caAdapter Model Mapping Service (MMS) 4.1

User's Guide

NATIONAL® CANCER INSTITUTE

Center for Biomedical Informatics
and Information Technology

August 15, 2008

CONTENTS

About This Guide

This section introduces you to the *caAdapter MMS 4.1 User's Guide*.

Topics in this section:

Purpose

This guide is the companion documentation to the caAdapter Model Mapping Service (MMS) Tool. The MMS Tool is part of the caAdapter toolset. The guide includes information and instructions for using the mapping tool graphical user interface (GUI) to map an object model to a data model.

Audience

Typical User

This guide is designed for developers who wish to map a data model to an object model. This mapping is necessary for building caCORE compatible applications.

Prerequisites

This guide assumes that you are familiar with the following concepts:

- UML modeling
- Enterprise Architect, or ArgoUML
- Object and data model terms and processes

Organization of This Guide

The *caAdapter MMS 4.1 User's Guide* includes the following chapters:

- *Chapter 1, Overview of caAdapter,* on page 5 discusses the caAdapter architecture and related data standards.
- *Chapter 2, Using the caAdapter MMS Tool,* on page 13 provides detailed instructions for using the caAdapter GUI for object model to data model mapping.
- *Appendix A, References*, on page 29 provides a list of references used to produce this guide or referred to within the text.
- *caAdapter Glossary* on page 31 defines acronyms, objects, tools and other terms related to the caAdapter MMS Tool.

Recommended Reading

The following table lists resources that can help you become more familiar with concepts discussed in this guide.

Resource	***URL***
Unified Modeling Language (UML)	http://www.cdisc.org/models/sds/v3.1/

Click the hyperlinks throughout this guide to access more detail on a subject or product.

Text Conventions Used

This section explains conventions used in this guide. The various typefaces represent interface components, keyboard shortcuts, tool bar 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.
Italics	Highlights references to other documents, sections, figures, and tables.	See *Figure 4.5*.
Italic boldface monospaced type	Represents text that you type.	In the **New Subset** text box, enter ***Proprietary Proteins.***
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 people contributed to the development of this document.

caAdapter Development and Management Teams		
Development	***User's Guide***	***Program Management***
Harsha Jayanna [2]	Harsha Jayanna [2]	Anand Basu [1]
Chunqing Lin [3]	Nick Schroedl [3]	Christo Andonyadis [1]
Sandeep Phadke [3]	Ki Sung Um [1]	Sichen Liu [1]
Nick Schroedl [3]	Ye Wu [3]	Sharon Settnek [3]
Ki Sung Um [1]	Eugene Wang [3]	Smita Hastak [2]
Eugene Wang [3]	Wendy Ver Hoef [2]	
Ye Wu [3]	Charles Yaghmour [2]	
Wendy Ver Hoef [2]	Carolyn Kelley Klinger [4]	
Charles Yaghmour [2]		

caAdapter Development and Management Teams		
Development	***User's Guide***	***Program Management***
[1] National Cancer Institute Center for Bioinformatics and Information Technology (NCI CBIIT)		[2] ScenPro, Inc.
[3] Science Application International Corporation (SAIC)		[4] Lockheed Martin Management System Designers

Contacts and Support	
NCICB Application Support	http://ncicb.nci.nih.gov/NCICB/support Telephone: 301-451-4384 Toll free: 888-478-4423

LISTSERV Facilities Pertinent to caAdapter		
LISTSERV	***URL***	***Name***
caAdapter_Users	https://list.nih.gov/archives/caadapter_users-l.html	caAdapter Users Discussion Forum
caBIO_Users	https://list.nih.gov/archives/cabio_users.html	caBIO Users Discussion Forum
caBIO_Developers	https://list.nih.gov/archives/cabio_developers.html	caBIO Developers Discussion Forum

Release Schedule

This guide has been updated for the caAdapter MMS 4.1 release. It may be updated between releases if errors or omissions are found.

CHAPTER

1

OVERVIEW OF CAADAPTER

This chapter provides an overview of caAdapter, its architecture, its various tools, and its related data standards.

Topics in this chapter include:

About caAdapter

The caAdapter (http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caadapter) consists of several components that support data sharing at NCI CBIIT (http://ncicb.nci.nih.gov) and/or cancer centers as part of the cancer Biomedical Informatics Grid (caBIG) (http://caBIG.nci.nih.gov) solution. The components include a core engine and a mapping tool for providing mapping and transformation services.

The caAdapter Core Engine is an open source toolkit which contains essential modules that drive various caAdapter capabilities. It includes modules for parsing and loading schema information for Comma Separated Value (CSV), HL7 v2 and v3, SDTM, and XMI files. It also includes schema validation, data transformation, and data viewing modules.

The caAdapter Mapping Tool is an open source application that supports several types of mapping and transformation. It enables analysts and database engineers, who are

knowledgeable about HL7, to create a mapping from CSV clinical data to an equivalent target HL7 v3 XML format. An analyst could use the Mapping Tool to map and convert HL7 v2 to HL7 v3 messages, and to map a CSV file to SDTM datasets. Moreover, the Mapping Tool supports basic metadata edit capabilities, drag-and-drop mapping between source and target elements, validation of specifications and mappings, and transformation of data between various formats.

The Model Mapping Service component of caAdapter allows users to parse and load data and object models from an xmi file, map the object model to the data model using drag-and-drop capabilities, add caCORE SDK-required tags and tag values into the xmi file, and generate Hibernate mapping files.

caAdapter integrates with NCI CBIIT cancer Common Ontologic Representation Environment (caCORE) components (http://ncicb.nci.nih.gov/NCICB/infrastructure). See the caCORE Technical Guide (ftp://ftp1.nci.nih.gov/pub/cacore) and the caCORE Software Development Kit Programmer's Guide (ftp://ftp1.nci.nih.gov/pub/cacore/SDK) for more information.

caAdapter Core Engine Architecture

Figure 1.1 illustrates the caAdapter core engine architecture design including its subsystems and components.

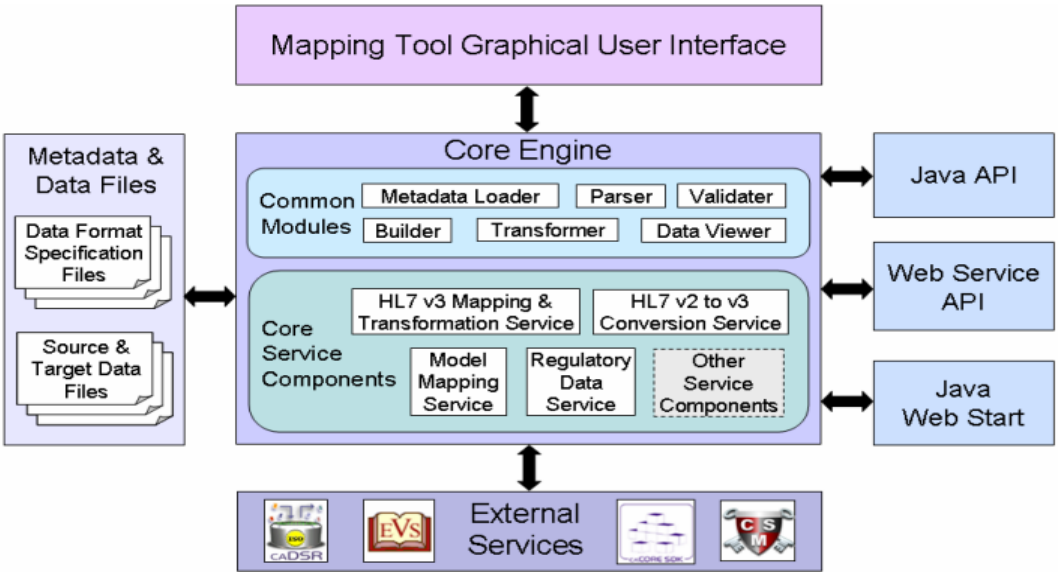

Figure 1.1 caAdapter Core Engine Architecture

The main features of the caAdapter core engine are

- Metadata Loader - loads metadata information for HL7 v2 and v3, CSV, and XMI files
- Parser - parses HL7 v2 and v3 messages, CSV, and XMI files
- Builder - builds HL7 v3 messages and SDTM datasets
- Transformer - transforms source to target data based on mapping specifications
- Data Viewer - displays transformed data in the UI

- Validater - validates schema files and mapping specifications

caAdapter Mapping Tool

The caAdapter Mapping Tool is a graphical application for mapping elements between data structures elements or model structures elements.

The mapping tool provides the following:

- Source and Target Specification - graphical interface for defining input and output formats.
- User Interface - simple mechanism for mapping source to target elements containing tree structure, drag-and-drop functionality, and functions and property definitions.
- Mapping Functions - capability to do simple source data manipulation.
- Transformation Service - generation of HL7 v3 XML message instances and SDTM text files from a source database based on user-defined mapping specifications.
- Validation - capability to validate structure and mapping specifications.

About HL7

Health Level Seven (HL7) (http://www.hl7.org/) is one of several American National Standards Institute (ANSI)-accredited Standards Developing Organizations (SDOs) operating in the health care arena. HL7 provides standards for data exchange to allow interoperability between health care information systems. It focuses on the clinical and administrative data domains. The standards for these domains are built by consensus by volunteers—providers, payers, vendors, government—who are members in the not-for-profit HL7 organization.

HL7 version 2 (v2) is a messaging standard that focuses on syntactic data interchange. HL7 messaging (v2 or higher) has been recommended as a data exchange standard by the e-Government initiative. In fact, various releases of this version are in use in over 90% of U.S. hospitals, and v2 is considered the most widely implemented standard for healthcare information in the world. However, since it lacks an explicit methodology, conformance rules, and grouping of messages, it cannot be considered an interoperability standard.

HL7 v2 messages are composed of segments (individual lines in a message) which are composed of fields (data values) which may in turn be composed of components and sub-components. Several different delimiters or field separators are used to mark boundaries between the various elements. Specifications for messages using these structures are published in a text document format which does not easily lend itself to being computable. Furthermore, messages are often customized at local sites making it difficult to share messages between sites. caAdapter consequently includes a computable version of the message specifications which can be tailored to suit the needs of cancer centers and hospitals.

HL7 as an organization aimed to address some of the problems of HL7 v2 in its next major version, version 3 (v3). The key goal of the HL7 community is syntactic and

semantic interoperability. This goal is supported in HL7 v3 by what are commonly called the four pillars of semantic interoperability:

1. A common Reference Information Model (RIM) spanning the entire clinical, administrative, and financial healthcare universe. The RIM is the cornerstone of the HL7 v3 development process. An object model created as part of the v3 methodology, the RIM is a large pictorial representation of the clinical data domains and identifies the life cycle of events that a message or groups of related messages will carry. It is a shared model between all the domains and is the model from which all domains create their messages. Explicitly representing the connections that exist between the information carried in the fields of HL7 messages, the RIM is essential to HL7's ongoing mission of increasing precision and reducing implementation costs.
2. A well-defined and tool-supported process for deriving data exchange specifications from the RIM. HL7 has defined a methodology and process for developing specifications, artifacts to document the models and specifications, tools to generate the artifacts and an organization for governing the overall process of standards development. Such structure avoids ambiguity common to many existing standards.
3. A formal and robust data type specification upon which to ground the RIM. Data types are the basic building blocks of attributes. They define the structural format of the data carried in the attribute and influence the set of allowable values an attribute may assume. HL7 defines an extensive set of complex data types which provide the structure and semantics needed to describe data in the healthcare arena.
4. A formal methodology for binding concept-based terminologies to RIM attributes. Within HL7, a vocabulary domain is the set of all concepts that can be taken as valid values in an instance of a coded field or attribute. HL7 has defined vocabulary domains for some attributes to support use of the RIM in messages. It also provides the ability to use, document, and translate externally coded vocabularies in HL7 messages.

The specifications that are developed upon this foundation are documented in a progressive set of artifacts that represent varying levels of abstraction of the domain data. The artifacts go from purely abstract and universal in scope to implementation-specific and very narrow in subject matter:

- The RIM is the foundational Unified Modeling Language (UML) class diagram representing the universe of all healthcare data that may be exchanged between systems.
- A Domain Message Information Model (DMIM) is a subset of the RIM that includes RIM class clones, attributes, and associations that can be used to create messages for a particular domain (a particular area of interest in healthcare). DMIMs use HL7 modeling notation, terminology, and conventions.
- A Refined Message Information Model (RMIM) is a subset of a DMIM that is used to express the information content for an individual message or set of messages with annotations and refinements that are message specific.

- A Model Interchange Format (MIF) is an XML representation of the information contained in an HL7 specification, and is the format that all HL7 v3 specification authoring and manipulation tools will be expected to use.
- A Message Type (MT) is the specification of an individual message in a specific implementation technology.

The caAdapter APIs make use of the MIF and MT artifacts. While the HL7 standard is not implementation specific, caAdapter uses XML as its implementation technology.

The NCI CBIIT provides training resources to assist the caBIG community and other interested parties in implementing HL7 v3 messaging. These resources include online tutorials, self-paced training, and links to HL7 resources (http://ncicb.nci.nih.gov/infrastructure/cacore_overview/caadapter/indexContent/HL7_Tutorial).

About the Study Data Tabulation Model (SDTM)

The Study Data Tabulation Model, or SDTM, is a set of standards developed by the Clinical Data Interchange Standards Consortium (CDISC). It provides structured guidelines for submitting study data tabulations to a regulatory authority such as the United States Food and Drug Administration (FDA).

SDTM datasets are organized by *domains*, where each domain contains a list of *variables*. Each domain is identified by a two-letter acronym. An eight-character naming convention is used to refer to variables within a domain. An example domain is *Demographics*, which is referred to by the acronym *DM*. The Demographics dataset contains variables such as patient name, patient date of birth, race, and sex.

Domains are grouped into *classes*. Domain classes include the following:

- Trial Design
- Interventions
- Events
- Findings, and,
- Special Purpose

Figure 1.2 lists SDTM Domain Classes and associated Domains.

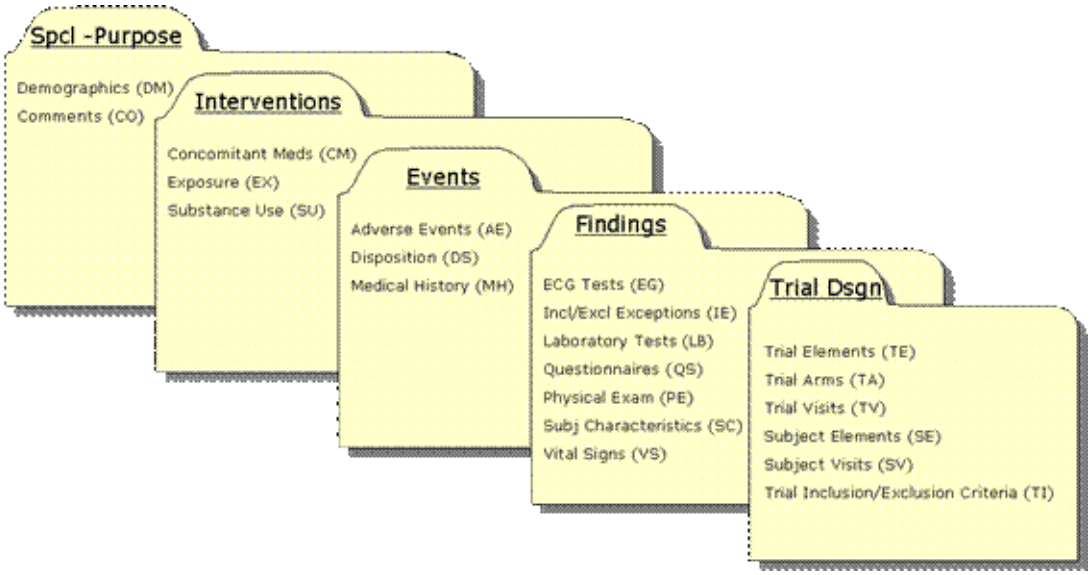

Figure 1.2 SDTM Domain Classes

SDTM dataset structures are fully defined in the guide "Study Data Tabulation Model Implementation Guide: Human Trials," which is available from the CDISD web site at (http://www.cdisc.org). Furthermore, SDTM datasets are defined in an XML document often referred to as the `define.xml`. CDISC provides a sample `define.xml` document which was used in the implementation of the CSV to SDTM Mapping capability of caAdapter.

About the Object and Data Model

The caAdapter Model Mapping Service takes advantage of caAdapter's mapping infrastructure to facilitate object to data model mapping. The model mapping service requires an `.xmi` file (with full Enterprise Architect (EA), or ArgoUML, round-trip capability) that includes a data and an object model. It loads both models into the tool and gives the user the ability to map object elements to a table elements using drag-and-drop capability. It also allows the user to map the relationships between objects to the relationships between tables. Once the mapping is completed, caAdapter adds all caCORE SDK-required tagged values into the `.xmi` file. It also creates and saves the information to a `.map` file for backwards compatibility. This module also has the capability to create Hibernate files of the mapping.

This guide focuses on using this feature of caAdapter.

Prerequisites for Using the caAdapter Mapping Tool

Successful use of the caAdapter mapping tool requires the following prerequisites:

- Thorough familiarity with object and data models
- Training on the caAdapter Mapping Tool
- Familiarity with this document

Resources for Installing caAdapter MMS Tool

Complete instructions for installing caAdapter MMS are located in the caAdapter MMS Installation Guide at http://ncicb.nci.nih.gov/infrastructure/cacore_overview/caadapter/indexContent/docs/caAdapter_Documentation/.

Starting the caAdapter MMS Tool

Starting the caAdapter MMS Tool from the Binary Distribution

To launch the caAdapter MMS Tool GUI, follow these steps:

1. In a Command Prompt window, enter `cd {home directory}` to go to your home directory (Windows example: `C:\caadapter`).
2. Enter `java -jar caadapter_ui.jar`.

   The Welcome to the caAdapter screen momentarily appears, followed by the caAdapter GUI.

Starting the caAdapter MMS Tool from the Source Distribution

To launch the caAdapter MMS Tool, follow these steps:

1. In a command prompt window, enter `cd {home directory}` to go to your caAdapter home directory (Windows example: `C:\caadapter`).
2. Enter `cd ..\caadaper`
3. Enter `ant all`.
4. Enter `ant launchui`.

   The Welcome to the caAdapter screen momentarily appears, followed by the caAdapter Mapping Tool GUI.

Starting the caAdapter MMS Tool from the Windows Distribution

To launch the caAdapter MMS Tool, select **caAdapter** from the Start menu.

Starting the Mapping Tool on the Web (WebStart)

You can also use caAdapter MMS in your web browser by entering the following URL: http://caadapter.nci.nih.gov/caadapter-mms/caadapter-mms.jnlp.

CHAPTER

2

USING THE CAADAPTER MMS TOOL

This chapter describes how to use caAdapter MMS Tool to facilitate object to data model mapping.

Topics in this chapter include:

About the caAdapter Model Mapping Service

The caAdapter Model Mapping Service takes advantage of caAdapter's mapping infrastructure to facilitate object to data model mapping. The model mapping service requires an `.XMI` file (with full EA, or ArgoUML, round-trip capability) that includes data and object models. The service module loads both models into the tool. Object and data model elements can be mapped using drag-and-drop capability. Once mapping is complete, caAdapter adds SDK-required tag values to the original `.XMI` file (and, for backwards compatibility, it also creates a `.map` file). At this point, the caCORE SDK can use the annotated `.XMI` file to perform code generation tasks.

Mapping Tool Operational Scenario for Model Mapping Service

In order to generate silver-level compliant software, caCORE SDK developers must first develop an object model and a data model in EA, or ArgoUML. Second, the developers must add tag values to associate objects to tables, attributes to columns, and object relationships to table relationships. Defining the associations manually can be error prone and very time consuming. The caAdapter model mapping service component automates this process. The user can map objects/attributes to tables/columns using drag-and-drop capability. Next, caAdapter will automatically add tag values to the original `.XMI` file. The user can then use the updated xmi file for caCORE SDK code generation.

Users who are not using the caCORE SDK but want to map an object model to a data model can use the MMS Tool to perform the mapping, and create Hibernate files which capture mapping information.

Using the caAdapter Model Mapping Service

The caAdapter Model Mapping Service provides the following functionalities:

- Parse and load data model and object model from an XMI file
- Map object model to data model elements using drag-and-drop capability. This includes mapping objects to tables, object attributes to table columns, and object relationships to table relationships
- Add caCORE SDK required tags and tag values into the XMI file
- Generate Hibernate mapping files

The process flow for integrating the caAdapter Model Mapping Service with other components follows these steps, assuming, for example, EA was used as the UML tool:

1. Develop an object model and a data model in EA.
2. Export an XMI file from EA so that the caAdapter Model Mapping Service can load the XMI file.
3. Map objects to tables and map attributes and associations to columns by dragging and dropping.

As illustrated in *Figure 2.1*, this process enables caAdapter to directly generate a set of Hibernate HBM mapping files. Alternatively, the original XMI file that has caCORE SDK-compliant tag values can be saved for later use.

.

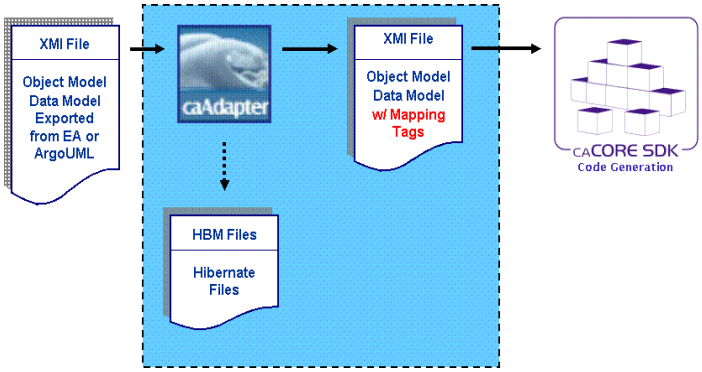

Figure 2.1 caAdapter Model Mapping Service - Overall Process

Note: While it still supports the `.map` file, this version of caAdapter does not require the file. All mapping information is now stored in the `.xmi` file. The `.map` file is created for backward compatibility purpose.

The following subsections describe each of the mapping process steps in detail.

Generating an XMI File from EA

Before beginning to map between an object model and a data model through the caAdapter Model Mapping Service, you need to generate an XMI file from EA.

To generate the file, follow these steps in the EA application:

1. Open the `.eap` file (*i.e.*, the file that contains the object and data models).
2. In the Project View pane, right-click **Logical View**.
3. On the shortcut menu, select the following commands:

   Import/Export > **Export package to XMI file** (*Figure 2.2*).

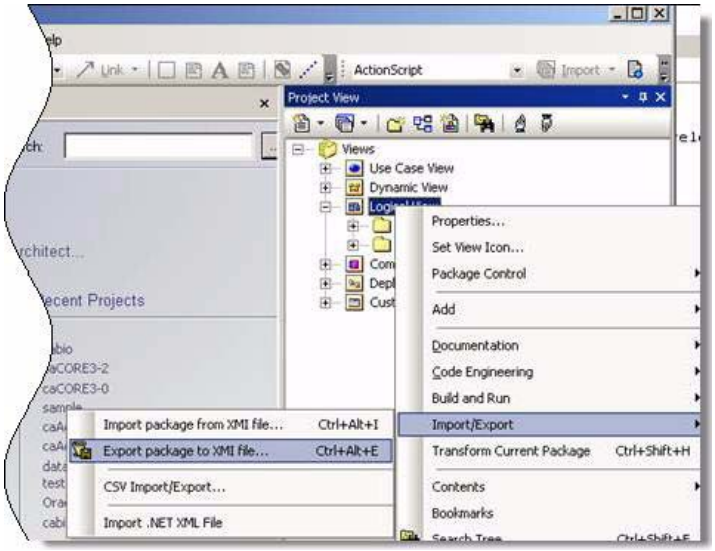

Figure 2.2 Exporting an XMI File from EA

4. In the Export Package to XMI window, follow these steps:
   a. In the **Filename** field, specify the output file name of the XMI file.
   b. Check the following boxes:
      — **Format XMI Output**
      — **Enable Full EA Roundtrip**
   c. Click **Export**.

   The generated XMI file can now be parsed by the caAdapter Model Mapping Service module (*Figure 2.3*).

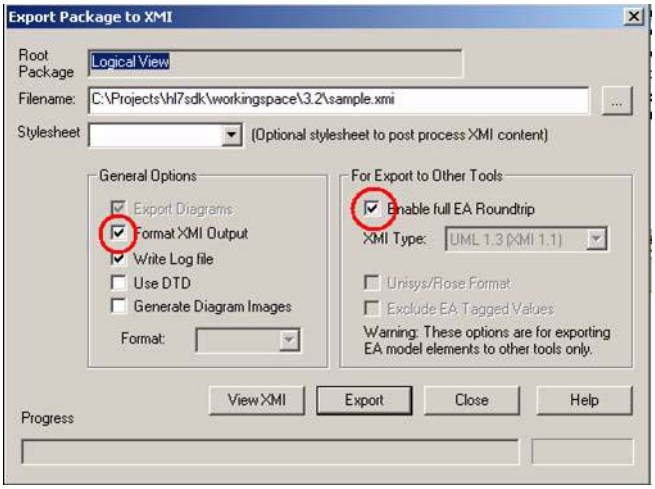

Figure 2.3 Options to Export XMI File from EA

Creating an Object Model to Data Model Map Specifications

Perform the following steps to create a new map specification.

1. Select **File** > **New** > **Model Mapping Service** > **Object Model to Data Model Map Specification** (*Figure 2.4*) to open a new mapping tab with empty source and destination panels.
2. Click **Open XMI file…** to display the Open XMI file … dialog box (*Figure 2.5*). Select the XMI file to start mapping an object model to a data model.

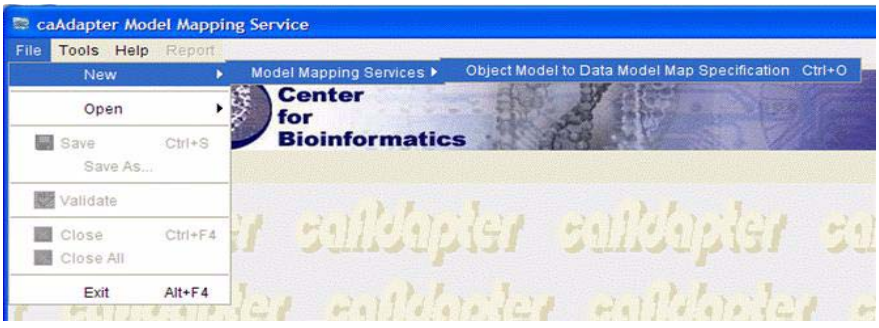

Figure 2.4 Creating an Object Model to Data Model Map Specification

3. After the XMI file is loaded, caAdapter displays the object model in the left panel, and the data model in the right panel. Start mapping objects and attributes to tables and columns.

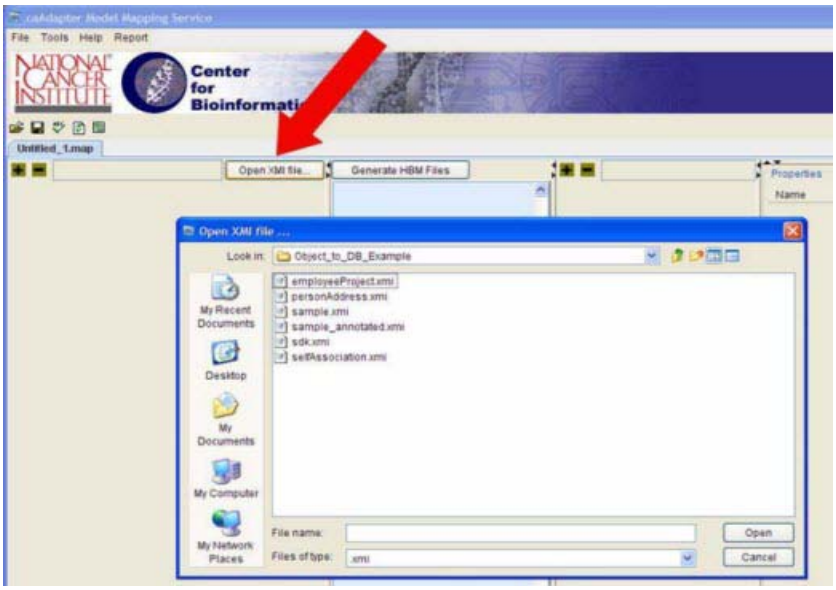

Figure 2.5 Open XMI File

Opening an Existing Object to Data Model Mapping Specification

Perform the following steps to open an existing map specification.

1. Select **File** > **Open** > **Object Model to Data Model Map Specification**. The Open Map File dialog box displays.
2. Select either the XMI file or the map specification file and click Open. (For backwards compatibility, this version of caAdapter saves the mappings in the `.map` file as well as the XMI file, so either may be used to open the mapping specifications).
3. If you select a `.map` file and the XMI associated with the mapping cannot be found, the Select XMI file dialog box opens. Browse to the correct XMI file and click **Open**.

Basic Mapping

Perform the following steps to create an object to data model mapping specification (dependency mapping, attribute mapping, and association mapping).

1. Select a source element (object, attribute, or association) from the object model and drag it to the appropriate target element (table, column or foreign key) in the data model. The cursor indicates whether the source element is, or is not, allowed to be mapped to the target element (*2.*). Drop the source element on the target element.
2. Once a source element is mapped to a target element, a mapping line appears between them in the mapping panel. *Figure 2.6* shows a mapping line between `Amendment` in the object model, on the left, and `AMENDMENT` in the data model, on the right.

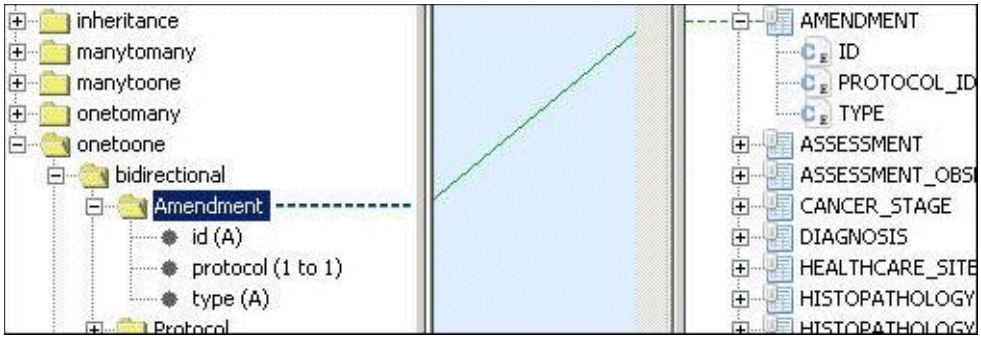

Figure 2.6 Mapping line between source element and target element

Dependency Mapping (Object to Table)

A dependency mapping is a mapping between an object and a table. Perform the following steps to create a dependency mapping.

1. Select a source element from the object model on the right. The example in *Figure 2.7* shows `HealthcareSite`. Click and drag to `HealthcareSite` to `HEALTHCARE_SITE` in the data model.
2. A mapping line between `HealthcareSite` in the object model and `HEALTHCARE_SITE` in the data model should now be visible. Dependency mapping lines are color-coded green.

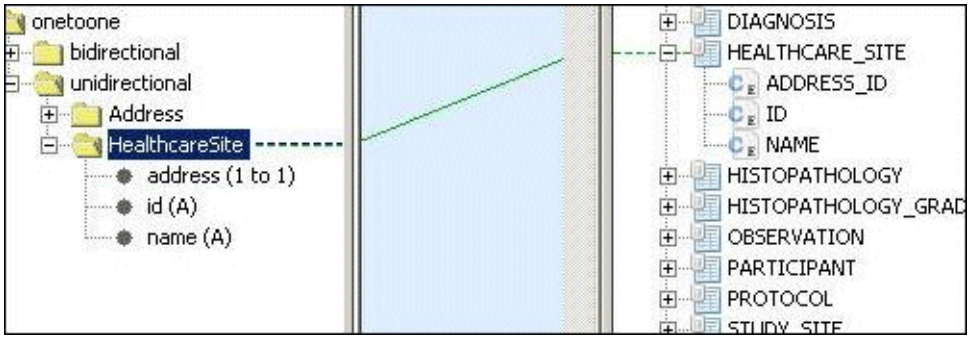

Figure 2.7 Dependency Mapping

Attribute Mapping

An attribute mapping is a mapping between an attribute in the object model and a column in the data model. (Before any attribute mapping can be performed, the user has to complete dependency mapping first). Perform the following steps to create an attribute mapping.

1. The example in *Figure 2.8* shows the attribute `id (A)` for the class `HealthcareSite`. Select '`id (A)`' in the object model and drag it to ID in the data model.
2. A mapping line should be visible between the attribute and column. Attribute mapping lines are color-coded blue. Repeat this for '`name (A)`' to `NAME`.

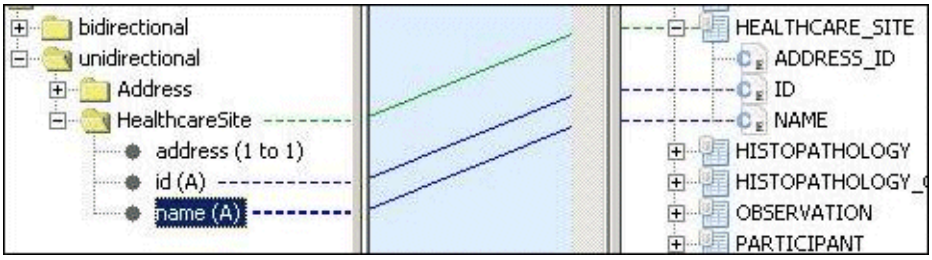

Figure 2.8 Attribute Mapping

3. If the object has not already been mapped to the table, an attempt to map object's attributes to the table's columns will result in an error message (*Figure 2.9*).

Figure 2.9 Attribute Mapping error message

Association Mapping

An association mapping is a mapping between one end of an association listed under an object in the object model and a foreign key column in a table in the data model. Perform the following steps to create an association mapping.

1. First create a dependency mapping between the object and the table. For example, in *Figure 2.10*, the green line shows a dependency between '`HealthcareSite`' and '`HEALTHCARE_SITE`'.
2. Map '`id (A)`' to ID and '`name (A)`' to `NAME`.

3. Click and drag '`address (1 to 1)`' to `ADDRESS_ID`. When complete, the final result should look like *Figure 2.10*. Association mapping lines are color-coded red.

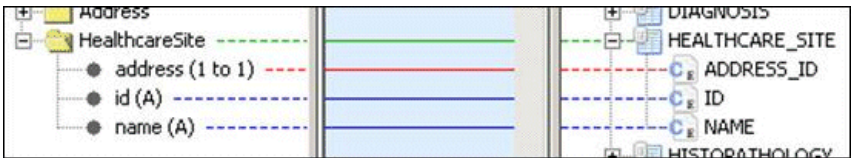

Figure 2.10 Association Mapping

Deleting Mapping Lines

Perform the following steps to delete a mapping line.

1. Select the mapping line by left clicking on it in the mapping panel. The line is highlighted.
2. Right click on the highlighted mapping line and select **Delete** (*Figure 2.11*). The line is removed from the mapping panel.

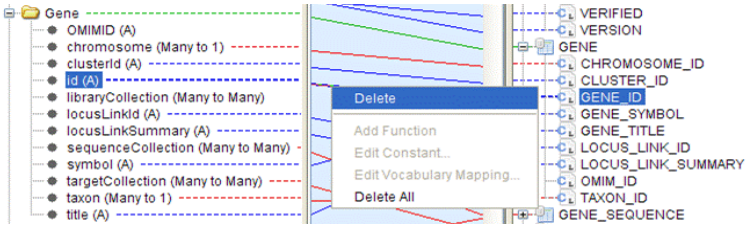

Figure 2.11 Deleting Mapping Lines

Validating Mapping Specifications

Validating a mapping specification identifies any pertinent business rules that have been violated and indicates any changes that need to be made. Perform the following steps to validate the object to data model mapping specification.

1. Click the `Validate` button (top of *Figure 2.12*). The following message displays: `Validation process completed successfully with no message received`. If there are errors in the validation process, the following message displays: `Validation process completed but received <some number> ERRORs`.
2. If there are errors the Message Dialog (bottom of *Figure 2.12*) window opens and allows examination of any messages, errors, or warnings. Error messages may identify what actions to perform to correct errors, while warnings and

informational messages may require no changes at all. It is recommended that mappings be re-validated after changes are made.

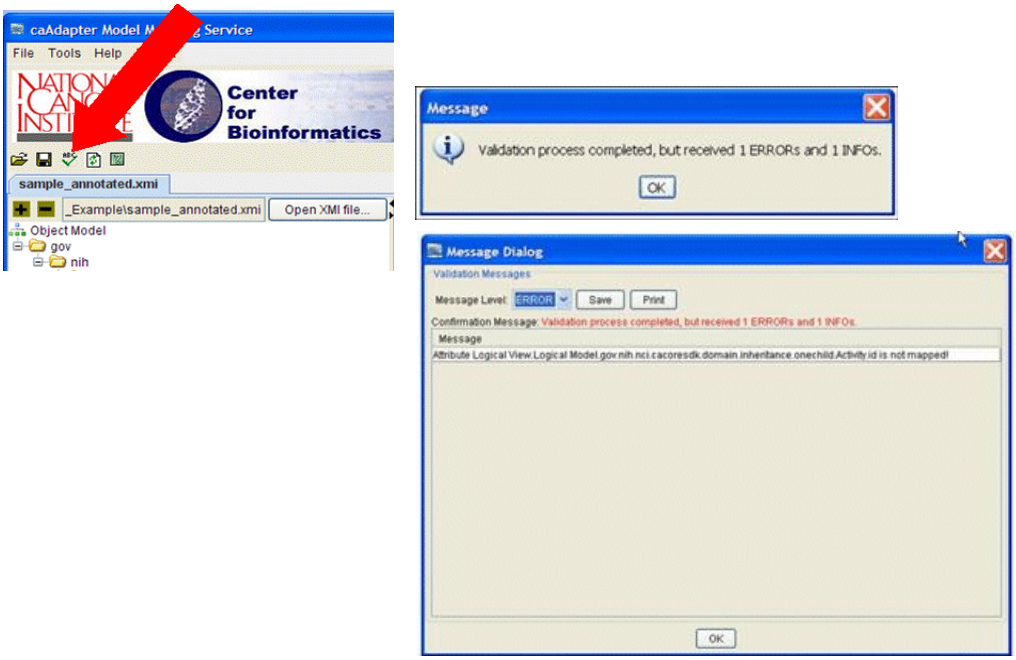

Figure 2.12 Validate Mapping Specification

Saving Mapping Specifications

To save a mapping specification, select **File** > **Save**. caAdapter saves the mappings to the XMI file and to the `.map` file (for backward compatibility). The Save Complete dialog displays when completed.

The XMI file created can be used by the caCORE SDK for code generation purposes.

Generating Hibernate Files

An alternative to creating caCORE SDK APIs is to generate Hibernate files and use those files in an application to access data from a database. Perform the following steps to generate Hibernate files from the current object to data model mapping.

1. Click the **Generate HBM Files** button; the Open dialog box displays (*Figure 2.13*).
2. Select a directory to save the HBM file(s) and click **Open**.
3. The HBM files are saved to the directory specified.

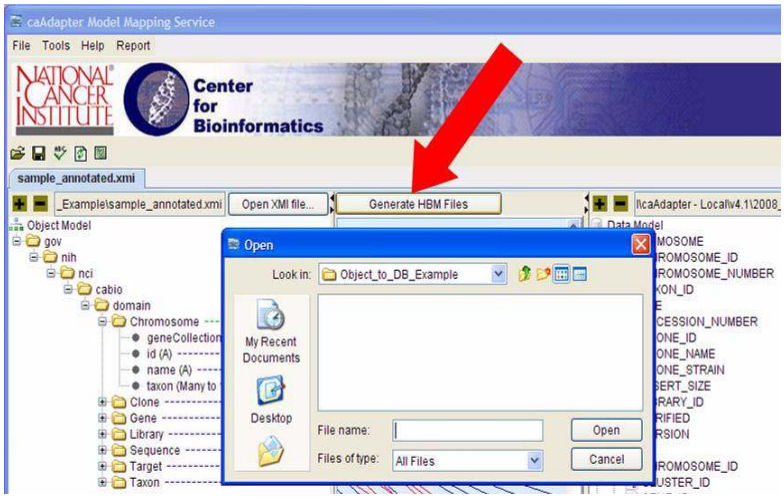

Figure 2.13 Generate HBM Files

The Seven Mapping Scenarios

Before performing any of the following mapping scenarios, all dependency mappings between objects and tables have to be completed.

One-to-One Bi-Directional

To map one-to-one bi-directional relationships, the rule is to map the association from the object whose corresponding table has the foreign key (source) to the foreign key in that table (target). In the example (Protocol and Amendment) in *Figure 2.14*, drag the association (`Amendment.protocol (1 to 1)`) and drop it onto the foreign key (`PROTOCOL_ID`) of the corresponding table (`AMENDMENT`). For one-to-one bi-directional mapping, only one end of the relationship needs to be mapped; the other end (`Protocol.amendment (1 to 1)`) does not need to be mapped.

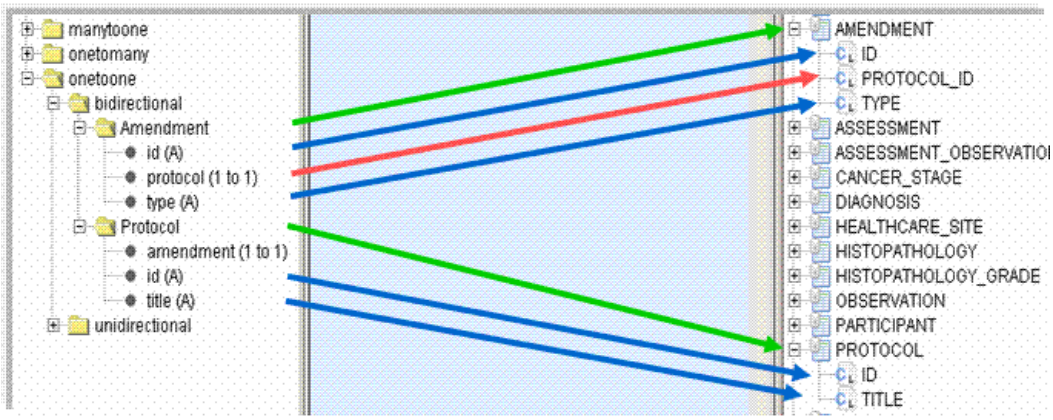

Figure 2.14 One-to-One Bi-Directional Mapping

One-to-One Uni-Directional

To map one-to-one uni-directional relationships, the rule is to map the association from the object whose corresponding table has the foreign key (source) to the foreign key in that table (target). In the example (`HealthcareSite` and `Address`) in *Figure 2.15*, drag the association (`HealthcareSite.address (1 to 1)`) and drop it onto the foreign key (`ADDRESS_ID`) of the corresponding table (`HEALTHCARE_SITE`).

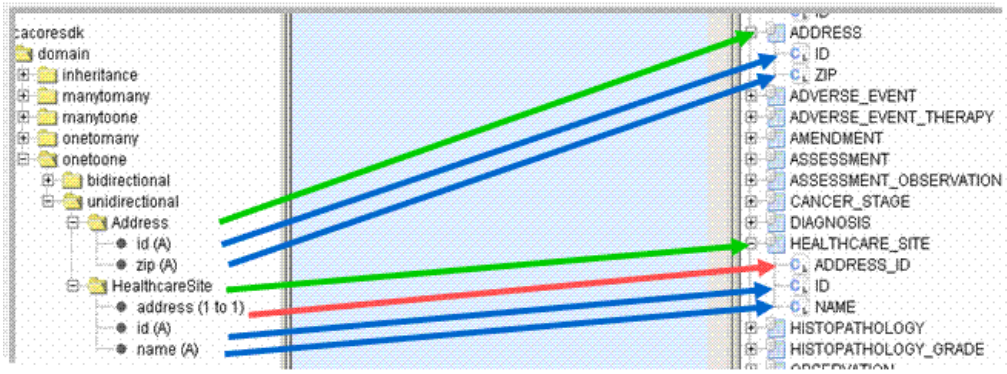

Figure 2.15 One-to-One Uni-Directional Mapping

One-to-Many / Many-to-One Bi-Directional

To map one-to-many, or many-to-one, bi-directional relationships, the rule is to map the association from the object whose corresponding table has the foreign key (source) to the foreign key in the corresponding table (target). In the example (`AdverseEvent` and `AdverseEventTherapy`) in *Figure 2.16*, drag the association (`AdverseEventTherapy.adverseEvent (Many to 1)`) and drop it onto the foreign key (`ADVERSE_EVENT_ID`) of the corresponding table (`ADVERSE_EVENT_THERAPY`). For one-to-many bi-directional mapping, the other end of the association is not required to be mapped through caAdapter.

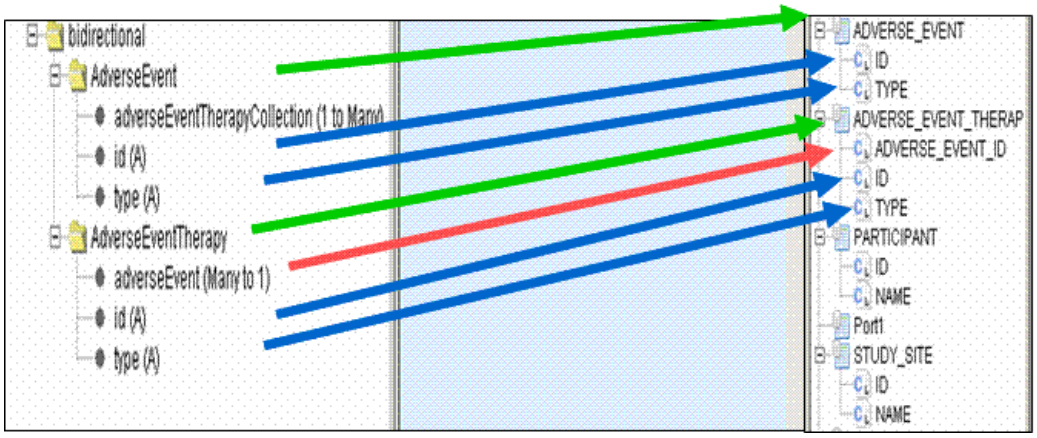

Figure 2.16 One-to-Many Bi-Directional Mapping

One-to-Many / Many-to-One Uni-Directional

To map one-to-many, or many-to-one, uni-directional relationships, the rule is to map the association from the object whose corresponding table has the foreign key (source) to the foreign key in the corresponding table (target). In the example (`Histopathology` and `HistopathologyGrade`) in *Figure 2.17* , drag the association (`HistopathologyGrade.histopathology (Many to 1)`) and drop it onto the foreign key (`HISTOPATHOLOGY_ID`) of the corresponding table (`HISTOPATHOLOGY_GRADE`).

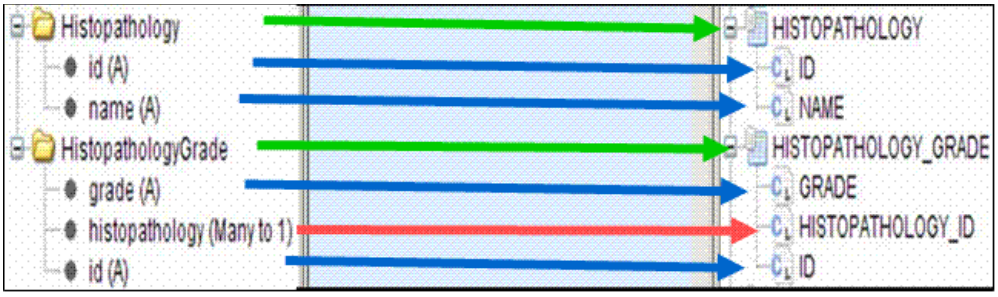

Figure 2.17 One-to-Many Uni-Directional Mapping

Many-to-Many Bi-Directional

To map a many-to-many bi-directional association, first identify the intersection table. In the example in *Figure 2.18*, `STUDY_SITE_PARTICIPANT` is the intersection table (typically the name of the intersection table is a concatenation of the two tables that correspond to the two objects). Then, drag both ends of the associations and drop them onto the two corresponding columns in the intersection table.

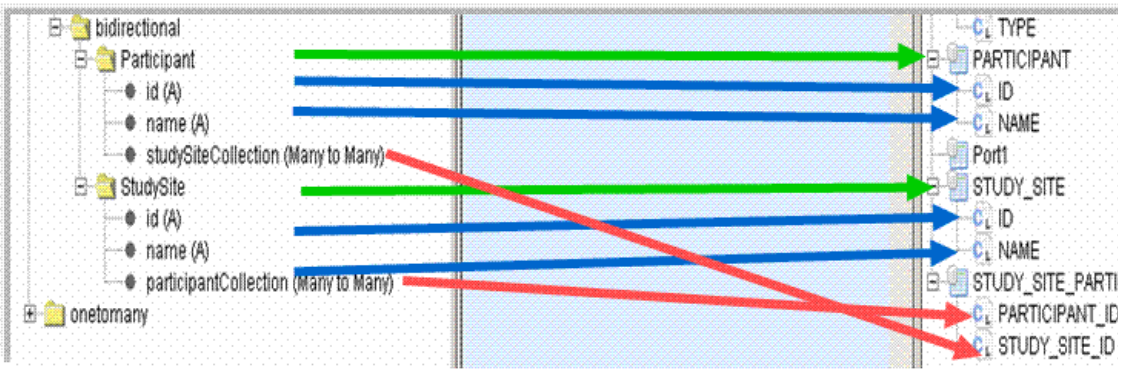

Figure 2.18 Many-to-Many Bi-Directional Mapping

Many-to-Many Uni-Directional

To map a many-to-many uni-directional association, first identify the intersection table. In the example in *Figure 2.19*, `ASSESSMENT_OBSERVATION` is the intersection table (typically the name of the intersection table is a concatenation of the two tables that correspond to the two objects). Then, drag the association (`Assessment.ObservationCollection (Many to Many)`) and drop it onto the corresponding column (`OBSERVATION_ID`)) in the intersection table (`ASSESSMENT_OBSERVATION`).

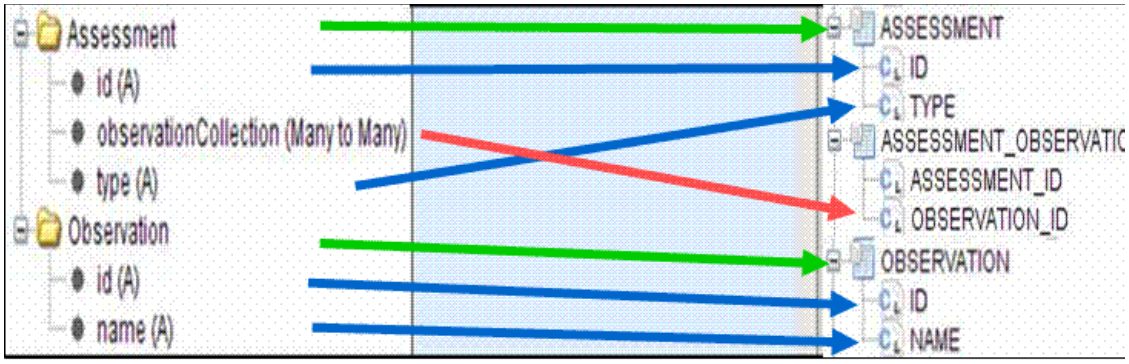

Figure 2.19 Many-to-Many Uni-Directional Mapping

Mapping Inheritance

To map inheritance through the caAdapter Model Mapping Service, follow one of the steps described in *The Seven Mapping Scenarios* on page 23 to map each child class or attribute to its corresponding table or column. The tool automatically marks inherited attributes as (`A – Derived`). Those attributes do not need to be mapped. During the validation, an information level message appears.

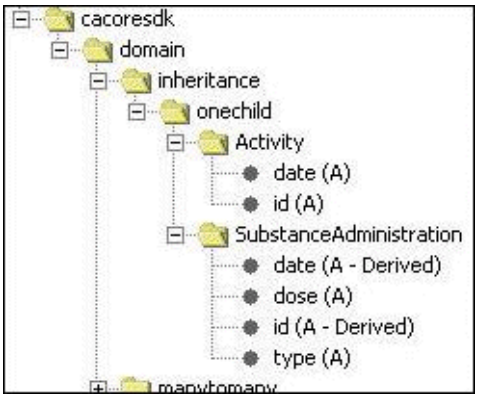

Figure 2.20 Mapping with Inheritance

User Interface Legend

Node Details

- `(A)` – The node is an attribute
- `(A – Derived)` –The node is an inherited attribute
- `(1 to 1)` – The node is a one-to-one association
- `(1 to Many)` – The node is a-one-to many association
- `(Many to 1)` – The node is a many-to-one association
- `(Many to Many)` – The node is a many-to-many association

Mapping Line Colors

- Green – Dependency Mapping
- Blue – Attribute Mapping
- Red – Association Mapping

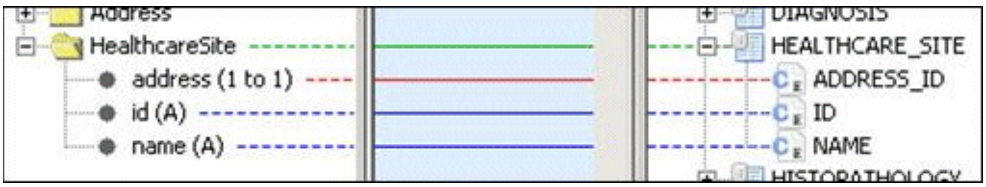

Figure 2.21 Mapping Line Colors

The following icons are used to indicate various element tagging as discussed in *Additional Module Features*, below:

Lazy	
CLOB	
Eager	
Discriminator	
Primary Key	

Additional Module Features

To tag a column in the data model as Lazy-Load, CLOB, or Discriminator, right-click on the column and select the appropriate tag (*Figure 2.22*).

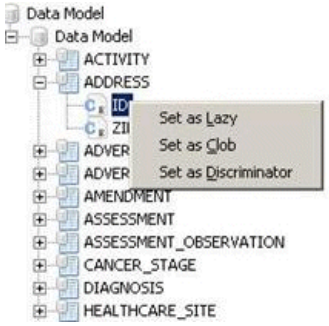

Figure 2.22 Tagging a Column with Lazy, CLOB, or Discriminator

Only one attribute can be tagged in an object as a Primary Key. Right click on the attribute and select **Make Primary Key**. If no primary key was specified, caCORE SDK will assume that the field with the name “id” is the primary key (*Figure 2.23*).

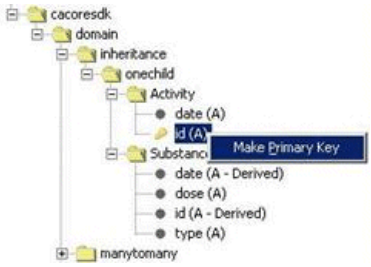

Figure 2.23 Designating a Primary Key for an Object

Specify the prefix to use for object or data model elements using the Tools > Preferences menu option (*Figure 2.24*).

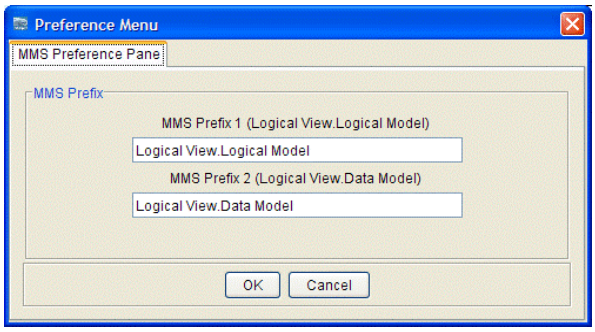

Figure 2.24 Designating Prefixes

The following table show various tags implemented by caAdapter.

Tag Name	***Tag Value***	***Location***
id-attribute	Fully qualified class name	Class attribute
mapped-attributes	Fully qualified attribute name	Table column
implements-association	Fully qualified association name	Table column (foreign key)
inverse-of	Fully qualified association name	Table column (foreign key)
discriminator	Fully qualified class name (when used on the column), Discriminating value (when used on the class)	Table column (foreign key), Class
correlation-table	Join table name	Association
documentation	Comments on the class or attribute	Class or Attribute of class
description	Comments on the class or attribute	Class or Attribute of class
Lazy-load	Yes/No	Association
Type	CLOB	Table column. Corresponding class attribute must be String type)

Table 2.1 caAdapter Implemented Tags

Example Data Files

Example data are included in the caAdapter MMS Tool distribution. You can use the example data to become acquainted with the mapping tool before using your own data. Example data are located at the `{home directory}\workingspace\`.

APPENDIX A
REFERENCES

Articles

- Java Programming: http://java.sun.com/learning/new2java/index.html
- Extensible Markup Language: http://www.w3.org/TR/REC-xml/
- XML Metadata Interchange: http://www.omg.org/technology/documents/formal/xmi.htm

caBIG Material

- caBIG: http://cabig.nci.nih.gov/
- caBIG Compatibility Guidelines: http://cabig.nci.nih.gov/guidelines_documentation

caCORE Material

- NCI CBIIT: http://ncicb.nci.nih.gov
- caCORE: http://ncicb.nci.nih.gov/core
- caBIO: http://ncicb.nci.nih.gov/core/caBIO
- caDSR: http://ncicb.nci.nih.gov/core/caDSR

Software Products

- Java: http://java.sun.com
- Ant: http://ant.apache.org/

caAdapter Glossary

Acronyms, objects, tools and other terms related to the caAdapter MMS Tool are described in this glossary.

Term	***Definition***
caCORE SDK	cancer Common Ontologic Representation Environment Software Development Kit
CLOB	Character large object
EA	Enterprise Architect
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

INDEX

T

U

V

X