather than specific class or attribute information for a project.

To view caDSR administered components using the CDE Browser:

1. Navigate to the following URL: <https://cdebrowser-sandbox.nci.nih.gov>.

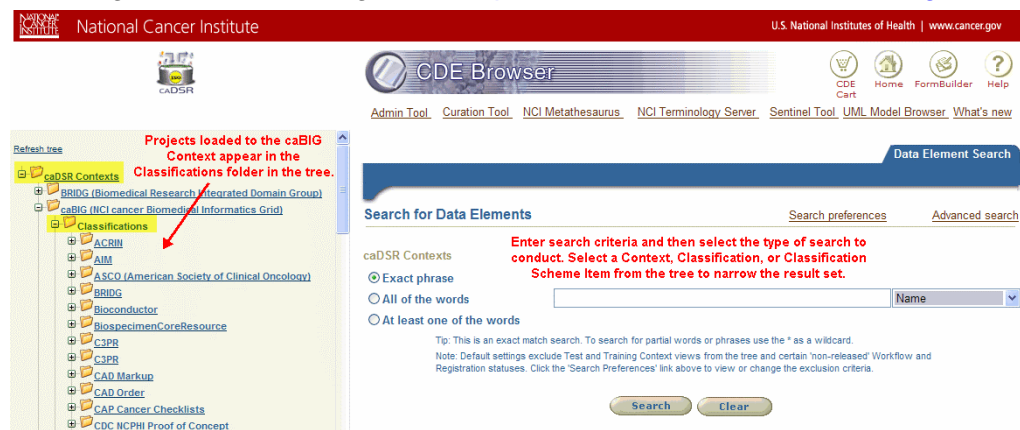


Figure 10.9 CDE Browser showing expanded Classifications under the caBIG Context

2. Within the CDE Browser window, you have the following options for browsing the administered components within caDSR:
 - To view all of the CDEs for a context, click the context name from the navigation tree on the left side of the window;
 - To view all of the CDEs mapped to your project, click the plus sign (+) to the left of the context to which your model was loaded (caBIG unless otherwise specified in your submission) and then click the plus sign (+) to the left of the Classifications node. Then click your project's name from the expanded classifications list.
 - Use the right side of the window to enter search criteria, select the type of search to perform, and then click **Search**.

The sections below provide more information on how to browse, search for, and view data element details using the CDE Browser.

Viewing all CDEs for a Context, Classification Scheme or Classification Scheme Item

When the CDE Browser first appears, the navigation tree on the left shows all of the contexts in caDSR in “collapsed” mode. Each of the items listed in the left navigation tree is an active link. Selecting an item in the tree instructs the CDE Browser to return all of the CDEs associated with the selected item.

You must click the plus sign to the left of a context to expand it and show the sub-nodes, which include a Classifications node. All projects loaded to a context are “classified” and are listed as classifications under the context. You must click the plus sign to the left of the Classifications node to expand it and show all of the projects (classification schemes or CS) for the context. You must click the plus sign to the left of a CS (which corresponds to the project name) to expand it and show all of the sub projects and/or packages (classification scheme items or CSI) associated with that project.

Expanding each node provides access to the active links for the items within the node, allowing you to view all of the CDEs for that node. This means you can view all CDEs for a context, for a CS or for a CSI, depending on the level of the node you select from the navigation tree.

Searching in the CDE Browser

The CDE Browser provides a straightforward search interface, allowing you to enter search criteria text and then to select whether you want to perform an exact search, a search for all of the words or a search for any of the words.

Unless otherwise indicated, the CDE Browser will perform the search for entered criteria across all caDSR contexts.

To search for CDEs across all caDSR contexts:

1. In the search text box, enter the item you want to search for. You may use the asterisk (*) as a wildcard.
2. Select whether you want to search for an Exact Phrase, All of the words, or At least one of the words.
3. Click **Search**.

The search results appear in a list below the search area of the window, along with several feature buttons for manipulating the resulting CDEs and a smaller “paging” indicator that lists the number of results and which of those results are currently shown.

U.S. National Institutes of Health | www.cancer.gov

Search for Data Elements 28 Matches [Search preferences](#) [Advanced search](#)

caDSR Contexts

☐ Exact phrase

☒ All of the words

☐ At least one of the words

Gene Name Name

Tip: This is an exact match search. To search for partial words or phrases use the * as a wildcard.

Note: Default settings exclude Test and Training Context views from the tree and certain "non-released" Workflow and Registration statuses. Click the "Search Preferences" link above to view or change the exclusion criteria.

[Search](#) [Clear](#) [New Search](#)

Search Results [Search within results](#)

Results fewer than expected? Check Search Preferences

[Download Data Elements to Prior Excel](#) [Download Data Elements to Excel](#) [Download Data Elements as XML](#)

Sort order : (Default) Registration Status>>Workflow Status>>Long Name [Ascending]

[Add to CDE Cart](#) [Add to CDE compare list](#) [Compare CDEs](#) 1 - 28 of 28

<input type="checkbox"/> Long Name	Preferred Question Text	Owned By	Used By Context	Registration Status	Workflow Status	Public ID	Version
<input type="checkbox"/> Gene Alias Name java.lang.String		caCORE	caBIG	Qualified	RELEASED	2223776	3.0
<input type="checkbox"/> Gene Ontology Name java.lang.String		caCORE	caBIG	Qualified	RELEASED	2529575	1.0
<input type="checkbox"/> Gene Ontology Term Name java.lang.String		caBIG		Qualified	RELEASED	2557994	1.0
<input type="checkbox"/> Gene Product Name java.lang.String		caBIG		Qualified	RELEASED	2638667	1.0
<input type="checkbox"/> Probe Set Information Summary Entrez Gene Name java.lang.String		caBIG		Qualified	RELEASED	2557974	1.0

Figure 10.10 CDE Browser showing search criteria and result set

Clicking a Long Name link from the result set opens a detailed view of the data element, including additional information tabs and links to object class, classifications and other information.

If your result set is too large, you can click the Search Within Results link. This refreshes the search window, retaining the original result set, but changing the Search button to a Search Within Results button, and changing the breadcrumbs to indicate the subset of CDEs you will be searching.

U.S. National Institutes of Health | www.cancer.gov

CDE Browser

[Admin Tool](#) [Curation Tool](#) [NCI Metathesaurus](#) [NCI Terminology Server](#) [Sentinel Tool](#) [UML Model Browser](#) [What's new](#)

[CDE Cart](#) [Home](#) [FormBuilder](#) [Help](#)

Search within results

Search Within Search Results 399 Matches [Search preferences](#) [Advanced search](#)

caDSR Contexts

Search Criteria>>All of the words (name=Gene)

☐ Exact phrase

☒ All of the words

☐ At least one of the words

Tip: This is an exact match search. To search for partial words or phrases use the * as a wildcard.

Note: Default settings exclude Test and Training Context views from the tree and certain "non-released" Workflow and Registration statuses. Click the "Search Preferences" link above to view or change the exclusion criteria.

Will limit search to within current result set [Search within results](#) [Clear](#) [New Search](#)

Search Results

Results fewer than expected? Check Search Preferences

[Download Data Elements to Prior Excel](#) [Download Data Elements to Excel](#) [Download Data Elements as XML](#)

Figure 10.11 CDE Browser - searching within results

If you want to search for CDEs within a context, CS or CSI, you must first select that item from the navigation tree on the left. This populates the breadcrumbs with the name of the node and limits the CDE search to only those related items.

To search for CDEs within a context or classification:

1. From the navigation tree on the left of the CDE Browser window, click on the name of the grouping you want to search. You can click directly on a context to search that context, or you may have to expand (using the plus sign) the context and classification nodes to get to the name of the CS or CSI you want to search.
2. After the result set shows the CDEs for the node you selected and the breadcrumbs display the node you want to search, enter the search criteria into the text box, and select the proper search type (exact, all words, any words).
3. Click **Search**.

The new result set appears and the breadcrumbs change to reflect the new search used to create the currently displayed result set.

APPENDIX

A

TROUBLESHOOTING

In addition to the troubleshooting tips listed below, the caDSR team maintains a list of commonly asked questions and tips about the SIW and UML Model Loader on the UML modeling FAQs wiki page located at: <https://wiki.nci.nih.gov/x/7wFy>.

The Status Bar

At the bottom of the main viewer window, the Status Bar displays confirmation and informative messages. Use it to confirm that a file was saved, that changes were applied or need to be applied.

Apply Changes vs. Saving Changes

Apply - Applies the changes made to the concept information. The Apply button is disabled until all concept fields are populated for newly added concepts, or until a change is made to existing concept information.

You must click **Apply** before navigating away from the changed item's node, or your changes will be discarded. The SIW prompts you to apply your changes if you attempt to view another node before clicking **Apply**.

Clicking **Apply** is not the same as saving the file. Apply simply applies your changes to the currently open session. You must still save the file (**File > Save** or **File > Save As**) to create an updated XMI file containing the applied changes.

Multiple Tabs Open

Each type of element in the SIW viewer will open in its own tab. this means that if you select different types of elements from the navigation tree to view, you will likely have multiple tabs open at once. Click the element title on each tab to see the information about that element; use the "X" on the tab to close the tab. See [Detail View](#) on page 37 for more information on tabbed viewing.

Search Dialog Box

The SIW contains several different search functions, including the EVS search, the data element search and the value domain search. The different search functionality for each item is similar but not identical. Furthermore, when you open a search dialog box to perform a search, you may find the previous search information still located in the dialog box and/or result set. Not all of the searches clear their information between search sessions.

All of the search features, however, allow you to resize both the dialog box and the column width of the result set, providing better visibility of the items listed. In addition, the **Previous** and **Next** buttons always function as “page turners” when there are multiple pages of results, and the **Close** button always closes the dialog box without performing any mapping.

For more information about the different search capabilities, see the following sections of this guide:

- [*Searching EVS for Concept Information*](#) on page 78
- [*Mapping a UML Attribute to an Existing Common Data Element*](#) on page 100
- [*Mapping a UML Attribute to a Value Domain*](#) on page 104.

GLOSSARY

This glossary describes acronyms, objects, tools and other terms referred to in the chapters or appendixes of this guide.

Term	Definition
<code>{home_directory}</code>	Indicates the directory where you installed the SDK
<code>{package_structure}</code>	Indicates the package structure from the UML models
<code>{project_name}</code>	Indicates the <code>project_name</code> specified in the <code>deploy.properties</code> file
AGT	Artifact Generation Tool
AOP	Aspect Oriented Programming
API	Application Programming Interface
Writeable API	Methods exposed by the CSM to create, update and delete a domain object. These methods are generated using the code generation component.
Application Service	This refers to the CSM interface which exposes all the writeable as well as business methods for a particular application
BO	Business Object
C3D	Cancer Centralized Clinical Database
caBIG	cancer Biomedical Informatics Grid
caBIO	Cancer Bioinformatics Infrastructure Objects
caCORE	cancer Common Ontologic Representation Environment
caDSR	Cancer Data Standards Repository
caMOD	Cancer Models Database
cardinality	Cardinality describes the minimum and maximum number of associated objects within a set
CASE	Computer Aided Software Engineering
CCR	Center of Cancer Research
CDE	Common Data Element
CGAP	Cancer Genome Anatomy Project

Term	Definition
CMAP	Cancer Molecular Analysis Project
CODEGEN	Code generator tool
Code Generator Tool	The SDK tool that leverages Model-Driven Architecture to convert a UML model to a fully-functioning caCORE-like system
CS	Classification Scheme
CSI	Classification Scheme Item
CSM	Common Security Module
CTEP	Cancer Therapy Evaluation Program
CVS	Concurrent Versions System
DAO	Data Access Objects
DAS	Distributed Annotation System
DCP	Division of Cancer Prevention
DDL	Data Definition Language
DEC	Data Element Concept
DOM	Document Object Model
DTD	Document Type Definition
DU	Deployment Unit
EA	Enterprise Architect
EBI	European Bioinformatics Institute
EMF	Eclipse Modeling Framework
EVS	Enterprise Vocabulary Services
FK	Foreign Key - A collection of columns (attributes) that enforce a relationship to a Primary Key in another table used in data model tables in Enterprise Architect
FreeMarker	A "template engine"; a generic tool to generate text output (anything from HTML or RTF to auto generated source code) based on templates
GAI	CGAP Genetic Annotation Initiative
GEDP	Gene Expression Data Portal
IDE	Integrated Development Environment
ISO	International Organization for Standardization
JAF	JavaBeans Activation Framework
Jalopy	Source code formatting tool for the Sun Java Programming Language. (http://jalopy.sourceforge.net/manual.html)
JAR	Java Archive
Javadoc	Tool for generating API documentation in HTML format from doc comments in source code (http://java.sun.com/j2se/javadoc/)
JDBC	Java Database Connectivity

Term	Definition
JDiff	Javadoc doc-let which generates an HTML report of all the packages, classes, constructors, methods, and fields which have been removed, added or changed in any way, including their documentation, when two APIs are compared (http://javadiiff.sourceforge.net/)
JET	Java Emitter Templates
JMI	Java Metadata Interface
JSP	JavaServer Pages
JUnit	A simple framework to write repeatable tests (http://junit.sourceforge.net/)
MDR	Metadata Repository
metadata	Definitional data that provides information about or documentation of other data.
MMHCC	Mouse Models of Human Cancers Consortium
multiplicity	Multiplicity of an association end indicates the number of objects of the class on that end that may be associated with a single object of the class on the other end
MVC	Model-View-Controller, a design pattern
NCI	National Cancer Institute
NCICB	National Cancer Institute Center for Bioinformatics
NSC	Nomenclature Standards Committee
navigability	Navigability defines the visibility of an object to its associated source/target object at the other end of an association. Navigability is the same as directionality.
OMG	Object Management Group
OR	Object Relation
ORM	Object Relational Mapping
PCDATA	Parsed Character DATA
persistence layer	Data storage layer, usually in a relational database system
PK	Primary Key – Key used to uniquely identify a data model table in Enterprise Architect.
QA	Quality Assurance
RDBMS	Relational Database Management System
RUP	Rational Unified Process
SCM	Software Configuration Management
SDK	Software Development Kit
Semantic Connector	The SDK tool that links model elements to NCICB EVS concepts.
SPORE	Specialized Programs of Research
SQL	Structured Query Language

Term	Definition
Tagged value	A UML construct that represents a name-value pair; can be attached to anything in a UML model. Often used by UML modeling tools to store tool-specific information
UML	Unified Modeling Language
UML Loader	SDK tool that converts a domain model in UML to corresponding common data elements (CDEs) in caDSR.
UPT	User Provisioning Tool
URI	Uniform Resource Identifier
URL	Uniform Resource Locators
WSDL	Web Services Description Language
XMI	XML Metadata Interchange (http://www.omg.org/technology/documents/formal/xmi.htm) - The main purpose of XMI is to enable easy interchange of metadata between modeling tools (based on the OMG-UML) and metadata repositories (OMG-MOF) in distributed heterogeneous environments
XML	Extensible Markup Language (http://www.w3.org/TR/REC-xml/) - XML is a subset of Standard Generalized Markup Language (SGML). Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML
XP	Extreme Programming

Symbols

A

Add button 38

adding a concept 38

adding concept mapping to an element 76

adding concepts in review mode 96

administered component 121, 122

aggregations 10

Alternate Definition

as used for classification 124

creating for data element 130

creating for DEC 131

creating for object class 128

creating for Property 130

description 128

Alternate Name

as used for classification 124

creating for data element 129

creating for DEC 131

creating for object class 128

creating for Property 130

description 128

alternate names 111

annotated file

definition 18

editing concept mapping 95

final output 93

mapping to EVS concepts 122

model owner verified 105

naming convention 49

registering with UML Loader 122

reviewing after Semantic Connector 71

review step 34

submit for registration 122

submitting for registration 124

viewing in SIW 40

annotation

adding concept mapping to element 38

annotated file 18

associations 57

cleanup GME tags 117

description 18

editing mapped concepts 95

generate GME tags 113

mapping elements to CDE 99

mapping elements to concepts 18

mapping via Semantic Connector 67

overview 55

purpose of annotation 55

purpose of element descriptions 56

purpose of viewing unannotated file 55

review annotated file 18, 34, 91

review unannotated file 18, 32, 51

searching EVS for concept to map 78

unannotated file 18

verifying reviewed file 105

via Curate XMI file step 73

via Semantic Connector task 71

viewing annotated file 40

viewing semantic order 40

viewing unannotated file 39

Apply button 38

applying changes 38, 82

apply to all 86

apply vs. save 39, 151

ArgoUML 17, 28

file extensions 48

generating XMI file 13

output type 10

arrows for reordering concepts 83

associations 10, 42

annotating 57

definition 17

location in tree view 37

mapping elements to concepts 42

mapping to object class relationship 140

recognized relationships 10

UML model constraints 10

viewing in tree view 45

viewing with classes 45

attribute

as model element 17

as value meaning 25

complex type 10

datatype mapping to value domain 103

definition 17

description tags 11, 39, 56

display inherited attributes 46

mapping datatype to value domain 103

mapping to caDSR administered component 122

mapping to CDE 99, 103

mapping to CDE via Roundtrip task 64

mapping to concepts via Curate mode 76

mapping to DEC 20

mapping to EVS concepts 17

mapping to existing Property 133

mapping to Property 133

mapping to value domain 104

naming convention 12

- naming convention in model 11
- sort by name in tree view 46
- tagged values 18
- using local value domain 23, 26, 137
- audience for Curate XMI file section 74
- audience for guide 2
- automatically search EVS on EVS link 44, 45

B

- before you begin 10
- browsing metadata by project 144, 147
- browsing metadata through search 145, 148

C

caCORE

- architecture overview 7
- components 8
 - caDSR 9
 - Common Logging Module (CLM) 9
 - Common Security Module (CSM) 9
 - EVS 9
 - SDK 9
 - SIW 8
 - UML Loader 9

- definition 1
- UML modeling 17
- website links 15
- wiki links 15

caDSR

- accessing UML derived metadata 143
- administered components 121, 122
- as caCORE component 9
- CDE Browser 126
- classifications 124
- contexts 126
- creating Alternate Definitions 132
- creating new object class 133
- creating new Property 134
- description 2, 9, 27
- mapping attribute to Property 138
- mapping CDEs to model elements 99
- object class details 133
- Property details 134
- registering metadata from UML models 121
- required file for loading 93
- reviewing model element mapping 143
- sandbox 124
- using tools to browse metadata 143

CADSR_Description tags 11, 39, 56

CADSR Value Domain 135

caGrid 110

camel case 11, 12, 67

CDE Browser

- reviewing metadata 143
- searching 148
- using 147
- viewing classifications 126

CDE see "common data element"

changing the order of mapped concepts 82, 98

checkmarks in tree view 106

class

- adding value meaning attributes 25
- as local value domain 24
- as model element 17
- definition 17
- describing in model 11, 39
- documentation tags 11, 39, 56
- mapping to caDSR administered component 122
- mapping to CDE via Roundtrip task 64
- mapping to concepts via Curate mode 76
- mapping to DEC 20
- mapping to OC 103, 128, 132
- mapping to value domain 134
- naming convention 11, 12
- sorting by name in tree view 46
- stereotyped as enumeration 24
- stereotyped as value domain 10
- stereotyping as CADSR Value Domain 24
- viewing by name in tree view 46
- viewing inherited attributes from super class 46

classifications 124, 147

classification scheme 126

classification scheme item 126

classifying metadata 124

cleanup GME tags 117, 118

common data element (CDE)

- see also "data element"
- definition 20, 139
- mapping through UML Loader 139
- mapping to model element 21
- viewing all CDE for a classification 147

Common Logging Model (CLM) 9

Common Security Model (CSM) 9

comparing concepts in model against EVS 87

compatibility with UML modeling programs 28

concept

- mapping 19

concept code summary 44

concept name summary 44

concepts

- adding in review mode 96
- adding new 73, 76
- adding to element mapping 38
- annotating associations 57
- attribute details 131
- changing order of mapped concepts 82, 98
- definition 18

- deleting mapped concepts 84, 99
- display Primary Concept first 46
- editing information 85
- editing mapping 38, 81
- mapping association elements 42
- mapping by EVS curators 33
- mapping example 77
- mapping in semantic order 40, 83
- mapping through Semantic Connector 33
- mapping to model element 18
- mapping via Curate XMI step 73
- Primary Concept 18
- Qualifier Concept 18
- replacing mapped concepts 82, 97
- review annotated file (concept mapping) 34
- semantic order 41, 98
- updating in caDSR 131
- using to map attribute to Property 133
- using to map class to OC 132
- validating model against EVS 87
- viewing element mapping details 37
- viewing elements after mapping 40
- viewing elements before mapping 39
- viewing semantic order 40

context 126

controlled vocabulary 15

creating CSIs 126

Curate XMI file 33, 73

- verifying mapping 89

D

data element (DE)

- see also "common data element"
- as metadata 20
- creating Alternate Definitions 129
- creating Alternate Names 129
- creating new 139
- definition 20, 139
- description 139
- graphical representation 20
- mapping to attribute in SIW 103
- mapping to model element 21
- mapping via Roundtrip mode 59
- using existing 139

data element concept (DEC)

- creating new 138
- definition 20, 138
- description 138
- mapping to class element 20
- mapping to existing 138
- mapping to model elements 138

datatype

- defined as local value domain 137

DEC see "data element concept"

default GME tags 113

definitions

- Alternate Definition 128
- Alternate Name 128
- annotated XMI file 18
- association 17
- common data element (CDE) 20
- concepts 18
- data element (DE) 20
- data element concept (DEC) 20, 138
- EVS concepts 18
- metadata 16
- Primary Concept 18
- Qualifier Concept 18
- semantic order 40
- tagged values 18
- terms you should know 16
- UML 17
- unannotated XMI file 18

deleting mapped concepts 84, 99

description tags 11, 39, 51, 56

DE see "data element"

detail view 37

- Add button 38
- Apply button 38
- automatically launch EVS search 45
- display Primary Concept first 46
- map to CDE 100, 103
- map to concepts 100, 103
- multiple tabs 151
- Next button 38
- Previous button 38
- Search EVS button 38
- search for data element 101
- semantic order of concepts 41
- viewing element descriptions 45

detail view for associations 42

display inherited attributes 46

display Primary Concept first 46

E

editing concept information 85

editing concept mapping 81

editing concepts in review mode 95

editing element mapping 38

element see "model elements"

Enterprise Architect (EA)

- file extensions 48
- generating XMI file 13
- output type 10, 17, 28

enumerated value domain 25

errors tab 42, 44

EVS

- as caCORE component 9
- concept definition 18

- concepts 122
- description 9
- searching for concepts 78
- EVS curation team 73
- EVS curation team verifying mapping 89
- exporting errors to file 43

F

- Fatal Error in Model dialog box 54
- filtering log tab information 44
- fix XMI file task 28
- freestyle search 101
- fully annotated XMI file 34

G

- Generate Default GME Tags 117, 119
- generating updated errors tab 44, 92
- GME alternate names 111
- GME and caGrid 110
- GME namespace tags 113
- GME order of operations 111
- GME Overview 109
- GME scenarios 111
- GME tagged values 111
- GME tags 44, 113, 117
- GME tags cleanup 118
- GME tags in SIW 115
- GME tags in SIW viewer 48
- GME tags used 110

H

- human verified 89

I

- inheritance mapping 47
- inheritances 10, 141
- inherited attributes 46

J

- Java naming conventions 11
- Java term restrictions 11
- Java Web Start 28

L

- local value domain 137
 - as attribute datatype 26
 - CADSR value domain 24
 - enumerated 25
 - non-enumerated 24
 - referenced value domain 24

- tagged values 24
- using within model 23, 100, 137
- value meanings 25
- local value domains
 - packaging in the model 23
- log tab 42, 44

M

- mapping
 - applying changes 38
 - association elements 57
 - association elements to concepts 42
 - attribute element to Property 133
 - attributes to concepts 41
 - attribute to CDE 99, 100, 103
 - attribute to Property 134
 - based on previous model 32, 59
 - by EVS curators 33
 - changing semantic order 83
 - changing the order of concepts 82, 98
 - class element to OC 132, 133
 - class element to value domain 134
 - elements to concepts 18
 - elements to concepts via Curate task 73
 - elements to concepts via Semantic Connector 67
 - elements to EVS concepts 56
 - human verified 89
 - local value domains 23
 - rejecting changes 82
 - replacing mapped concepts 82
 - replacing mapped concepts in review mode 97
 - review annotated file 91
 - reviewing element mapping 95
 - search for data element 101
 - searching EVS for concept to map 78
 - semantic order of concepts 40, 98
 - stereotyped class element to value domain 134
 - to EVS concepts through curation mode 33
 - to EVS concepts using Semantic Connector 33
 - undo changes 82
 - using local value domains 100
 - using Semantic Connector 33
 - value domains 22
 - verifying by curator 89
 - verifying element mappings 106
 - viewing semantic order of concepts 41
- Map to CDE button 100
- Map to Concepts button 100
- metadata
 - definition 16
 - mapping to model elements 20
 - purpose 17
 - registering 19
 - registering through UML Loader 121

- relationship between model and caDSR 124
- using pre-mapped CDEs 131
- model elements
 - adding a concept 38, 76, 96
 - annotated file requirements 92
 - annotating 18
 - applying changes 38
 - associations 17
 - attribute 17
 - camel case naming 11, 67
 - changing the order of mapped concepts 82, 83, 98
 - class 17
 - classifying 124
 - deleting mapped concepts 84, 99
 - descriptions in model 11
 - editing concept information 81, 85
 - editing mapping 38
 - example of mapping to concepts 77
 - mapping attribute to DEC 20
 - mapping attribute to value domain 104
 - mapping class to DEC 20
 - mapping to caDSR administered component 21, 122
 - mapping to CDE 21, 99, 100
 - mapping to EVS concepts 18, 56
 - marking as verified 89, 106
 - naming convention 11
 - providing descriptions for 18
 - purpose of element descriptions 56
 - registering as metadata 121
 - replacing mapped concepts 82, 97
 - review annotations 34
 - review descriptions 32
 - review mapping 34
 - searching caDSR 145, 148
 - searching CDE Browser 148
 - searching UML Model Browser 145
 - semantic order for mapped concepts 40
 - tagged values 11, 18
 - UML descriptions 11, 39
 - value domain 21
 - verify concept mapping 34
 - verifying curation mapping 89
 - viewing after mapping 40
 - viewing before mapping 39
 - viewing descriptions 51
 - viewing details 37
 - viewing element annotations 40
 - viewing element descriptions 37, 39
 - viewing in caDSR 144, 147
 - viewing in SIW 35
 - viewing multiple tabs 151
 - viewing semantic order of concepts 41
- modeling programs 28
- model owner verified 105

N

- navigating the SIW viewer 35, 38
- navigation tree
 - associations in tree view 45
 - checkmarks in tree view 106
 - display inherited attributes 46
 - sort element by name 46
 - viewing associations 45
- Next button 38
- non-enumerated value domain 24

O

- object class
 - creating new for mapping 133
 - mapping to class element 132
 - reusing existing for mapping 132
- object class relationship 140
- opening the SIW 29
- order of GME operations 111
- output from Curate XMI step 74, 90
- output from Generate GME Tags mode 117, 119
- output from generate GME tags mode 113
- output from GME Cleanup mode 118
- output from Roundtrip mode 65
- output from Semantic Connector mode 68, 71
- overview
 - GME 109

P

- packages 126
- perform Roundtrip XMI step 32
- Previous button 38
- Primary Concept
 - definition 18, 76
 - viewing first in detail view 46
- primary concept 40
- Property
 - creating new for mapping 134
 - mapping attribute element 133, 138
 - reusing existing for mapping 133

Q

- Qualifier Concept
 - definition 18, 76
- qualifier concept 40

R

- referencing metadata by project 124, 126
- regenerate errors listing 44
- registering metadata to sandbox 124
- rejecting changes to concept mapping 82

- rejecting changes to concepts 96
- rejecting concept deletion 84
- replacing mapped concepts 82, 97
- review annotated file 18, 34, 93
- review GME tags 115
- reviewing registered metadata 143
- review unannotated file 18, 32, 51
- Roundtrip XMI step 32, 59

S

- sandbox 124, 143
- save vs. apply 48, 151
- saving the XMI file 39, 48
- saving verified curated file 90
- SDK as caCORE component 9
- search EVS 78, 79
 - automatic search launch 45
 - result set limit 81
 - Search EVS button 38, 44
- search for data element
 - freestyle search button 101
 - from detail view 101
 - result set 102
 - search data element button 101
 - suggest button 102
- search for value domain 105
- searching for registered metadata 148
- searching for registered model elements 145
- searching for your model elements in caDSR 148
- searching the CDE Browser 148
- searching the UML Model Browser 145
- Search Thesaurus dialog box 79
- security warning dialog box 30
- selecting classes or packages to process 35
- semantic annotation
 - purpose 15
- Semantic Connector 56, 67
- Semantic Connector Report 68, 71
- semantic integration
 - annotation 15, 18
 - before you begin 10
 - defined terms 16
 - process 18
 - purpose 19, 27
 - UML model constraints 10
- semantic order
 - changing order of mapped concepts 83, 98
 - concept order reflects semantic order 98
 - definition 40
 - example of concept mapping 83
- show concept code summary 44
- show concept name summary 44

- show GME tags 44
- SIW
 - as caCORE component 8
 - before using 28
 - cleanup GME tags 117
 - Curate XMI file step 33
 - default preferences 44
 - description 1, 8, 27
 - detail view 37
 - errors tab 42
 - exporting error information 43
 - filtering log tab information 44
 - generating GME tags 113
 - generating XMI file from UML model 13
 - Java Web Start 28
 - launching 29
 - log tab 42, 44
 - model pre-requisites 10
 - navigating the viewer 35
 - navigation tree 36
 - options 31
 - parsing select classes or packages 35
 - perform Roundtrip XMI step 32
 - process 31
 - processing select packages 35
 - purpose 18, 27
 - review annotated file 18, 34, 91
 - review GME tags 115
 - review unannotated file 18, 32
 - running Semantic Connector 67
 - security warning on initial open 30
 - Semantic Connector step 33
 - setting preferences 44
 - tree view 36
 - updating errors tab information 44
 - viewer overview 35
 - viewer preferences 44
 - viewing associations 42, 45
 - viewing errors 42
 - viewing multiple tabs in detail view 38
 - viewing parsing log 42
 - viewing semantic order of concepts 41
 - welcome screen 31

SIW Viewer

- show GME tags 48
- SIW viewer 35
 - Add button 38
 - Apply button 38
 - associations in detail view 42
 - associations in tree view 42, 45
 - checkmarks on navigation tree 37
 - detail view 37, 38
 - display inherited attributes 46
 - display Primary Concept first 46
 - element descriptions in detail view 45

- errors tab 42
- exporting error information 43
- filtering log tab information 44
- log tab (parsing log) 42, 44
- multiple tabs in detail view 38
- navigation tree 36
- Next button 38
- preferences 44
- Previous button 38
- reviewing annotated file 95
- Search EVS button 38
- setting preferences 44
- sort element by name 46
- structural order of tree view 37
- updating errors tab 44
- use pre-thesaurus to validate concepts 47
- viewing associations 42, 45
- viewing element descriptions 45
- viewing errors 42
- viewing semantic order 41
- sort by name 46
- starting SIW 29
- stereotyped class as value domain 134
- stereotyped class tagged values 135
- stereotyped class value meanings 136
- suggest data element matches 102

T

- tabs in detail view 38
- tagged values
 - associations 57
 - CADSR_Description 39
 - CADSR Local Value Domain 137
 - CADSR Value Domain 135
 - caDSR value domain 22
 - class stereotyped as value domain 134
 - definition 18
 - description tags 39
 - element description tags 39
 - local value domain 10, 23, 24
 - purpose of element descriptions 56
 - referenced value domain 24, 135
 - review unannotated file 32
 - role annotations (associations) 57
 - source annotations (associations) 57
 - target annotations (associations) 58
 - updating in model from XMI file 49
 - value domain 22, 135
 - value meanings 25, 136
 - verifying element descriptions 55
- troubleshooting detail view 38

U

- UML definition 17

UML descriptions

- viewing in tree view 45

UML Loader

- as caCORE component 9
- creating CSIs 126
- creating new concepts in caDSR 131
- creating new data element 139
- creating new DEC 138
- creating new object class 133
- creating new Property 134
- description 2, 9, 121
- mapping associations 140
- mapping attribute element to Property 133
- mapping class/attribute to DEC 138, 139
- mapping class element to OC 132
- mapping data elements 139
- mapping DEC's 138
- mapping DE to model elements 21
- mapping inheritances 141
- mapping model elements to metadata 121
- process of registration 124
- specifying caDSR classification 126
- updating concepts in caDSR 131
- using existing data element 139
- using existing DEC's 138
- using pre-mapped CDEs 131

UML model

- class operations 122
- constraints 10
- description tagged values 11, 39, 56
- exported file type 28
- mapping attributes 100
- mapping elements to caDSR components 122
- pre-requisites 10
- submitting for registration 124
- supported modeling software 10, 28
- updating model definitions 49

UML Model Browser 144, 145

UML modeling programs 28

unannotated file

- definition 18
- review step 32, 51
- viewing in SIW 39

- undo changes to mapping 82

- undo concept changes 96, 99

- undo concept deletion 84

- updating errors tab 44, 92

- updating model with SIW changes 49

- use pre-thesaurus to validate concepts 47

- using arrows to change concept order 83

- using CDE Browser 143, 147

- using CDEs mapped through SIW 131

- using UML Model Browser 144

V

- validate concepts 87
- validate concepts using pre-thesaurus 47
- value domain
 - as model element 21
 - CADSR value domain 24
 - definition 21
 - enumerated 22, 25, 135
 - local value domain 10, 23
 - mapping LVD to caDSR 23
 - mapping stereotyped class element 134
 - mapping to CDE via Roundtrip task 64
 - mapping to datatype 103
 - mapping with data element 139
 - non-enumerated 22, 24, 135
 - referenced value domain 24
 - search for value domain in SIW 105
 - stereotyped class 10
 - stereotyped class elements 134
 - tagged values 135
 - tagged values for LVD 24
 - using caDSR value 22
 - using local value domain 137
 - using LVD for attribute datatype 26
 - value meanings 25, 136
- value meanings 25, 136
- verifying concept information in model 87
- verifying curated XMI file 89
- verifying reviewed file 105
- view associations in class tree 45
- viewer default setting 44
- viewer preferences 44
 - automatically search EVS on EVS link 45
 - display inherited attributes 46
 - display Primary Concept first 46
 - display UML description last 45
 - sort element by name 46
 - use pre-thesaurus to validate concepts 47
 - viewing associations in class tree 45
- viewing
 - all CDEs for a classification 147
 - annotated file 40
 - associations 42
 - classifications in CDE Browser 126
 - contexts in CDE Browser 126
 - element annotations 40
 - element descriptions 39, 45, 51
 - inherited attributes in tree view 46
 - log tab (parsing log) 44
 - metadata by project 147
 - model elements in caDSR 144, 147
 - navigation tree sorted by class name 46
 - Primary Concept first in detail view 46
 - registered metadata 147

- registered model elements 144
- semantic order 40
- unannotated file 39, 51
- unannotated file errors 55

- vocabulary terms 16

W

- warnings on mapped inheritance 47
- website links 3
- when to use GME 111
- wiki links 3, 18, 22
- workflow for semantic integration 18
- workflow for the SIW 31

X

- XMI file
 - annotate based on previous model 32
 - annotated file 18
 - applying changes 39
 - as EA or ArgoUML output 10
 - compatible modeling software 17
 - curate file through EVS curators 33
 - definition 17
 - element descriptions in file 39, 56
 - file extensions 48
 - filtering packages 35
 - fully annotated file 93
 - generating for SIW 13
 - model output file extensions 10, 28
 - model owner verified 105
 - naming a reviewed file 93
 - output from Curate XMI step 90
 - package filtering 35
 - processing select packages 35
 - purpose for element descriptions 56
 - review annotated file 91
 - reviewing annotated file 34
 - reviewing newly exported file 32
 - Roundtrip XMI step 59
 - saving changes 48
 - Semantic Connector 33
 - submitting annotated file to caDSR 124
 - unannotated file 18
 - uploading model definitions 49
 - valid file export from model 28
 - verify concept mapping 34
 - verifying reviewed file 105
 - viewing annotated file 40
 - viewing element descriptions 51
 - viewing errors in file 42
 - viewing model elements 35
 - viewing structural order of model 37
 - viewing unannotated file 39, 51
- XMI Roundtrip 59

XMI tag reference [111](#)
XSD [109](#)

