

CAADAPTER 1.3

User's Guide



NATIONAL[®]
CANCER
INSTITUTE

Center for Bioinformatics

CREDITS AND RESOURCES

caAdapter Development and Management Teams		
Development	User's Guide	Program Management
Eric Chen ¹	Eric Chen ¹	Anand Basu ²
Scott Jiang ¹	Scott Jiang ¹	Christo Andonyadis ²
Ki Sung Um ²	Wendy Ver Hoef ³	Steven Liu ²
Ye Wu ¹	Charles Yaghmour ³	Sharon Settnek ¹
Wendy Ver Hoef ³		Smita Hastak ³
¹ Science Application International Corporation (SAIC)		³ ScenPro, Inc.
² National Cancer Institute Center for Bioinformatics (NCICB)		

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

<i>LISTSERV facilities pertinent to the caAdapter</i>		
<i>LISTSERV</i>	<i>URL</i>	<i>Name</i>
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
NCIEVS-L Listserv	https://list.nih.gov/archives/ncievs-l.html	NCI Vocabulary Services Information

TABLE OF CONTENTS

Chapter 1

Using the caAdapter User's Guide	1
Audience for the caAdapter User's Guide	1
Recommended Reading	1
Document Text Conventions	2
Organization of this Guide	2

Chapter 2

Overview of HL7 and caAdapter	5
HL7 Overview	5
caAdapter Overview	7
caAdapter Within the Overall NCICB Clinical Trials Architecture	7
caAdapter Core Engine Architecture	9
caAdapter Mapping Tool Architecture	9
caAdapter Installation	10

Chapter 3

Using caAdapter	11
caAdapter API Process Flow	11
caAdapter API Operational Scenario	14
caAdapter Mapping Tool Operational Scenario	14

Chapter 4

Using the caAdapter APIs	17
caAdapter Directory Structure	17
caAdapter APIs	18
Meta Data Loader	18
HL7 v3 Message Parser	18
HL7 v3 Message Builder	19
Transformation Service	19
Vocabulary Validation	20

caAdapter API Error Logs	21
Chapter 5	
Using the caAdapter Mapping Tool	23
Prerequisites for Using the caAdapter Mapping Tool	23
Starting the caAdapter Mapping Tool	23
Starting the Mapping Tool from the Binary Distribution	23
Starting the Mapping Tool from the Source Distribution	24
Starting the Mapping Tool from the Windows Distribution	24
caAdapter Mapping Tool Process Flow	24
caAdapter Mapping Tool Common Features	25
caAdapter Mapping Tool Interface	25
caAdapter Mapping Tool Validation	28
Source Specification	30
Segmented CSV Specification	30
Target Specification	35
HL7 v3 Specification	35
Map Specification	52
Business Rules	52
Step-by-Step Instructions	53
HL7 v3 Message	60
Business Rules	60
Step-by-Step Instructions	60
Chapter 6	
caAdapter File Types	65
caAdapter File Formats and Locations	65
CSV Data File	66
CSV Specification	66
HL7 v3 Specification	68
Function Specification	69
Function Specification Overview	69
Adding Functions to the Function Library	70
Map Specification	70
HL7 v3 Message	71
Appendix A	
caAdapter Example Data	73
Appendix B	
References	75
Technical Manuals/Articles	75

caBIG Material	75
caCORE Material	75
HL7 Concepts and Material	75
Software Products	76
Appendix C	
caAdapter Glossary	77
Index	83

CHAPTER 1

USING THE CAADAPTER USER'S GUIDE

This chapter introduces you to the *caAdapter User's Guide*.

Topics in this chapter include:

- *Audience for the caAdapter User's Guide* on this page
- *Recommended Reading* on this page
- *Document Text Conventions* on page 2
- *Organization of this Guide* on page 2

Audience for the caAdapter User's Guide

The *caAdapter User's Guide* is the companion documentation to caAdapter. This guide includes information and instructions for using caAdapter which consists of two components: a set of Application Programming Interfaces (APIs) and a mapping tool graphical user interface (GUI). See [Chapter 2](#) for an overview of caAdapter. The technical audience (Java programmers, system architects, etc.) use this guide to utilize the major caAdapter Application Programming Interfaces (APIs) to parse, build and validate Health Level Seven (HL7) version 3 (v3) messages. Analysts (HL7 analysts, database administrators, business analysts, etc.) use this guide to follow the step-by-step procedures to create v3 XML message instances using the GUI.

This guide assumes that the reader is familiar with HL7 terms and processes and only provides a brief overview of HL7 in [Chapter 2](#). Prerequisites for using the caAdapter Mapping Tool are included in [Chapter 5](#).

Recommended Reading

Following is a list of recommended reading materials and resources which can be useful for familiarizing oneself with concepts contained within this guide.

- HL7: <http://www.hl7.org>

- National Cancer Institute Center for Bioinformatics (NCICB) HL7 Tutorial: http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter/HL7_Tutorial

Uniform Resource Locators (URLs) are also used throughout the document to provide more detail on a subject or product.

Document Text Conventions

The following table shows various typefaces to differentiate between regular text and menu commands, keyboard keys, tool bar buttons, dialog box options and text that you type. This illustrates how text conventions are represented in this manual:

Convention	Description
Notes	Notes: Notes are enclosed for emphasis
Bold	Bold type is used for emphasis, buttons or tabs to select on windows, and names of dialog boxes.
TEXT IN SMALL CAPS	TEXT IN SMALL CAPS is used for keyboard keys that you press (for example, ALT+F4)
<i>Italics</i>	Italics are used to reference other documents, sections, figures, and tables.
Special type-style	Special typestyle is used for filenames, directory names, commands, file listings, and anything that would appear in a Java program, such as methods, variables, and classes.
<i>Bold italics typestyle</i>	Bold italics is used for information the user needs to enter.
{ }	Curly brackets are used for replaceable items (for example, replace {home directory} with its proper value such as C:\caadapter).

Organization of this Guide

The *caAdapter User's Guide* contains the following chapters:

Chapter 1 Using the caAdapter User's Guide —This chapter provides an introduction to this user's guide.

Chapter 2 Overview of HL7 and caAdapter —This chapter provides an overview of HL7, caAdapter and caAdapter's architecture.

Chapter 3 Using caAdapter —This chapter provides operational scenarios for caAdapter using real-life examples.

Chapter 4 Using the caAdapter APIs —This chapter provides Java developers information required to use caAdapter.

Chapter 5 Using the caAdapter Mapping Tool —This chapter provides detailed instructions for using the caAdapter Graphical User Interface (GUI).

Chapter 6 caAdapter File Types — This chapter provides the different types of files used by caAdapter and an example of each.

Appendix A caAdapter Example Data — This appendix provides a description of the example data delivered with caAdapter.

Appendix B References — This appendix provides a list of references used to produce this guide or referred to within the text.

Appendix C caAdapter Glossary — This appendix includes a list of terms or abbreviations and their meanings.

CHAPTER 2

OVERVIEW OF HL7 AND CAADAPTER

This chapter provides an overview of HL7 and caAdapter.

Topics in this chapter include:

- *HL7 Overview* on this page
- *caAdapter Overview* on page 7
- *caAdapter Core Engine Architecture* on page 9
- *caAdapter Mapping Tool Architecture* on page 9
- *caAdapter Installation* on page 10

HL7 Overview

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 healthcare arena. HL7 provides standards for data exchange to allow interoperability between healthcare 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.

The key goal of the HL7 community is syntactic and semantic interoperability. This goal is supported in HL7 version 3 (v3) by what is 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 provides 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 has 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 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 NCICB provides training resources to assist the caBIG community and other interested parties in implementing HL7 v3 messaging. These resources include online tutorial, self-paced training and links to HL7 resources (http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter/HL7_Tutorial).

caAdapter Overview

caAdapter (<http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter>) consists of two components that support data sharing via the HL7 v3 messaging standard at NCICB (<http://ncicb.nci.nih.gov>) or any cancer center as part of the cancer Biomedical Informatics Grid (caBIG) (<http://caBIG.nci.nih.gov>) solution: the core engine and a mapping tool.

The caAdapter core engine is an open source toolkit for building, parsing and validating HL7 v3 messages from source clinical systems to promote data exchange in an international, standards-based messaging format. The core engine is a messaging framework that is based on an object-oriented data model, the HL7 RIM, and a set of v3 defined data types. This framework enables clinical applications to build and parse HL7 v3 messages based on specific schema definitions and perform vocabulary validation of the RIM structural attributes through NCI's Enterprise Vocabulary Services (EVS). caAdapter integrates with NCICB 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/caCORE3.0.1_Tech_Guide.pdf) and *caCORE Software Development Kit Programmer's Guide* (ftp://ftp1.nci.nih.gov/pub/cacore/SDK/caCORE_SDK1.1_Programmers_Guide.pdf) for more information. This supports NCICB's mission of developing a translational research infrastructure and building a clinical research network by providing a common platform for sharing data.

The caAdapter Mapping Tool is an open source application that enables analysts and database engineers, who are knowledgeable about HL7, to create a mapping from Comma Separated Value (CSV) clinical data to an equivalent target HL7 v3 XML format. It provides a front end GUI and a back end engine to support specification of file formats, drag-and-drop mapping between source and target, validation of specifications and data, and transformation of actual CSV data into HL7 v3 XML message instances.

Note: caAdapter has only been tested with the Individual Case Safety Report (ICSR) message and related sub-components from the HL7 MAY 2005 Normative Edition (RIM v2.07).

caAdapter Within the Overall NCICB Clinical Trials Architecture

NCICB's clinical trials architecture vision (*Figure 2.1*) supports open sharing of clinical and research data across the cancer research and clinical care communities. Clinical systems supply trial data to a message exchange, using caAdapter to build HL7 v3 messages to carry the data.

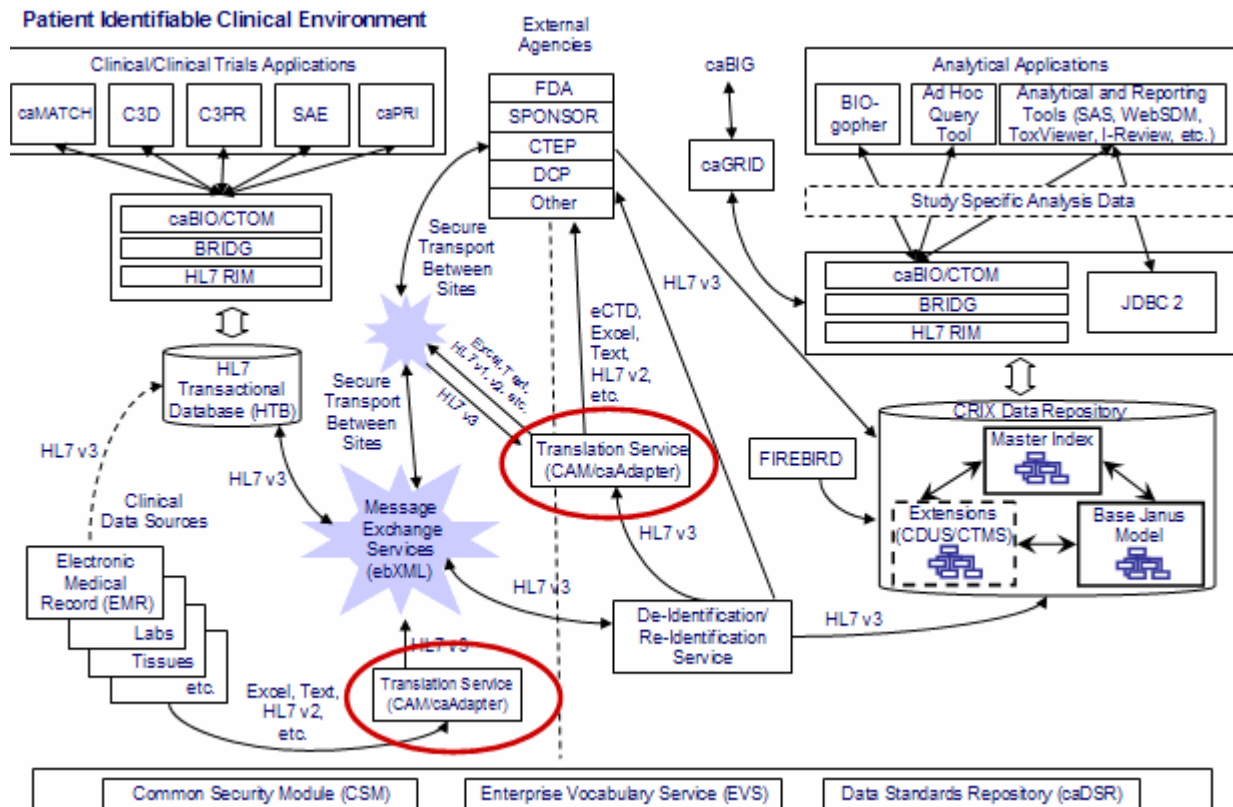


Figure 2.1 NCICB Clinical Architecture Vision

The NCICB clinical architecture vision contains the following components:

- caAdapter - Toolkit which facilitates v3 message building, parsing and validation
- Message Exchange – Service supporting message transmission and routing
- HL7 Transactional Database - HL7 RIM based transactional database and data access service
- De-identification Service – Service for de-identifying patient information from HL7 messages
- Research Application - CMAP (Cancer Molecular Analysis Program) use case for demonstration purposes
- Cancer Data Standards Repository (caDSR) - Shared metadata repository
- Enterprise Vocabulary Services (EVS) - Terminology service for hosting controlled vocabulary, including HL7 registered terminology

caAdapter Core Engine Architecture

Figure 2.2 illustrates the caAdapter core engine architecture design including its sub-systems and components.

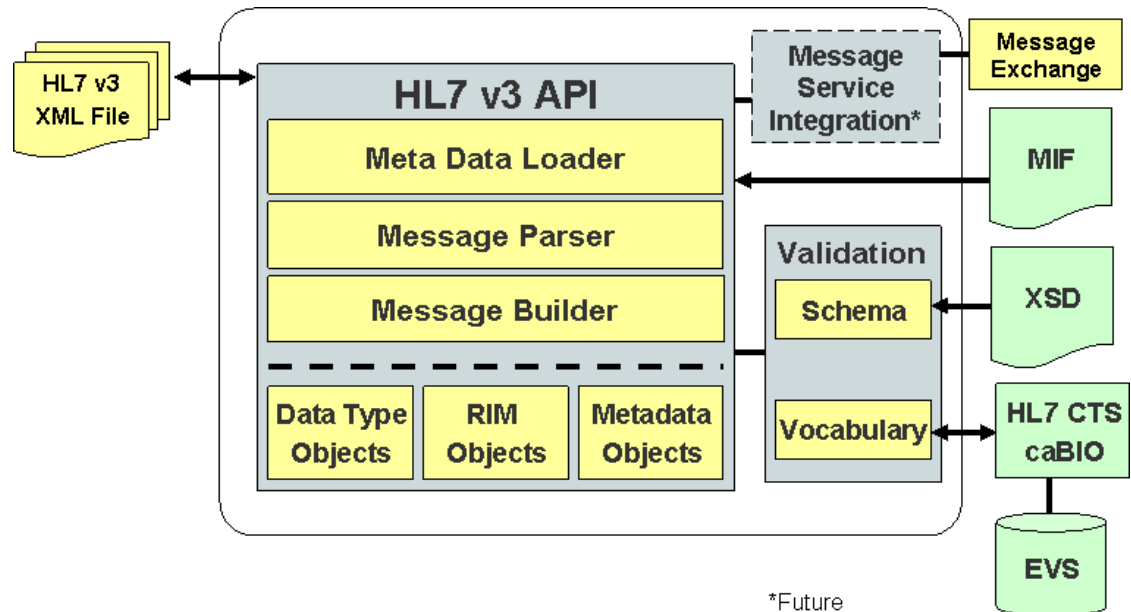


Figure 2.2 caAdapter Core Engine Architecture

The main features of the caAdapter core engine as illustrated in *Figure 2.2* are:

- Meta Data Loader - represents HL7 v3 metadata in-memory
- Message Parser – parses HL7 v3 messages to Reference Information Model (RIM) object graph
- Message Builder – builds HL7 v3 messages from the RIM object graph
- HL7 v3 Artifacts – implements RIM objects, data type objects and metadata objects
- Validation Services – integrates with NCICB caCORE components such as Enterprise Vocabulary Service (EVS)
- Message Service Integration (future plans) – integrates with message exchange services

caAdapter Mapping Tool Architecture

The caAdapter Mapping Tool is a graphical application for mapping clinical data to an HL7 v3 message. *Figure 2.3* illustrates the caAdapter Mapping Tool architecture design depicting its subsystems and components.

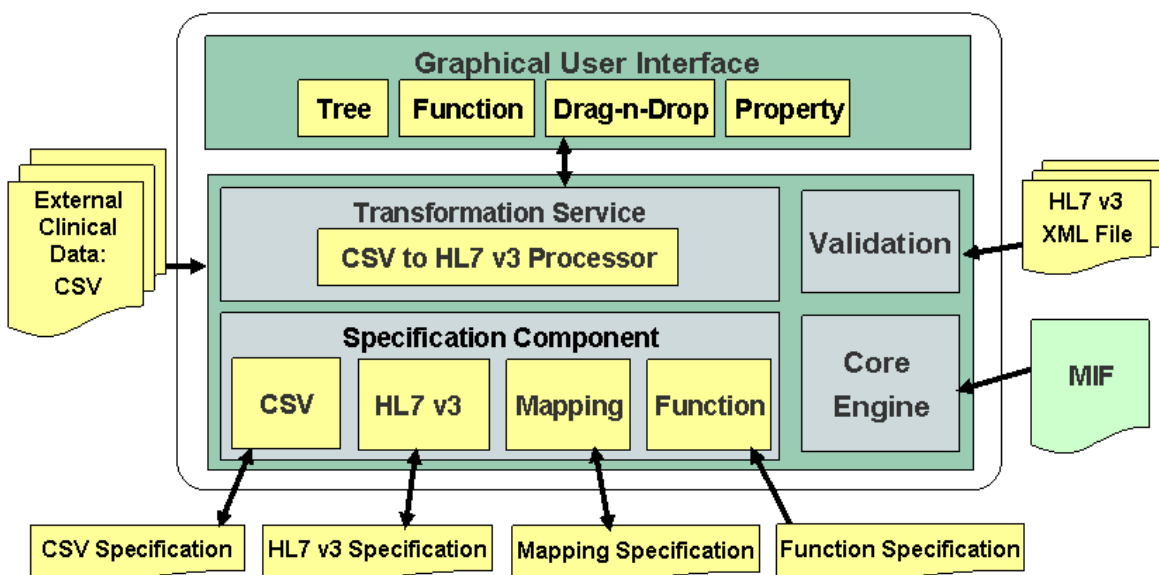


Figure 2.3 caAdapter Mapping Tool Architecture

The mapping tool provides the following:

- Source and Target Specification - graphical interface for defining input and output data formats
- User Interface - simple interface for mapping source fields to target elements containing tree structure, drag-and-drop functionality, functions and property definitions
- Mapping Functions - capability to do simple source data manipulation
- Transformation Service - generation of XML message instances from source data based on the mapping
- Validation - capability to validate the structure and content of files

caAdapter Installation

Complete instructions for installing caAdapter are found in the *caAdapter Installation Guide* located at <http://ncicb.nci.nih.gov/download/downloadhl7.jsp>.

CHAPTER 3 USING CAADAPTER

This chapter provides a high level overview for using caAdapter.

Topics in this chapter include:

- [caAdapter API Process Flow](#) on this page
- [caAdapter API Operational Scenario](#) on page 14
- [caAdapter Mapping Tool Operational Scenario](#) on page 14

caAdapter API Process Flow

This section describes the process to parse an HL7 v3 adverse event (AE) message, also known in HL7 as an ICSR, validate the instance against the schema and validate the core attribute vocabulary against the HL7 v3 vocabulary loaded in NCICB's EVS.

The basic steps to accomplish this using caAdapter are:

1. caAdapter receives an HL7 v3 message instance from a clinical system and validates the instance against the schema description in the MIF based on the unique message id.
2. caAdapter builds an object graph based on the schema definition and loads the instance into the object graph.
3. caAdapter parses the message by validating the value in each of the core attributes against the approved HL7 vocabulary in EVS.
4. caAdapter uses the object graph and builds the HL7 v3 ICSR message instance.

[Figure 3.1](#) illustrates the initial parsing and validating of the XML structure. The HL7 v3 message type is passed to the API and used to determine which XML schema describes its structure. The caAdapter API uses that schema, derived from the MIF, to parse the message and validate that its structure is compliant with the HL7 standard

RIM classes for this message type. This is the first step toward getting the data into memory where it can then be validated against EVS..



Figure 3.1 Parsing XML and validating structure

Figure 3.2 illustrates what happens after the XML structure has been validated. The HL7 v3 message content is loaded from the XML message into an RIM-based object graph in memory. With the data in memory, it can be validated next.

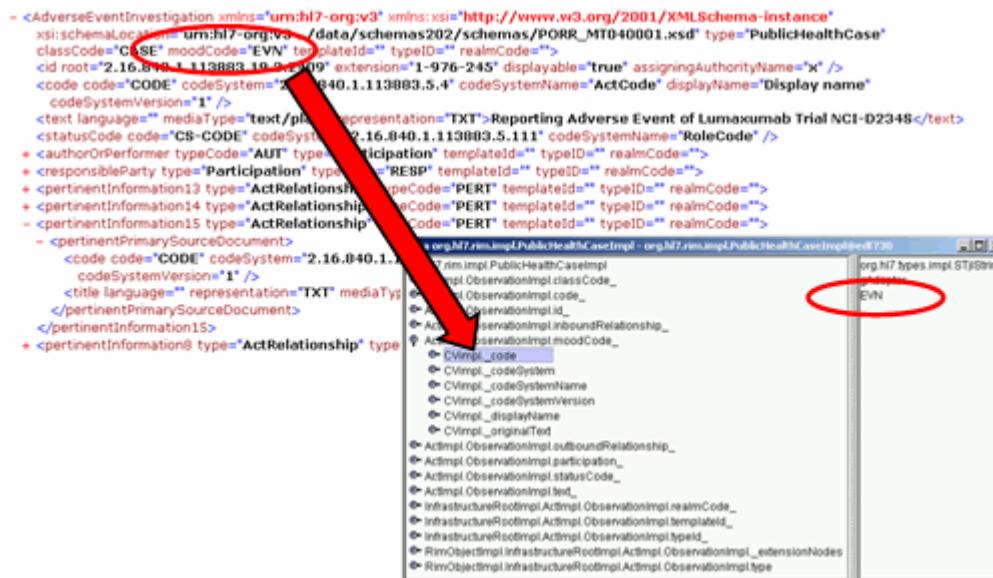


Figure 3.2 Parsing XML to object graph

Figure 3.3 illustrates traversing the object graph and validating all core structural attributes against the approved HL7 vocabulary stored in the NCI Thesaurus using EVS APIs. All warnings and errors are logged in the `caadapter.log` file. With this com-

plete, the object graph can be used to build messages or passed on to other applications.

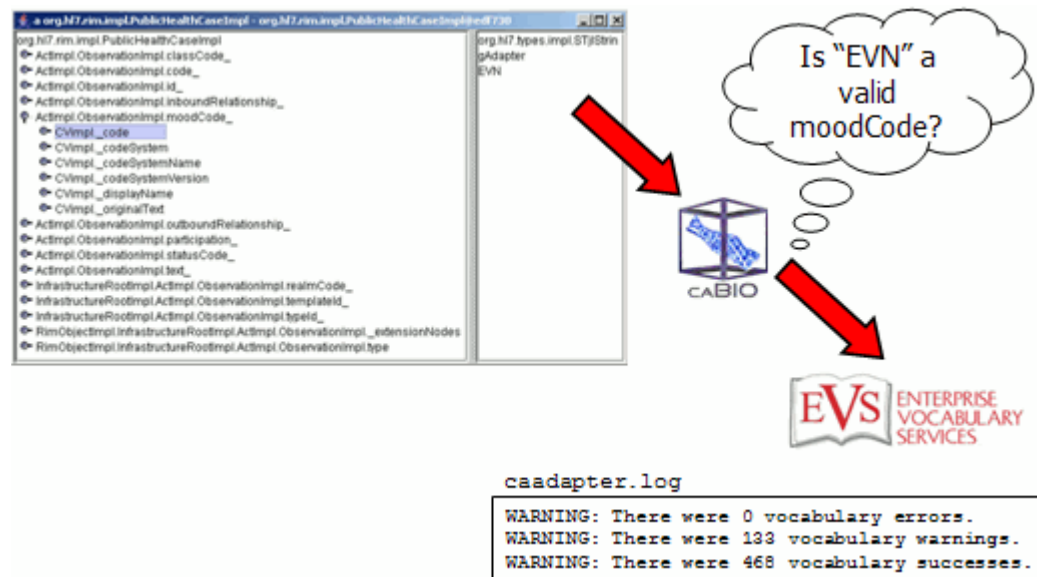


Figure 3.3 Validating the object graph against EVS

Figure 3.4 illustrates building the XML from the data in the object graph and validating the result against an HL7 v3 schema. All warnings and errors are logged in the `caadapter.log` file. The validated message can then be routed by a messaging service, stored, or whatever is needed by the owning organization.

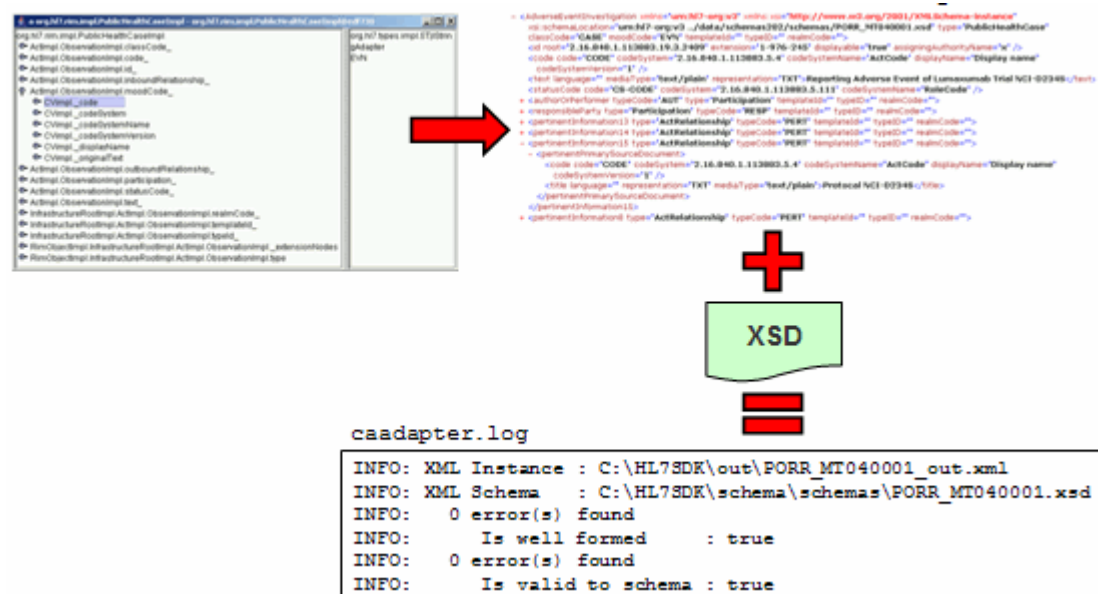


Figure 3.4 Building and validating XML output

caAdapter API Operational Scenario

A clinical trials coordinating center is automating the receipt and routing of AE reporting from the member hospitals and clinical centers. They have researched the options and chosen to implement HL7 v3 messaging. Their hospitals are implementing the messages and the coordinating center is preparing to handle the incoming messages. They have identified the caAdapter APIs as one part of their messaging infrastructure.

When their messaging service receives an HL7 v3 message from a hospital or clinical center, it calls a caAdapter API to parse the incoming message. The parser validates the message against the appropriate XML schema description based on the message ID. It then builds an object graph in memory based on the schema definition and loads the data into the object graph. Another caAdapter API is called to validate the vocabulary used for the HL7 v3 structural attributes using the NCI's EVS. This overall process builds a caAdapter log file that the system administrator can monitor.

With the validated message content held in the object graph, the system can now perform the following:

- Generate the HL7 v3 message for rerouting to the FDA using another caAdapter API for building messages.
- Pass the caAdapter object graph in an API call to a separate persistence application where the data is stored for research/data mining and administrative purposes.
- Notify the sending system that the message was received and processed using identifying data from the object graph.
- And so on.

caAdapter Mapping Tool Operational Scenario

A research hospital has been faxing AE reports to a clinical trials coordinating center for submission to the Food and Drug Administration (FDA). Instead of using a manual effort to fill out the MedWatch 3500A form, they would like to automate and streamline the process. They have a clinical data management system (CDMS) where the necessary AE data is stored. They would like to automate the process by pulling data from this system and transforming it into an HL7 v3 message to route to the FDA. Their clinical systems analyst researched the HL7 standards and identified the correct specification to use, called the ICSR. The analyst uses the caAdapter Mapping Tool to implement this plan.

The clinical systems analyst uses the caAdapter Mapping Tool to define a file specification that describes the source file for the transformation. This source specification outlines the format of a CSV file where each line is a segment containing a logical grouping of fields. Each segment may have one or more dependent child segments to handle one-to-many relationships between logical groups of data. The analyst also uses the caAdapter Mapping Tool and the HL7 ICSR's MIF file to generate a target file specification. This specification is based on the number and types of elements in the HL7 message that are needed to support their AE data. After source and target specifications are defined, the analyst then maps source fields to target fields using caAdapter's map specification tab. The application allows the analyst to drag-and-drop

CSV source fields onto HL7 target fields and use functions to manipulate the data on the way. The result of this step is that a mapping specification is generated by the caAdapter Mapping Tool. After the mapping is complete, the analyst then uses the caAdapter Mapping Tool to test the generation of HL7 v3 ICSR XML message instances using a sample CSV file obtained from the CDMS.

When this development process is complete, the caAdapter specification files and transformation APIs can be implemented as part of a message routing infrastructure to deliver AE data to the FDA in a streamlined fashion.

CHAPTER 4

USING THE CAADAPTER APIs

This chapter describes the set of primary caAdapter APIs.

Topics in this chapter include:

- [caAdapter Directory Structure](#) on this page
- [caAdapter APIs](#) on page 18
- [caAdapter API Error Logs](#) on page 21

caAdapter Directory Structure

Table 4.1 contains the directories under your {home directory}.

Directory	Contents
build	Binaries (.class files), only for source distribution and is created at runtime
docs	Javadocs and other useful information
etc	Important supplementary files
images	Images used by the GUI
lib	Java libraries and dependencies; and the MIF.zip file
license	License and legal information
schema	HL7 v3 Schema files
src	Source code (.java files) (only for source distribution)
workspace	Default directory where you can save project files. It contains log files and HL7 v3 XML instances. It also contains an <code>examples</code> directory with example data (see Appendix A caAdapter Example Data).

Table 4.1 Directory structure for caAdapter

caAdapter APIs

There are five primary modules in the set of caAdapter APIs.

1. Meta Data Loader
2. HL7 v3 Message Parser
3. HL7 v3 Message Builder
4. Transformation Service
5. Vocabulary Validation

The following sections provide a description of each.

Meta Data Loader

HL7 provides the following format for specifying message metadata (structure, format, and constraints):

- Model Interchange Format (MIF)

MIF is XML based. When the message is being parsed, the Meta Data Loader drives how the RIM object graph is built.

Note: The Meta Data Loader supports both format types, but only the MIF loader is fully tested.

The Meta Data Loader classes are located in the `org.hl7.meta` package.

The following example demonstrates how to use the Meta Data Loader. `JdomMessageLoader` is an implementation of `MessageTypeLoader` that loads message type metadata from an MIF file. It accepts two parameters:

1. MIF file - located at `lib` directory
2. RIM file - located at `etc` directory

```
JdomMessageTypeLoader jmtl = new JdomMessageTypeLoader(mif,
rimFile);
```

```
MessageType messageType = jmtl.loadMessageType(msgTypeId);
```

HL7 v3 Message Parser

The message parser utilizes a Simple API for XML (SAX) parser to parse the XML nodes of the input HL7 V3 message. As SAX events fire, data from the SAX event is used to query meta classes for metadata. The specific metadata allows the parser to instantiate a node as a RIM object or HL7 data type object. The metadata also denotes where to attach the instantiated object. When an instance of a RIM class or data type is needed, it is created by a properties based factory. Once the entire message is parsed, the resulting object graph is an in-memory representation of the RIM object graph.

The main parsing classes (content handlers) are located in the `org.hl7.xml.parser` package.

The following example demonstrates how to use the HL7 v3 Message Parser. The `MessageContentHandler` bootstraps the XML parsing process. `xmlInputStream`

is the HL7 v3 message. `messageType` represents the metadata corresponding to the HL7 v3 message.

```
RimObject graph = MessageContentHandler.parseMessage(xmlInputStream, messageType);
```

HL7 v3 Message Builder

The HL7 v3 Message Builder is the reverse process of the HL7 v3 Message Parser. The message builder's design resembles the message parser since it relies on small specialized content handlers. If you could read or parse the in-memory RIM graph and pretend to be a SAX XMLReader, essentially generating the same events as if you were reading an XML stream, then you could use that reader (in an identity transform) to create XML output.

The mechanism that masquerades as a SAX XML reader is made up of the classes `XMLSpeaker` and `XMLRimGraphSpeaker`. As the SAX events are emitted by "parsing" the in-memory RIM graph, they are handled by "MessageBuilders". The message builders are then dynamically switched for one another as appropriate. Finally, one of the builders emits a SAX event to an internal member of type `org.xml.sax.ContentHandler`. Once this event fires, the XSLT transform takes over and the XML for that particular node is created.

The message building classes are located in the `org.hl7.xml.builder` package.

The following example demonstrates how to use the message builder.

`javax.xml.transform.Transform` is a class which can process XML from a variety of sources and write the transformation output to a variety of sinks. `RimGraphXMLSpeaker` serializes HL7 RIM object graphs.

```
Transformer transformer = TransformerFactory.newInstance().newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

RimGraphXMLSpeaker speaker = new RimGraphXMLSpeaker();

Source source = new SAXSource(speaker,
    new RimGraphXMLSpeaker.InputSource((RimObject) graph, messageType.getRootClass()));

StringWriter sw = new StringWriter();

try {
    transformer.transform(source, new StreamResult(sw));
} catch (Exception e) {
    e.printStackTrace();
}

String builtXml = sw.toString();
```

Transformation Service

The transformation service reads the mapping file and converts a compliant source file into a series of HL7 v3 XML instances. The mapping file contains a reference to the

source specification, target specification, function library specification, and mapping information.

The transformation service classes are located in the `gov.nih.nci.hl7.map` package.

The following example demonstrates how to use the transformation service. Given the CSV source file and the mapping file, the `TransformationService` class transforms the CSV file into the `MapGenerateResult` class, which contains the generated HL7 v3 message text and the corresponding validation results.

```
String mapFile040011 = FileUtil.getExamplesDirPath() + "/
040011/040011.map";

String csvSource040011 = FileUtil.getExamplesDirPath() + "/
040011/040011.csv";

TransformationService transformationService = new Transforma-
tionService(mapFile040011, csvSource040011);

List<MapGenerateResult> mapGenerateResults = transformaSer-
vice.process();

for (int i = 0; i < mapGenerateResults.size(); i++)
{
    MapGenerateResult mapGenerateResult = mapGenerateRe-
sults.get(i);

    String hl7V3Message = mapGenerateRe-
sult.getHl7V3MessageText();

    System.out.println("Generated HL7 Message:\n" +
hl7V3Message);

    ValidatorResults validatorResults = mapGenerateResult.getVal-
idatorResults();

    System.out.println("Validation Results:\n" + validatorRe-
sults);
}
```

Vocabulary Validation

Vocabulary validation provides the ability to validate HL7 structural attributes against the NCICB's EVS by using the Cancer Bioinformatics Infrastructure Objects (caBIO) API. It implements the HL7 Common Terminology Service (CTS) interfaces. Vocabulary validation walks through the RIM object graph based on the metadata.

The vocabulary validation classes are located in the `gov.nih.nci.hl7.valida-
tion` package.

The following example demonstrates how to use the vocabulary validation class.

```
VocabularyService vService = new VocabularyService();

boolean isValid = vService.validateCoreAttributes(rimObject-
Graph, messageType);
```

caAdapter API Error Logs

Many of the targets provide logging information that is printed to the console and saved to a file. The log files can be found in the `{home directory}\workspace` directory. All log messages are saved to the file `caadapter.log.#` where # is the number of the log file created.

The logging utility is configurable; edit the `{home directory}\logging.properties` file to change your logging properties.

CHAPTER 5

USING THE CAADAPTER MAPPING TOOL

This chapter defines the step-by-step procedures to use the caAdapter Mapping Tool.

Topics in this chapter include:

- *Prerequisites for Using the caAdapter Mapping Tool* on this page
- *Starting the caAdapter Mapping Tool* on this page
- *caAdapter Mapping Tool Process Flow* on page 24
- *caAdapter Mapping Tool Common Features* on page 25
- *Source Specification* on page 30
- *Target Specification* on page 35
- *Map Specification* on page 52
- *HL7 v3 Message* on page 60

Prerequisites for Using the caAdapter Mapping Tool

You must have the following prerequisites to successfully use the mapping tool:

- Thorough familiarity with your source data
- Strong knowledge of HL7 artifacts, messages and data types
- Training on the caAdapter Mapping Tool
- Familiarity with caAdapter Mapping Rules documentation

Starting the caAdapter Mapping Tool

Starting the Mapping Tool from the Binary Distribution

Perform the following steps to launch the caAdapter Mapping Tool GUI.

1. In a Command Prompt window, enter **cd {home directory}** to go to your home directory (for example, in Windows C:\caadapter).
2. Enter **java -jar caadapter_ui.jar**
The Welcome to the caAdapter screen momentarily appears and the caAdapter Mapping Tool GUI displays.

Starting the Mapping Tool from the Source Distribution

Perform the following steps to launch the caAdapter Mapping Tool GUI.

1. In a Command Prompt window, enter **cd {home directory}** to go to your home directory (for example, in Windows C:\caadapter).
2. Enter **ant compile**
3. Enter **ant launchui**
The Welcome to the caAdapter screen momentarily appears and the caAdapter Mapping Tool GUI displays.

Starting the Mapping Tool from the Windows Distribution

Perform the following step to launch the caAdapter Mapping Tool GUI.

1. Select **caAdapter** from the **Start** menu shortcut.
The Welcome to the caAdapter screen momentarily appears and the caAdapter Mapping Tool GUI displays.

caAdapter Mapping Tool Process Flow

The basic steps to use the caAdapter mapping tool (as illustrated in [Figure 5.1](#)) are as follows:

1. Generate a CSV specification file from CSV data (see [Segmented CSV Specification](#) on page 30).
2. Generate an HL7 specification file from an HL7 MIF file (see [HL7 v3 Specification](#) on page 35).
3. Load the source specification (CSV specification) at the left side and load the target specification (HL7 specification) at the right side in the mapping tool GUI. Define mappings by drawing lines between elements of the source and target sides and save the mapping to an XML file (see [Map Specification](#) on page 52).
4. Select the CSV data and the mapping file and the mapping engine transforms the data into a RIM object graph based on the mapping file. caAdapter uses the

object graph and builds the HL7 v3 message instance (see [HL7 v3 Message](#) on page 60).

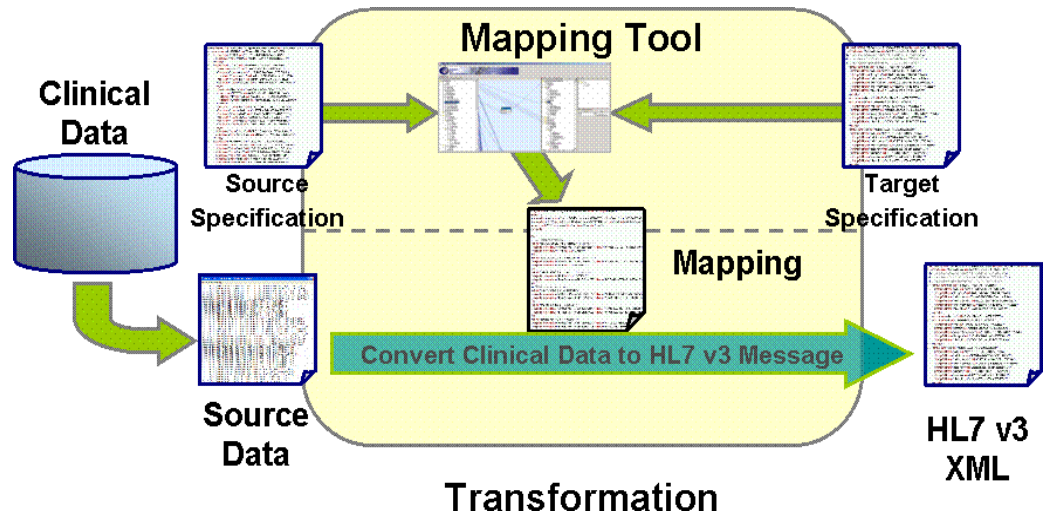


Figure 5.1 How the mapping tool works

caAdapter Mapping Tool Common Features

caAdapter Mapping Tool Interface

The caAdapter Mapping Tool interface ([Figure 5.2](#)) is based on a Windows layout with a menu bar, a Center for Bioinformatics icon, a tool bar and open tabs located in the top

of the window. The different panels can be resized by selecting the edge of the panel and dragging. Scroll bars are used when needed to display all the information.

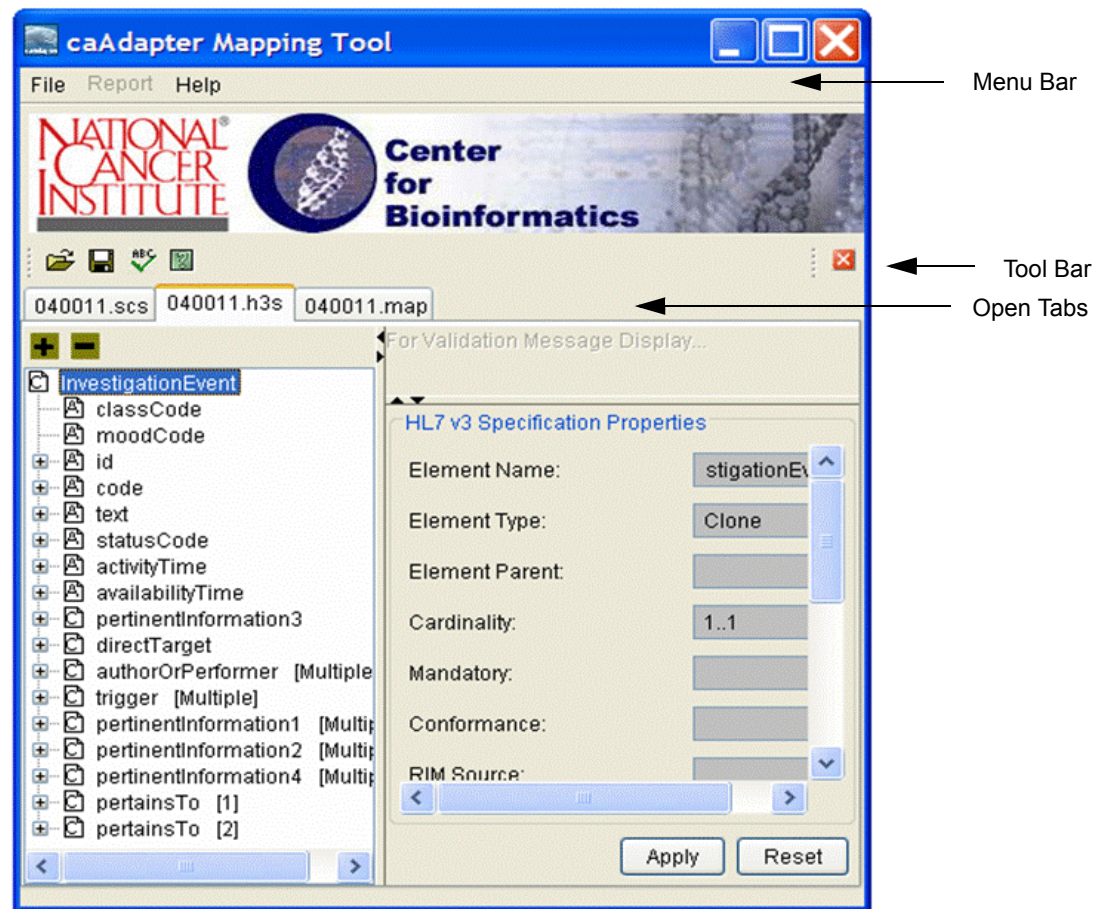


Figure 5.2 Mapping Tool interface

The menu bar is context sensitive. Only the options that are available for your current window are displayed in black font; the other options are grayed-out. For example, the **Report** option is grayed-out in [Figure 5.2](#) since it is not available for the .h3s file.

The menu bar consists of the **File**, **Report** and **Help** options. The **File** option allows you to perform the functions in [Table 5.1](#).

<i>File Options</i>	<i>Description</i>
New	Creates a new file for the type of file you select including: <ul style="list-style-type: none">• Map Specification• CSV Specification• HL7 v3 Specification• HL7 v3 Message

Table 5.1 File menu options

File Options	Description
Open	Opens an existing file for the type of file you select including: <ul style="list-style-type: none"> • Map Specification • CSV Specification • HL7 v3 Specification
Save (CTRL+S)	Saves the file you are currently working on (in the selected tab).
Save as	Opens a 'Save as' dialog box to allow you to save the file to another file name.
Validate	Validates the file you are currently working on (in the selected tab).
Close (CTRL+F4)	Closes the file you are currently working on (in the selected tab). The following Data Changed dialog box appears if you have not saved your work, "Data has been changed but is not saved. Would you like to save your changes?" Select Yes, No or Cancel.
Close all	Closes all the files.
Exit (ALT+F4)	Exits the caAdapter Mapping Tool.

Table 5.1 File menu options

The **Report** option currently allows you to generate reports for CSV and map specifications. The **Help** option contains About caAdapter and online help Contents and Index options.

The tool bar is context sensitive. Only the options that are available for your current window are displayed. [Table 5.2](#) contains a list of available icons. They are shortcuts for the same functionality that can be accessed from the menu bar.





Tool Bar Options	Description
	Opens a new file of the type of file that is currently open.
	Saves the file that is currently open.
	Closes the tab that is currently open.
	Validates the file that is currently open.

Table 5.2 Tool bar options



<i>Tool Bar Options</i>	<i>Description</i>
	Refreshes the mapping panel. It is only visible if it is on the mapping panel.
	Opens the Help window.

Table 5.2 Tool bar options

The caAdapter Mapping Tool uses a document-oriented paradigm where up to four files of different types can be open at the same time, each within its own tab in a single window. The four different types of tabs are:

1. CSV specification
2. Map Specification
3. HL7 v3 Specification
4. HL7 v3 Message

Only one of each file type may be open at a time. If you open a new file of a file type that is already open, then the existing file will be replaced with the new file. The tab name displayed is the name of the file in the tab (for example, 040011.h3s as shown in [Figure 5.2](#)) or it is labeled `untitled.<ext>`, where `<ext>` is the appropriate file extension for that type of tab as shown in [Table 6.1](#) on page 65. The window layout changes depending on the type of tab displayed. For example, the HL7 v3 specification tab displays a tree structure in the left-hand panel and the properties and validation messages in the right-hand panel.

caAdapter Mapping Tool Validation

The caAdapter Mapping Tool contains validation on each of its tabs. Validation is used to:

1. Validate the given specification to ensure it is technically correct before continuing onto the next step.
2. Provide a user-friendly method to report errors so you can correct them.
3. Provide reminder notes on the process (information messages).

The results of the validation are displayed in the **Validation Messages** panel (see [Figure 5.3](#)). Only one level of message is displayed at a time. From this panel you can:

- Change the **Message Level** by selecting a different level from the drop-down list.
- Click **Save** to save the messages to a file.
- Click **Print** to send the messages to your printer.

- Select a message to display the full content of the selected message in a panel below the Validation Messages panel (see [Figure 5.3](#)).

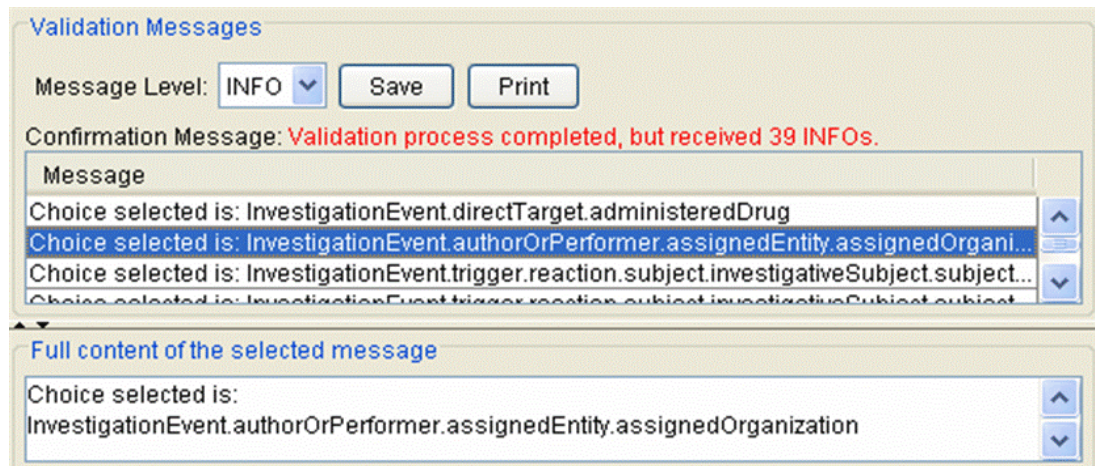


Figure 5.3 Validation Messages panel

[Table 5.3](#) contains the different levels of messages produced during validation.

Message Level	Description	Example
FATAL	The process leads the application into an unrecoverable situation where the application itself has to halt the process instead of moving forward.	A file with a wrong file type is given to the map specification module and it does not know how to open the file.
ERROR	The process leads the application into a recoverable situation with serious issues that require your attention. It is better if these errors are resolved before proceeding or you could receive partial or incorrect results.	The CSV data does not match a given specification.
WARNING	The process leads the application into a recoverable situation with medium level issues that won't prevent the application from proceeding further. However, it may require your attention to resolve them so the process will generate the expected results for you.	Not all segments and fields within the CSV specification have been mapped to the HL7 v3 specification.
INFO	Contains information for you, such as tips, suggestions, reminders, etc. You can simply ignore them if you want to.	Contains the choice selected for an element

Table 5.3 Validation messages

Source Specification

Segmented CSV Specification

Business Rules

Following are the business rules for a segmented CSV specification:

- Two or more segments cannot have the same name
- Two or more fields cannot have the same name in same segment (case-insensitive)
- Segment names must be a combination of any letters (A-Z) in CAPITOLS, numbers or the underscore character
- Field names must be a combination of any letters (A-Z or a-z), numbers or the underscore character

Step-by-Step Instructions

This section contains the detailed instructions to use the mapping tool to create or update a CSV specification.

Overview of CSV Specification Tab

The CSV specification tab (see [Figure 5.4](#)) allows you to identify the hierarchy of segments and fields that describe an incoming CSV data file that must be converted to one or more HL7 v3 XML messages. The CSV specification tab separates the tree structure in the left-hand panel from the validation results and properties in the right-hand panel. The tree structure displays the hierarchy of segments and fields that represent the way data in the source CSV files are organized. Typical features of the tree structure are used, such as dragging and dropping an element to another location in the tree, or the ability to expand and collapse a branch of the tree using the + and - symbols respec-

tively. The **Properties** section in the right-hand panel allows you to work with the meta-data on the left.



Figure 5.4 CSV specification tab

The following sections describe how to access, update, validate and save the CSV specification.

Creating and Opening a CSV Specification

First, you must create a new or open an existing CSV specification. Select **File > New > CSV Specification** to display the **New CSV Specification** dialog box. Select one of the following as the source to create a new CSV specification.

1. **Blank CSV Schema** - Click **OK** to open the CSV specification file, named `Untitled_1.scs`, in a new tab with an empty tree except for an initial root segment which is named `ROOT` by default.
2. **Generate from a CSV Instance** - Click **Browse** to display the **Open CSV File** dialog box. Select the appropriate `.csv` file and click **Open**. The file is displayed in the **New CSV Specification** dialog box. Click **OK** to open the CSV data file in a new tab, named `Untitled_1.scs`, with the information from the data file displayed.
3. **New from an Existing CSV Specification File** - Click **Browse** to display the **Open CSV Specification** dialog box. Select the appropriate `.scs` file and click **Open**. The file is displayed in the **New CSV Specification** dialog box. Click **OK** to open the CSV specification file in a new tab, named `Untitled_1.scs`, with the information from the selected file displayed.

Or open a CSV specification that was previously saved by performing the following.

4. Select **File > Open > CSV Specification** to display the **Open CSV Specification** dialog box. Select the appropriate `.scs` file and click **Open**. A new tab opens with the CSV specification displayed in the tree.

Updating the CSV Specification

Once you have a CSV specification open, you can perform the following basic functions to update the tree hierarchy. You want to update any default field names to have meaningful names.

1. Click on a segment in the tree structure, to display the details of that element in the **Segment Properties** section (see [Figure 5.5](#))

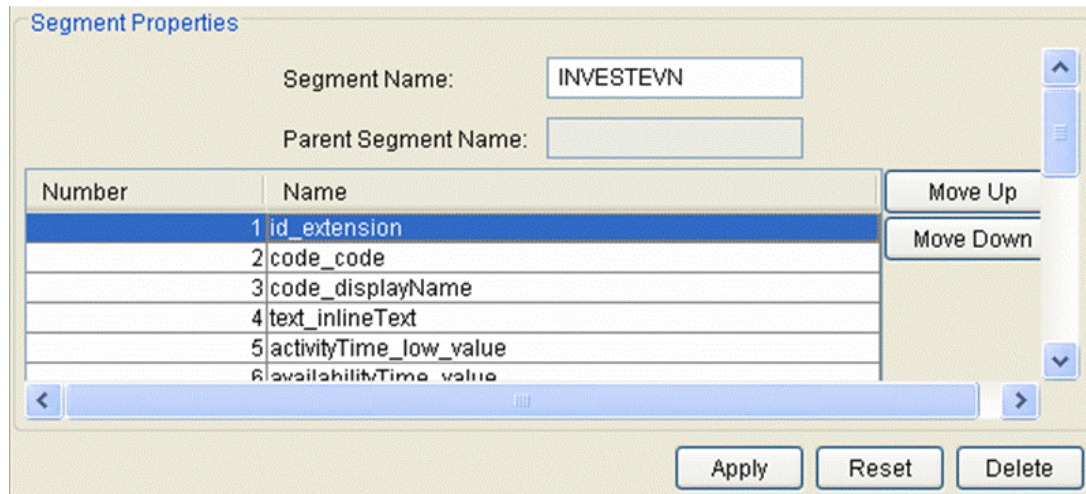


Figure 5.5 Segment properties

Click the **Move Up** and **Move Down** buttons to re-arrange the sequence of the fields displayed under the given segment. By default the **Move Up** and **Move Down** buttons are both disabled unless you select any element in the field list (they are enabled in [Figure 5.5](#) because **id_extension** is selected). Select a number/name row and click the **Move Up** or **Move Down** button until you have the fields arranged correctly. Click **Apply** to update the tree structure.

Edit the **Segment Name** and click **Apply** to update the tree in the left-hand panel.

2. Right-click on a segment to get the options available to perform on that segment (see [Figure 5.6](#))

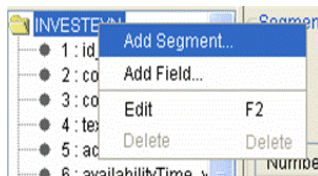


Figure 5.6 CSV segment right-click options

- a. Right-click and select **Add Segment** to display the **Add Segment** dialog box. Enter the **CSV segment name** and click **OK**. The segment is added to the tree structure.

- b. Right-click and select **Add Field** to display the **Add Field** dialog box. Enter the **CSV field name** and click **OK**. The field is added to the tree structure.
 - c. Right-click and select **Edit** to display the **Edit** dialog box. Edit the **CSV segment name** and click **OK**. The segment name is changed in the tree structure.
 - d. Select one or more segments, right-click and select **Delete** to display the **Confirmation** dialog box. Click **Yes** or **No**. The segment name(s) are deleted from the tree structure.
3. Click on a field in the tree structure, to display the details of that element in the **Field Properties** section (see [Figure 5.7](#)).

The image shows a 'Field Properties' dialog box. It contains three labeled text input fields: 'Segment Name' (containing 'INVESTEVN'), 'Field Name' (containing 'id_extension'), and 'Field Sequence Number' (a dropdown menu currently showing '1'). Below these fields are three buttons: 'Apply', 'Reset', and 'Delete'.

Figure 5.7 Field Name Metadata Properties

Edit the **Field Name** and click **Apply** to update the tree in the left-hand panel.

4. Right-click on a field to get the options (**Edit**, **Delete**) available to perform on that field
 - a. Right-click and select **Edit** to display the **Edit** dialog box. Edit the **field name** and click **OK**. The field name is changed in the tree structure.
 - b. Select one or more fields, right-click and select **Delete** to display the **Confirmation** dialog box. Click **Yes** or **No**. The field name(s) are deleted from the tree structure.
5. Drag-and-drop a field or segment to another area in the tree to rearrange the tree contents. Moving a segment takes its complete sub-tree with it. You may not drag-and-drop the root segment; it must remain as the root, but its fields may be moved. The cursor indicates when the field or segment can be dropped.
6. The **Reset** button can be used to reset changes made before selecting **Apply**.
7. The **Delete** button can be used to delete an element from the tree.

Validating the CSV Specification

Once you are satisfied with the CSV specification, you can validate it by performing the following steps.

1. Select **File > Validate** or select the Validate icon from the tool bar to display the **Validate** dialog box (see [Figure 5.8](#)).

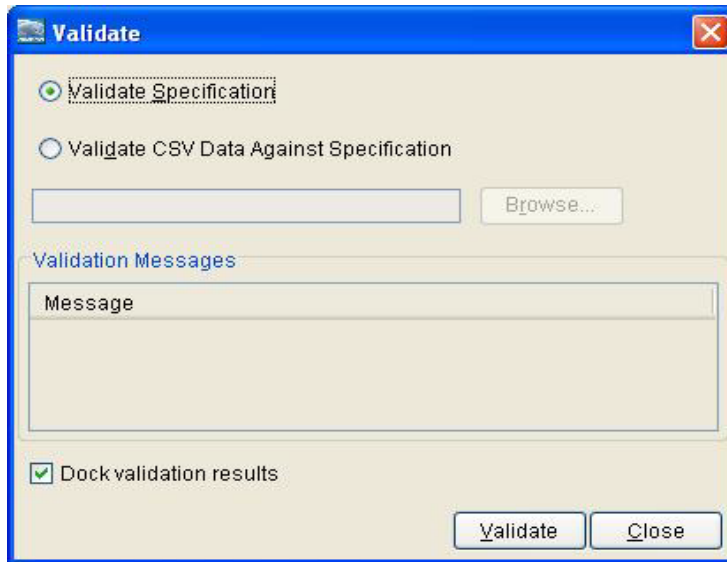


Figure 5.8 CSV Validate options

2. Select one of the following:
 - a. **Validate Specification** to validate the specification. Click **Validate**.
 - b. Or select **Validate CSV Data Against Specialization** to test a CSV data file against the specification. Click **Browse** to display the **Open CSV File** dialog box. Select the appropriate .csv file and click **Open**. Click **Validate**.
3. The **Dock validation results** check box is automatically selected so that the messages are displayed in the right-hand panel, after the validation dialog is closed. The read-only validation messages are displayed (see [caAdapter Mapping Tool Validation](#) on page 28 for more information on using the validation messages).

Saving a CSV Specification

When you are finished working on the CSV specification, select **File > Save** or **File > Save As** from the menu bar or click the save icon on the tool bar to create an XML-like file describing the structure of the tree structure. This file is portable and can be opened by the same or another user later.

Generating a Report

When a CSV specification tab is selected, you can export the CSV specification into an Excel spreadsheet by performing the following steps.

1. Select **Report > Generate Report** from the menu bar to display the **Select File to Save Generated Report** dialog box.
2. Enter a **File name** and click **Save**. A Report has been successfully generated message displays.

A part of a generated CSV report is shown in [Figure 5.9](#).

	A	B	C	D	E
1	Segment Name	Field 1	Field 2	Field 3	Field 4
2	INVESTEVN	id_extension	code_code	code_displayName	text_in
3	CASESERIOUSNESS_1	code_code	code_displayName	value_code	value_
4	APPROVAL_2	id_extension	effectiveTime_low_value		
5	TERRITORY_21	code_code	code_displayName		
6	GOVERNINGAGENCY_22	id_extension	name_inlineText	name_suffix	
7	PLAYINGMANUFACTURER_23	id_extension	code_code	code_displayName	name_
8	PARTMEDICATION_3	code_code	code_displayName	name_inlineText	name_
9	ASSIGNEDENTITY_4	code_code	code_displayName		
10	REPRESENTEDSCOPINGORGAN	code_code	code_displayName	name_inlineText	name_

Figure 5.9 Part of a generated CSV report

Target Specification

HL7 v3 Specification

Business Rules

Following are the business rules for the HL7 v3 specification:

- Abstract data types must be specialized.
- A choice must be selected on choice options.
- If an element's cardinality is one then you must have a mapping.
- If an element's cardinality is greater than one then you have a choice to add multiple fields.
- Only mandatory clones are populated.

Step-by-Step Instructions

This section contains the detailed instructions on how to use the mapping tool to create or update an HL7 v3 specification.

Overview of HL7 v3 Specification Tab

The HL7 v3 specification tab ([Figure 5.10](#)) allows you to identify the hierarchy of elements needed for your data based on what is available in the predefined structure of an HL7 v3 message type. You update the basic specification to reflect your specific requirements, such as adding multiples of fields with a cardinality of greater than one,

including or excluding clones, defining concrete data types for abstract ones or selecting choice options.

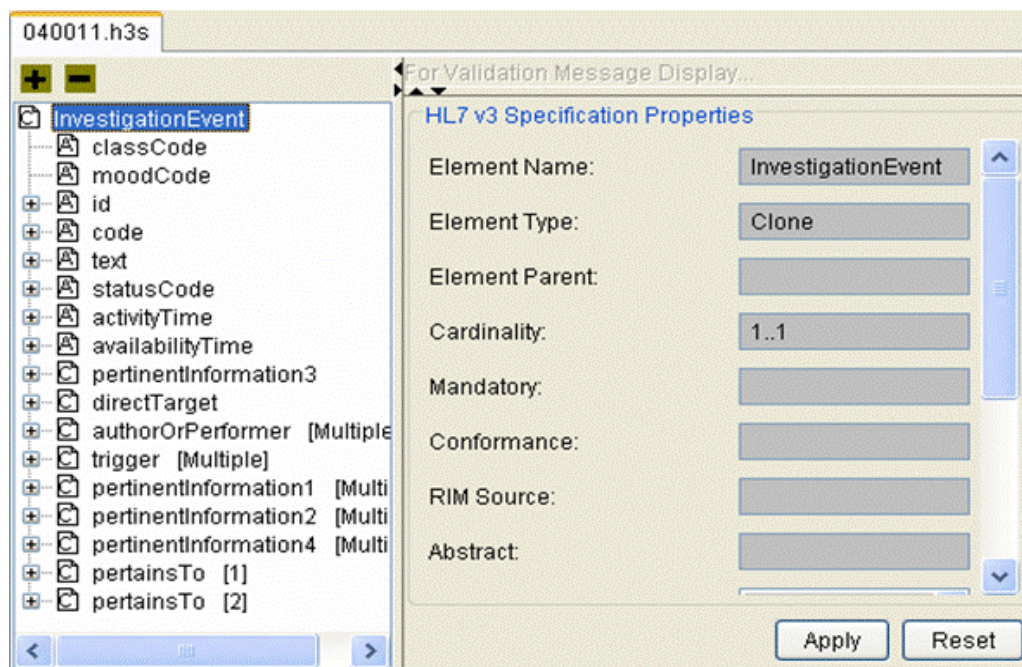


Figure 5.10 HL7 v3 Specification Tab

The HL7 v3 specification tab ([Figure 5.10](#)) separates the tree structure in the left-hand panel from the properties and validation messages in the right-hand panel. The tree structure displays the hierarchy of elements that represent the way data in the .h3s files are organized. The elements are designated as follows:

- **C** - Clone
- **A** - Attribute
- **D** - Data Type

The Common Message Element Type (CMET) code in the tree structure (for example, **COCT_MT120104**) indicates that an element represents the beginning of a CMET and is for informational purpose only.

Typical features of the tree structure are used, such as the ability to expand and collapse a branch of the tree using the **+** and **-** symbols respectively. The properties panel allows you to update some information such as the user-defined default value for a given data type field or to select a concrete data type for a given Attribute. Right-click

on an element to display the actions available as shown in the submenu in [Figure 5.11](#). The available options are regular font where the grayed-out options are not available.

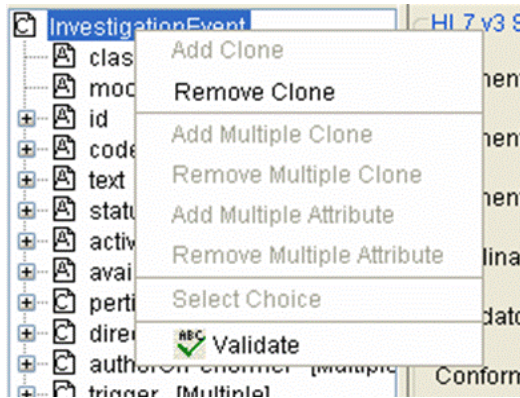


Figure 5.11 Options for HL7 v3 elements

The following sections describe how to create, update, validate and save the HL7 v3 specification.

Creating and Opening an HL7 v3 Specification

You must either create a new or open an existing HL7 v3 specification. Perform the following steps to create a new HL7 v3 specification.

1. Select **File > New > HL7 v3 Specification** to display the **HL7 v3 Specification** dialog box with valid message types.
2. Select the appropriate HL7 message type from the drop-down list (see [Figure 5.12](#)) and click **OK**.

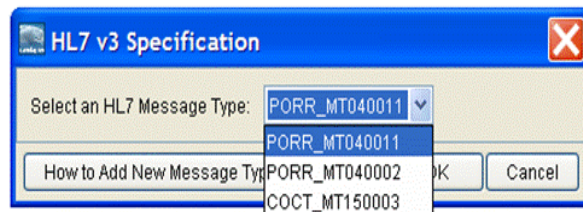


Figure 5.12 Example HL7 v3 message types

Currently, the mapping tool supports the following HL7 message types:

- **PORR_MT040011** (ICSR) - main Adverse Event message type
- **PORR_MT040002** (Drug Intervention) - Substance Administration Event message type (medium sized message)
- **COCT_MT150003** (Organization Contact) - A frequently used CMET (small sized message)

See [Extending caAdapter to Include New HL7 Message Types](#) on page 38 to add a new message type to the mapping tool.

3. A new HL7 v3 specification tab displays with the name `untitled_1.h3s`.

Or perform the following steps to open an existing HL7 v3 specification.

1. Select **File > Open > HL7 v3 Specification** to display the **Open HL7 v3 Specification (H3S) File** dialog box. Select a **File name** to open and click **OK**. The HL7 v3 specification displays in a tab with the name of the file, {file name}.h3s.

Extending caAdapter to Include New HL7 Message Types

Note: caAdapter was designed to be generic. It allows users to build and parse HL7 v3 messages based on the May 2005 Normative Edition (RIM v2.07). However, the current version of the software was only tested for the following messages: PORR_MT040011, PORR_MT040002, and COCT_MT150003. Extending caAdapter to generate other HL7 v3 messages may yield unpredictable results.

The following example demonstrates how to extend caAdapter to include an additional message type. Please follow each step carefully then stop and restart the caAdapter Mapping Tool GUI.

1. Update **message-types.properties** file on {caAdapter_Home} directory.
 - a. Add your message type to the **supported** list of messages. If your message type is not a Common Message Element Type (CMET), add your message type to the **nonprefixed** list as well. This will allow you to generate a new.h3s file based on its corresponding MIF ([Figure 5.13](#)).

Message type name has the following HL7 v3 artifact naming convention:
{UUDD_AAxxxxxx}

- UU = Sub-Section code
- DD = Domain code
- AA = Artifact or Document code. (Message Type will be MT)
- xxxxxx = Six digit zero-filled number

```
#####
#   Properties related to HL7 v3 Message Types
#####

#   Supported HL7 v3 Message Types
supported=PORR_MT040001,PORR_MT040002,COCT_MT090102,PRPA_MT101001 |

#   Messages whos root XML element should *not* be prefixed with the s
#   For example, <PORR_MT040002.SubstanceAdministrationEvent> vs. <Sub
nonprefixed=PORR_MT040001,PRPA_MT101001
```

Figure 5.13 Update message-types.properties

- b. After finishing these steps and restarting the caAdapter Mapping Tool GUI, the new message type will appear in the dropdown list when selecting: **File > New > HL7 v3 Specification** ([Figure 5.14](#)).



Figure 5.14 Select a Message Type on HL7 Specification Panel

2. Update the **cmet-file.properties** file, located in the **{caAdapter_Home}** directory, to include the MIF file names corresponding to the new message type. This file is used by caAdapter to locate the MIF files necessary to generate an .h3s file, or an HL7 message (Figure 5.15). Following is an explanation of the different parts of the `cmet-file.properties` file and their content:
 - a. **Section A** is a list of RMIM and corresponding MIF file names for all the CMETs and IMETs used in the targeted HL7 v3 message that you intend to generate. All CMETs and IMETs that will be accessed by the target message type (registered in **Section B or C**) **must be listed** in **Section A**. Failure to do so will generate a critical error and caAdapter will not be able to generate the message properly.
 - b. **Section B** is a list of target message types that you intent to generate. These are the main messages and are neither CMETs nor IMETs.
 - c. **Part C** is a list of target message types that are used for testing purposes only. These are generally CMETs or IMETs that will be used in a production system's main target message when testing is broken down into components to workout problems before testing the main target message CMETs or IMETs. To generate an .h3s file or an HL7 message of CMET or IMET message type independently, message type and the MIF file names must be listed in this part.

```

#####
# Section A: RMIM Names and Corresponding MIF File Names
#
# This section contains a list of all the RMIM and MIF file pairs
# CMETs and IMETs used in the targeted HL7 v3 messages listed in s
#####

# IMETs:
A_DeviceInterventionUniversal=PORR_MT040003UV01.mif
R_DeviceUniversal=PORR_HDO40005UV01.mif

# CMETs:
R_LocationLocatedEntityUniversal=COCT_MT070000UV01.mif
E_OrganizationUniversal=COCT_MT150000UV02.mif
E_OrganizationContact=COCT_MT150003UV03.mif
E_PlaceUniversal=COCT_MT710000UV01.mif

#####
# Section B: Message Type Names and Corresponding MIF File Names
#
# This section contains a list of all the target message types for
# that will be generated using caAdapter. This list contains only
# that are neither CMETs nor IMETs.
#####

PORT_MT030001=PORT_MT030001UV01.mif
PORR_MT040011=PORR_MT040011UV01.mif

#####
# Section C: Message Type Names and Corresponding MIF File Names
#
# This section contains a list of all the supporting message types
# that will be generated using caAdapter during testing. This list
# messages that are CMETs and IMETs.
#####

# IMETs:
PORR_MT040002=PORR_MT040002UV01.mif
PORR_MT040005=PORR_MT040005UV01.mif

# CMETs:
COCT_MT070000=COCT_MT070000UV01.mif
COCT_MT090000=COCT_MT090000UV01.mif
COCT_MT090100=COCT_MT090100UV01.mif

```

Figure 5.15 Update cmet-file.properties file

- d. In general, each MIF file name has the following format: {message type} + 'UV##.mif'. For example, 'PRPA_MT101001UV02.mif' means the second usage version of the MIF file of the **PORR_MT040011** message type.
3. Add, or ensure that, the appropriate MIF file and its related CMET MIF files are included in the zipped file {caAdapter_Home} \ lib \ mif.zip. (Figure 5.16)

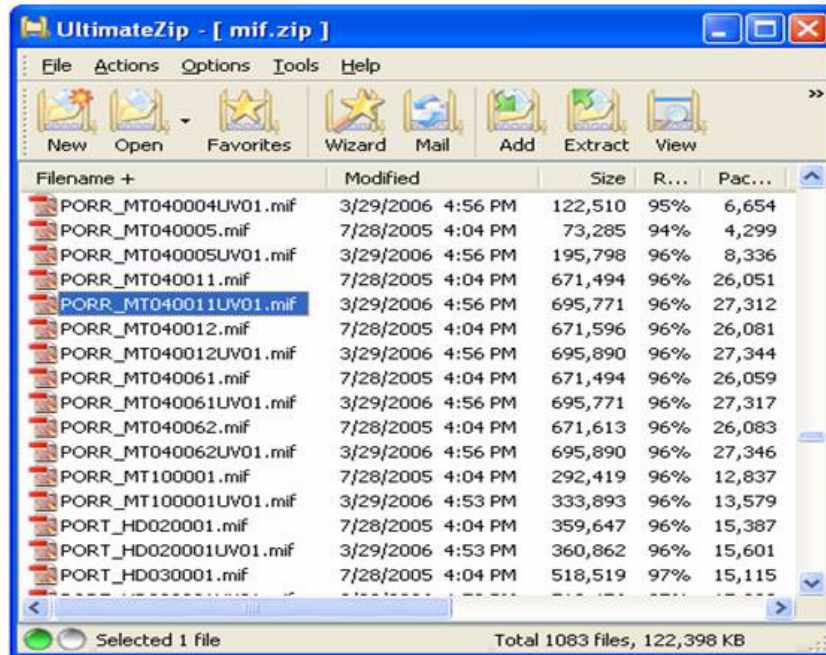


Figure 5.16 Place the MIF file into the mif.zip file

- Place the appropriate schema in the correct directory: **{caAdapter_Home} \ schema \ multicacheschemas** (Figure 5.17). Verify that the usage version of the XML schema file is same as that of the MIF file. For example, when the MIF file name is 'PRPA_MT101001UV02.mif', the corresponding XML schema file name must be 'PRPA_MT101001UV02.xsd'.

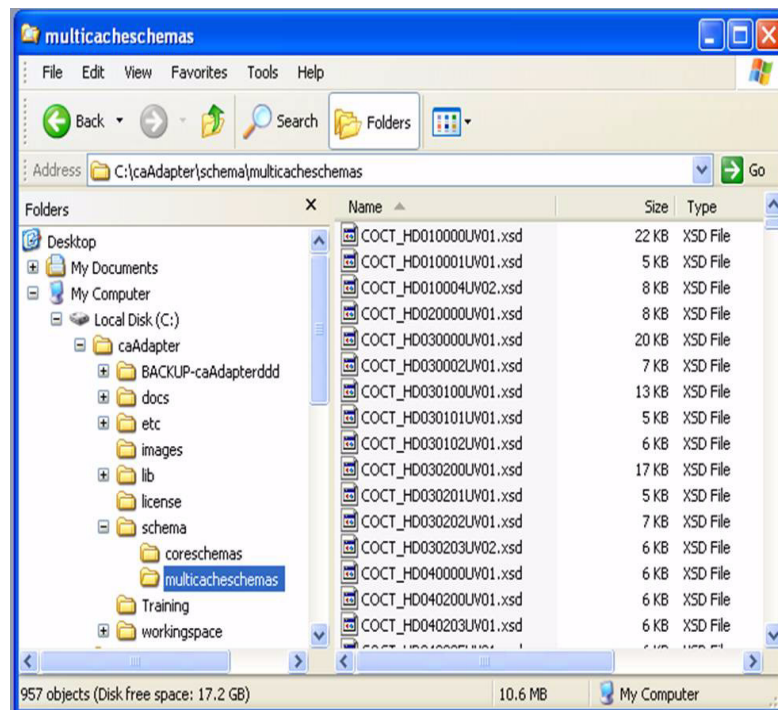


Figure 5.17 Schema file Location

5. If you already have the caAdapter Mapping Tool running, shut it down and restart it.

Updating the Properties of an HL7 v3 Specification

Select an element in the tree structure to display its properties in the **HL7 v3 Specification Properties** in the right-hand panel ([Figure 5.10](#)). These properties reflect the values found in the MIF. The properties panel includes fields that may not apply to the level or type of the element displayed. For example, a data type field, such as an **id.root**, would not have a **CMET** name and would not be **Mandatory**. For attributes that apply to an element's type, the associated data is displayed. This data also helps determine when an element may have multiples (based on association and attribute cardinality or collection data types).

Properties that are not grayed-out are allowed to be updated. For example, **User-defined Default Value** is a field that is editable and the **Data Type** field is editable when the element has an abstract data type of **ANY** or **QTY**. [Table 5.4](#) contains the properties and a description of each.

<i>Property</i>	<i>Description</i>
Element Name	Name of the element.
Element Type	Type of the element. Can be clone, attribute or data type.
Element Parent	Name of the element's parent in the tree hierarchy.
Cardinality	Specifies the minimum and maximum number of occurrences (repeats) of the field/association. For example, 1..* implies the minimum number of occurrences (repeats) is 1 whereas the maximum number of occurrences (repeats) is unlimited.
Mandatory	Valid values are: <ul style="list-style-type: none"> • M (mandatory) - means that the field/association must be present in the message, otherwise the message is invalid and is non-conformant. For Mandatory fields, HL7 sets the minimum cardinality to 1 (a value must be present). • Blank (not mandatory) - means that the field/association does not need to be present in the message.

Table 5.4 HL7 v3 property descriptions

<i>Property</i>	<i>Description</i>
Conformance	<p>Valid values are:</p> <ul style="list-style-type: none"> • R (required) - means that the sending application must support this field/association. If the data is available, then the field/association is included in the message. If the minimum cardinality is 0 and the data is not available, the field/association may be omitted from the message and still be conformant. If the minimum cardinality is 1 and the data is not available, a Null Flavor is required (see below). • NP (not permitted) - means that the field/association is not included in the message schema and never included in a message instance. • Blank (optional) - means that the field/association may/may not be sent and support for this field/association is not required of sending applications. Conformance for this element is to be negotiated on a site-specific basis.
RIM Source	Identifies the class from which the attribute or association originates.

Table 5.4 HL7 v3 property descriptions

Property	Description
Abstract	<p>This is a boolean assigned to classes or associations in a generalized-specialized hierarchy, which becomes a choice in a MIF. If the value is true, then this type may not appear in message instances - the implementation of an abstract element must specialize it to one of its subclasses. Examples include <i>ANY</i> (Any data type) or <i>QTY</i> (Quantity).</p> <p>Following are the valid data types for <i>ANY</i>:</p> <ul style="list-style-type: none"> • AD - Postal address • BL - Boolean • CD - Concept Descriptor • CE - Coded with Equivalents • CR - Concept Role • CS - Coded Simple Value • CV - Coded Value • ED - Encapsulated Data • EN - Entity Name • II - Instance Identifier • INT - Integer Number • IVL - Interval • OID - Object Identifier • ON - Organization Name • PN - Person Name • PQR - Physical Quantity Representation • PQ - Physical Quantity • REAL - Real numbers • RTO - Ratio • SC - Character String that may have a code attached • ST - Character String • TEL - Telephone number • TS - Point in Time <p>Following are the valid data types for <i>QTY</i>:</p> <ul style="list-style-type: none"> • TS - Point in Time • MO - Monetary Amount • RTO - Ratio • PQ - Physical Quantity • INT - Integer Number • REAL - Real numbers

Table 5.4 HL7 v3 property descriptions

<i>Property</i>	<i>Description</i>
Data Type	<p>Following are the valid data types:</p> <ul style="list-style-type: none"> • AD - Postal address • ADXP - Address part • BAG - Bag (collection data type) • BL - Boolean • CD - Concept Descriptor • CE - Coded with Equivalents • CR - Concept Role • CS - Coded Simple Value • CV - Coded Value • ED - Encapsulated Data • EN - Entity Name • ENXP - Entity Name Part • II - Instance Identifier • INT - Integer Number • IVL - Interval • MO - Monetary Amount • OID - Object Identifier • ON - Organization Name • ONXP - Organization Name Part • PN - Person Name • PQR - Physical Quantity Representation • PQ - Physical Quantity • REAL - Real numbers • RTO - Ratio • SC - Character String that may have a code attached • SET - Set (collection data type) • ST - Character String • TEL - Telephone number • TS - Point in Time
HL7 Default Value	This is the value specified in the HL7 MIF for this attribute. It is integral to understanding the semantics of the HL7 message. It cannot be changed without the message becoming HL7 non-compliant and is not editable in the caAdapter Mapping Tool.
HL7 Domain	Vocabulary Domain Specification. This is the set of all concepts that can be taken as valid values in an instance of a coded attribute or field, a constraint applicable to code values. Vocabulary Domains are managed and approved by HL7.
Coding Strength	<p>Valid values are:</p> <ul style="list-style-type: none"> • CWE (coded with extensions) - means that the code set can be expanded to meet local implementation needs) • CNE (coded no extensions) - means that the code set cannot be expanded).
CMET	This is displayed in an HL7 MIF and in the caAdapter HL7 v3 specification properties to indicate that this portion of the message comes from a CMET. CMETs are reusable messages that represent the standardization of a common set of information needed by many larger messages or even many domains.

Table 5.4 HL7 v3 property descriptions

<i>Property</i>	<i>Description</i>
User-defined Default Value	This is a value assigned to an attribute by the analyst who defines the HL7 v3 specification. It is used when the source data does not or might not provide a value for a given attribute. If a mapping is made to an attribute that has a user-defined default value as well, the value coming from the source data is used unless it is null, in which case the user-defined default value is used instead.

Table 5.4 HL7 v3 property descriptions

Defining inlineText

The data type field named **inlineText** is caAdapter's way of referring to text that appears between XML tags as opposed to being a value assigned to an XML attribute. For example, in the following XML, "Rockville" is the inlineText: <city>Rockville</city>. But in the following XML, "WP" is the value for an XML attribute: <addr use="WP">. Enter the required text for such an attribute in the **User-defined Default Value** field. This is more commonly used for address parts and name parts.

Defining Units of Measure

Some HL7 v3 data types contain units of measure properties. These units of measure must match those specified in the Unified Code for Units of Measure (UCUM). The UCUM is a code system intended to include all units of measure being contemporarily used in international science, engineering, and business. For a complete list, see <http://aurora.regenstrief.org/UCUM/ucum.html>.

Defining Default Data

User-defined default values are pre-defined constants for attributes and data type values. These defaults allow you to assign values for attributes that may not be available from the source data. For example, if the root for all user ids is common across the organization, this value can be entered in the target specification. HL7 structural attributes and other elements that have their values fixed by the HL7 v3 standard cannot have user-defined default values.

User-defined default values are overridden by values mapped from a data source. While required attributes should always be populated with either an HL7-defined or user-defined default value, optional ones are only populated when a map is present for that data type. [Table 5.5](#) shows the expected behavior for attributes that are mapped with a CSV value, mapped with a null CSV value and unmapped data types.

	<i>Mapped</i>	<i>Mapped Null</i>	<i>UnMapped</i>
Optional	CSV Value	CSV Value	Not Populated
Required	CSV Value	Default Value	Default Value
Mandatory	CSV Value	Default Value	Default Value

Table 5.5 Default Value Behavior

Mandatory means that the value may not be NULL, unless its container (clone, attribute, etc.) is NULL. Required means values must be supported but they may be NULL.

Defining Object Identifiers (OIDs)

HL7 v3 artifacts use OIDs to identify coding schemes and identifier namespaces. A full list of HL7 assigned OIDs, and the details of the registered schemes, is available from the **OID Registry** page of the www.hl7.org web site (**Members Only** section). There are two types of OIDs that can be used within an HL7 message:

1. HL7 OIDs
2. Existing OIDs

In HL7, you assign an OID by assigning it within the appropriate branch of the HL7 OID root (the HL7 OID root is 2.16.840.1.113883). Be sure to check that the scheme you are assigning doesn't already have an OID assigned to it within the HL7 OID hierarchy. The process of registering an existing OID with HL7 is adding an OID and its descriptive data to a central registry. The OID doesn't have to be within the HL7 root OID or any specific other root or branch OID. Once a scheme has been registered, no other OIDs that identify the same scheme can be registered.

Examples of OIDs used in HL7 are:

- Coding schemes created by professional bodies that are intended to be used widely (for example, Systematized Nomenclature of Medicine (SNOMED), Logical Observation Identifiers, Names and Codes (LOINC), International Classification of Diseases (ICD), etc.) need to be registered by HL7 International.
- Civil namespaces (identification schemes) such as driver's license, social security numbers need to be registered by the appropriate HL7 Affiliate.

In the HL7 v3 specification, when you have a **codeSystem** data type, you must assign the OID in the **User-defined Default Value** field (see [Figure 5.18](#)) or you must have a map.

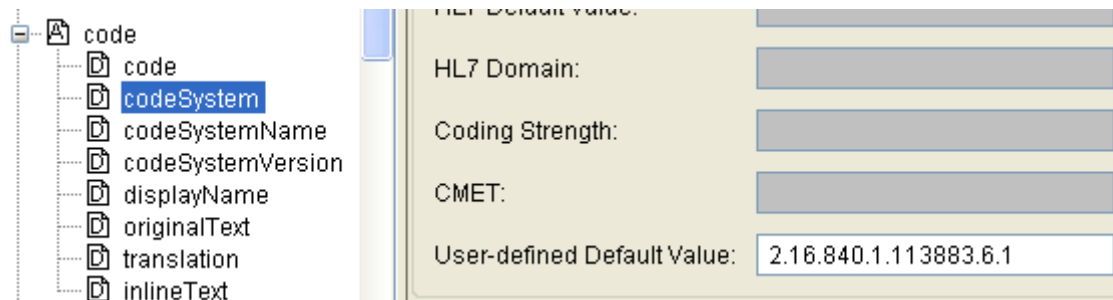


Figure 5.18 OID entered in the User-defined Default Value field

Using Structural Attributes of the HL7 RIM Classes

The default values for all structural attributes are displayed as property values in the properties panel which are required as part of the HL7 message specification. The structural attributes of HL7's main RIM classes are used to define the high-level semantics of HL7 messages and have an HL7-controlled vocabulary. [Table 5.6](#) contains the structural attribute and its Globally Unique Identifier (GUID). Of these attributes, there is a subset whose values are fixed in the context of a given HL7 message and cannot be changed. All other structural attributes may have values defined by you for a given message, but the values must be drawn from the HL7-approved vocab-

ulary domain. These values may be entered in the **User-defined Default Value** field in the **HL7 v3 Specification Properties** panel.

<i>Structural Attributes</i>	<i>GUID</i>
Act.moodCode	2.16.840.1.113883.5.1001
Act.classCode	2.16.840.1.113883.5.6
Act.negationInd	
Act.levelCode	
ActRelationship.typeCode	2.16.840.1.113883.5.1002
ActRelationship.ActRelationshipInversionInd	
ActRelationship.contextControlCode	2.16.840.1.113883.5.1057
ActRelationship.negationInd	
ActRelationship.contextConductionInd	
Entity.classCode	2.16.840.1.113883.5.41
Entity.determinerCode	2.16.840.1.113883.5.30
Participation.typeCode	2.16.840.1.113883.5.90
Participation.contextControlCode	2.16.840.1.113883.5.1057
Role.classCode	2.16.840.1.113883.5.110
Role.negationInd	
RoleLink.typeCode	2.16.840.1.113883.5.107

Table 5.6 Structural attributes of the HL7 RIM classes

Adding Clones to the HL7 v3 Specification

The ability to add or remove a clone is the way the caAdapter Mapping Tool accommodates optional associations in an HL7 message. Due to the size and complexity of numerous associations, nodes are initially created in the tree for mandatory associations only. You must customize the HL7 v3 specification to include the associations that are needed for your particular mapping plans by using the **Add Clone** and **Remove Clone** options.

Perform the following steps to add associations or expand one recursive child generation at a time.

1. Right-click on an element name with an optional or recursive relationship and select **Add Clone** to display the **Clone List** dialog box (see [Figure 5.19](#)).

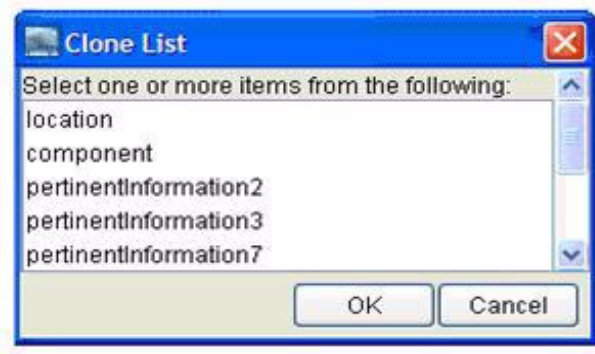


Figure 5.19 Clone List dialog box

2. In the **Clone List** dialog box, select one or more of the unused clones and click **OK**. The corresponding nodes are added to the tree in the left-hand panel.
3. There may be further optional associations available on the new clones just added. This is the case with a recursive association, where you could continue adding recursive levels to an arbitrary level as needed by using **Add Clone**.

Perform the following steps to remove clones.

1. Right-click on an element name with an optional or recursive relationship to its parent and select **Remove Clone** to display the **Clone List** dialog box (see [Figure 5.19](#)).
2. In the **Clone List** dialog box, select one or more of the unused clones and click **OK**. The corresponding nodes are deleted from the tree in the left-hand panel.

Adding and Deleting Multiples to the HL7 v3 Specification

Message elements that have either a cardinality of 0..* or 1..* and/or a data type that involves a collection (for example, SET, BAG, LIST) contain the **[Multiple]** label. The **[Multiple]** label is displayed as a numbered label to indicate the number of elements defined for that multiple (for example, **[1]**, **[2]**, etc.). These items appear in the HL7 v3 specification as simple repeats of the element. To accommodate the possible requirement of mapping more than one source element to the same target element, you must add multiples of these elements.

Perform the following steps to add multiple clones.

1. Right-click on a clone that contains a **[Multiple]** label (or a **[1]** label) and select **Add Multiple Clone** (see [Figure 5.20](#)).

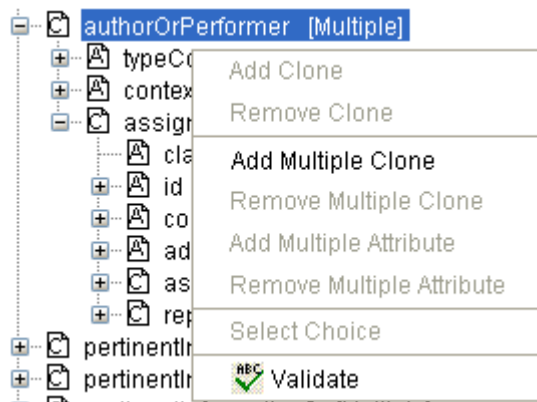


Figure 5.20 HL7 v3 specification multiple clones

2. Another element is created and the label is changed to the number of elements created.

Perform the following steps to remove multiple clones.

1. Right-click on a clone with the **[1]** label, indicating that it contains multiple numbered labels, and select **Remove Multiple Clone** (see [Figure 5.20](#)).
2. One multiple of the element is removed from the tree structure.

Perform the following steps to add multiple attributes.

1. Right-click on an attribute that contains a **[Multiple]** label (or a **[1]** label) and select **Add Multiple Attribute** (see [Figure 5.21](#)).

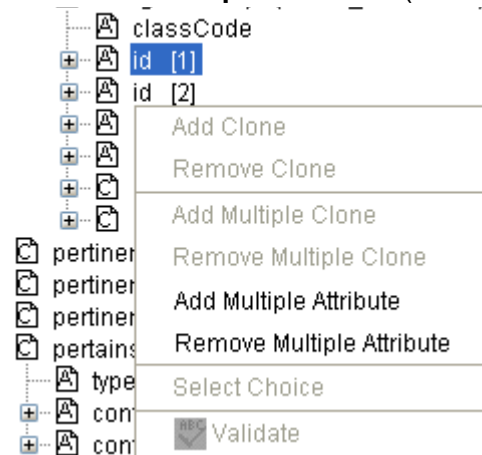


Figure 5.21 HL7 v3 specification multiple attributes

2. Another element is created and the label is changed to the number of elements created.

Perform the following steps to remove multiple attributes.

1. Right-click on an attribute with the **[1]** label, indicating that it contains multiple numbered labels, and select **Remove Multiple Attribute** (see [Figure 5.21](#)).
2. One multiple of the attribute is removed from the tree structure.

Updating Abstract Data Types in the HL7 v3 Specification

Abstract data types occur when HL7 message developers do not specify a particular data type to use when populating attributes and are indicated by a **[QTY]** or **[ANY]** label in an HL7 v3 specification. You must assign a specialized data type to the abstract element by performing the following steps.

1. Select the element name in the left-hand panel to display its properties in the **HL7 v3 Specification Properties** panel.
2. Use the drop-down list in the **Data Type** field (see [Figure 5.22](#)) to select the data type. See [Table 5.4](#) on page 42 for a list of valid data types. Click **Apply**.

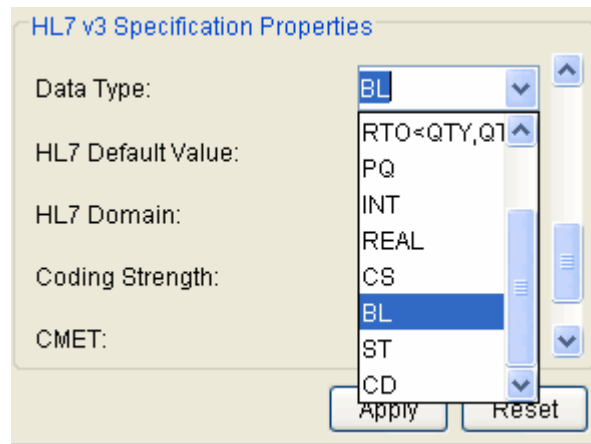


Figure 5.22 Data Type drop-down list

Using Choice Boxes in the HL7 v3 Specification

HL7 choice boxes pose a challenge in the representation of options in a mapping tool. Currently, the caAdapter Mapping Tool's limitation for the choice implementation is the ability to choose a single option to which all logical records in the source file may be mapped. The presence of a choice is indicated with a **[Selected Choice]** or **[Choice Unselected]** label.

Perform the following steps to make a choice selection for an element.

1. Right-click on an element name that contains a **[Choice - Unselected]** label and select **Select Choice** to display the **Clone List** dialog box.
2. Select one and only one clone from the displayed list and click **OK**. This creates an expandable node with the **[Selected Choice for]** label displayed beside the parent clone node (see [Figure 5.23](#)).

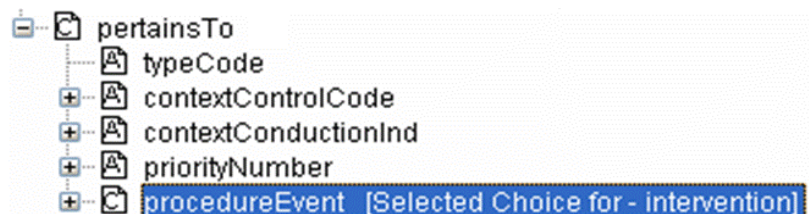


Figure 5.23 Selected choice

Note: Since a business rule for an HL7 v3 specification specifies a choice must be selected, there is no option to unselect a choice.

Validating the HL7 v3 Specification

You can validate a portion of or the entire HL7 v3 specification. A clone must be selected to perform the validation. The validation is performed on the selected clone and down the remaining tree structure.

Perform the following steps.

1. Select **File > Validate**, select the **Validate** icon from the tool bar or right click a clone and select **Validate** to perform the validation.
2. A **Message** dialog box displays ([Figure 5.24](#)) indicating the status of the validation. Click **OK**.

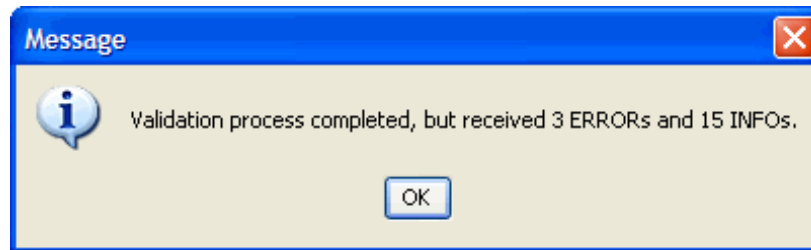


Figure 5.24 HL7 v3 specification validation

3. The messages display in the **Validation Messages** panel (see [caAdapter Mapping Tool Validation](#) on page 28).

Saving an HL7 v3 Specification

When you are finished working on the HL7 v3 specification, select **File > Save** or **File > Save As** from the menu bar or click the save icon on the tool bar to save the file. This file is portable and can be opened by the same or another user later.

Map Specification

A map is a user-defined, direct relationship between two pieces of specification elements. Using the mapping tool, you create links between source fields and target attributes and between source segments and target clones. Links between source fields and target attributes are used to represent data relationships. Links between segments and clones are used to explicitly link concepts that provide a context to the data fields/attributes and are also called container mappings. Links may also be created between source fields and input parameters of a variety of functions provided by caAdapter, and between the function's output parameters and target elements.

Business Rules

Following are the business rules for the map specification:

- It must contain a valid mapping pair (source and target files).
- The source field referenced in the map specification must exist in the source specification.

- The destination field referenced in the map specification must exist in the destination specification.
- A mandatory MIF element must have either a mapping in the map specification or an HL7-defined or user-defined default value in the HL7 v3 specification.
- Each input parameter for a function must have a mapping or a constant defined.
- Each output parameter for a function must have a mapping or a constant defined.

Step-by-Step Instructions

This section contains the step-by-step instructions to create the mappings. See [Appendix A caAdapter Example Data](#) on page 73 for detailed information on mapping scenario 8 included with the example data.

Overview of the Map Specification Tab

The map specification tab allows you to assign fields in a source specification to elements in a target specification. For the source (the CSV specification) and the target (HL7 v3 specification), the hierarchy is visually represented using an expandable/collapsible tree structure.

The map specification tab ([Figure 5.25](#)) consists of the following:

- **Two tree panels** - contains the source specification in the left-hand panel and the target specification in the right-hand panel.
- **Center mapping panel** - displays the lines that indicate the mapping between source fields and target fields and any functions that are used in the mappings.
- **Functions panel** - displays a tree of available functions.
- **Properties panel** - changes depending on the item selected in the other panels (for example, displays link properties, HL7 v3 specification data type field properties, CSV field properties, etc.).

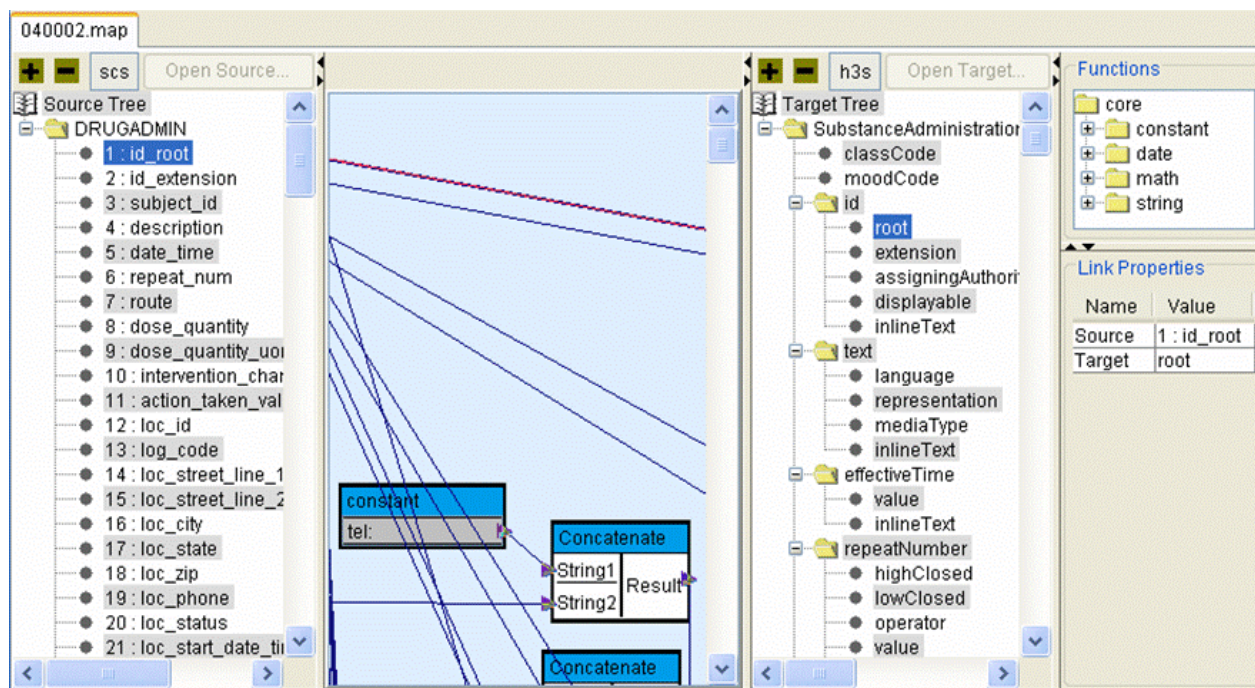


Figure 5.25 Map specification tab

The tree structures are read-only; you must make any changes to the tree structures in the source or target specification tabs. You can only define the mappings from this tab.

Warning: Adding to source or target specifications that are referenced in a map file is allowable, but editing or removing source or target elements may result in a related mapping (link) getting dropped behind the scenes or producing other unpredictable behavior.

The following sections describe how to access, create and save the map specification.

Creating and Opening a Map Specification

You must create a new or open an existing map specification.

Perform the following steps to create a new map specification.

1. Select **File > New > Map Specification** from the menu bar to open a new mapping tab with empty source and destination panels.
2. Click **Open Source** to display the **Open Source File** dialog box.
3. Select the source file and click **Open** to populate the source with its tree structure.
4. Click **Open Target** to display the **Open Target File** dialog box.
5. Select the target file and click **Open** to populate the target with its tree structure.

Perform the following steps to open an existing map specification.

1. Select **File > Open > Map Specification**. The Open Map File dialog box displays.

2. Select the map specification file and click **Open** to display the source and target trees along with any existing mappings.

Updating the Map Specification

Perform the following steps to create a mapping.

1. Select a source element and drag it to the appropriate target element. The cursor indicates (see [Figure 5.26](#)) if the source is not allowed to be mapped to the target element.

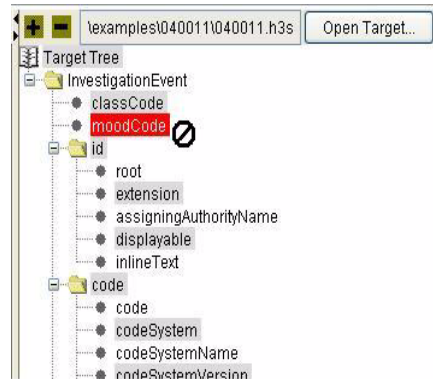


Figure 5.26 Mapping is not allowed

The cursor indicates (see [Figure 5.27](#)) when the source can be mapped to the target element. Drop the source on the target element.

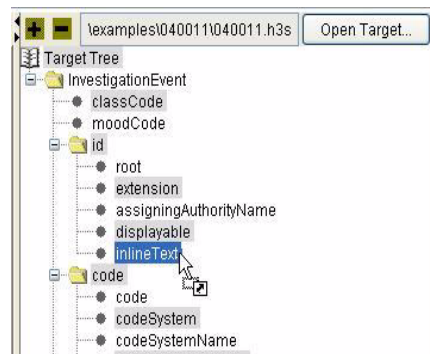


Figure 5.27 Mapping is allowed

- 2. Once a source field is mapped to a target element, a mapping line appears between them in the mapping panel. *Figure 5.28* shows a mapping line between **id_root** and **root**.

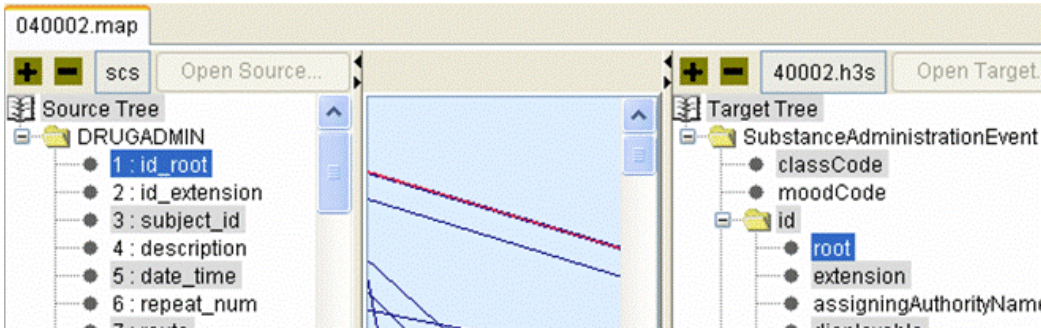


Figure 5.28 Mapping line between a source field and target element

- To delete a mapped line or a function in the center panel, perform the following steps.
- 1. Select the item you want to delete, right-click and select **Delete**.
 - 2. Click **Yes** to confirm the deletion. The selected item is deleted from the mapping.

The **Properties** panel displays information on the selected element. When you select a source element, the **CSV Field Properties** displays (see *Figure 5.29*).

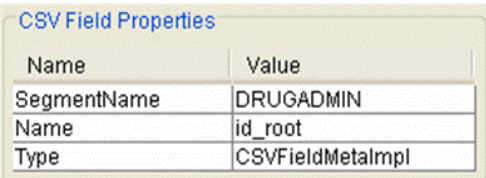


Figure 5.29 CSV Field Properties panel

When you select a mapping line, the **Link Properties** displays (see *Figure 5.30*).

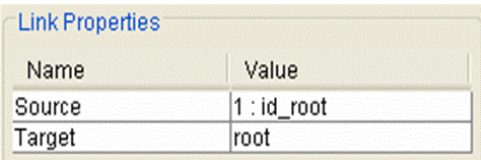


Figure 5.30 Link Properties panel

When you select a target element, the **HL7 v3 Specification Attribute Properties**, **HL7 v3 Specification Data Type Field Properties** or the **Clone Attribute Object Properties** displays. When you select a function either in the **Mapping** panel or in the **Functions** panel, the **Function Properties** display. When you select a function group in the **Functions** panel, the **Function Group Properties** display.

Using Functions in Map Specifications

The **Functions** panel (see [Figure 5.31](#)) provides a list of system defined functions that facilitate the data transformation requirement. Functions are grouped by functional categories (for example, constant, date, math, string, etc.). You may use a function in the mapping to effect a change of the source element to the target element. For example, you can use the concatenate function to add a prefix to an element.

Note: You can add your own required functions to the function library. See [Adding Functions to the Function Library](#) on page 70 for instructions.

When a function is selected in the function library, its properties information displays, such as name and number of input and output parameters, in the **Function Properties** panel (see [Figure 5.31](#)).

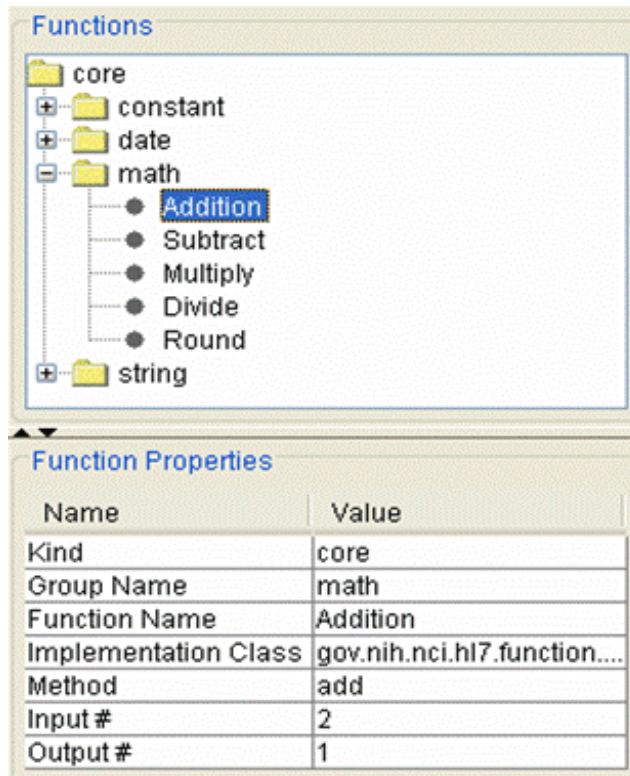


Figure 5.31 Functions in mapping specification

Perform the following steps to include a function in your mapping specification.

1. Add a function to the mapping panel. Select a function in the **Functions** panel, right-click in the center panel and select **Add Function**, or drag-and-drop the required function from the **Functions** panel to the mapping panel. Move this function box around the mapping panel as convenient to attach the mapping lines.

2. Drag-and-drop the source field(s) onto the input parameters. *Figure 5.32* shows the selected field **text** being dropped as an input to the **Initcap** function.

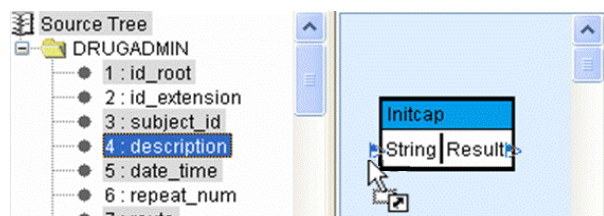


Figure 5.32 Adding an input to a function

3. Drag-and-drop the target field onto the output parameter. The mapping lines go from the source fields into the function box and out of the function box to the target fields.

Editing a Constant Function

Perform the following steps to edit a constant function.

1. Select a constant function in the mapping panel, right-click and select **Edit Constant**.
2. In the **Edit Constant** dialog box, change the **Type** and/or **Value** for the constant and click **OK**.

Using the Date Function

The date function, **changeFormat**, uses the Java `SimpleDateFormat` class. See <http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html> for more information. *Table 5.7* shows the correct syntax for each date or time component.

Date or Time Component	Presentation	Example Pattern 1	Example Pattern 2
Year	Lowercase y	yy => 05	yyyy => 2005
Month	Uppercase M	MM => 07	MMM => JUL
Day	Lowercase d	dd => 07 or 17 (7th or 17th date of the month)	d => 7 or 17
Hour	Uppercase H or lowercase h	Using 2 PM HH => 14	hh => 02, h => 2
Minute	Lowercase m	mm => 09	m => 9
Second	Lowercase s	ss => 12	
Millisecond	Uppercase S	SSS => 002	

Table 5.7 Date formats

For example, July 7th 1988, PM 02:23:14 can be presented in the following ways:

- yyyyMMddHHmmss => 19880707142314
- dd-MMM-yyyy, HH:mm:ss => 07-JUL-1988, 14:23:14

- MM/dd/yy => 07/07/88

Refreshing the Map Specification Tab

Click the **Refresh** button on the tool bar to check and update the associated CSV specification or HL7 v3 specification used in the mapping panel. If changes to the mapping panel were required, an information message (see [Figure 5.33](#)) displays.

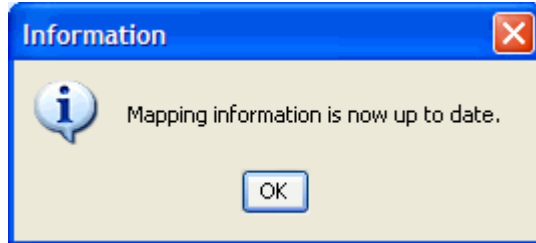


Figure 5.33 Mapping panel refreshed message

This option allows you to update and save the associated CSV or H3S file in their own tabs, while you are also performing the mapping between the two.

If either the CSV or H3S files are updated and saved in their own tabs, and you switch back to the mapping panel, then a dialog (see [Figure 5.34](#)) displays to notify you of the changes. You are not forced to refresh the mapping panel at this time, since you may have some pending mapping activity unsaved.

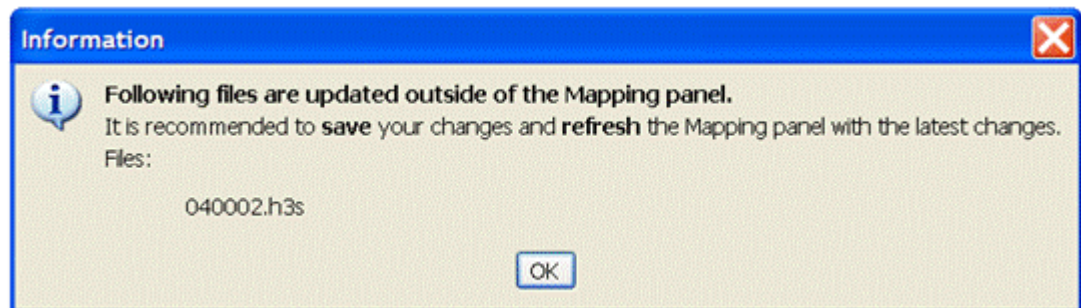


Figure 5.34 Refresh mapping panel recommendation

Validating the Map Specification

Perform the following steps to validate the map specification.

1. Select **File > Validate** or select the **Validate** icon from the tool bar to perform the validation.
2. A **Message** dialog box displays indicating the status of the validation. Click **OK**.
3. The detailed messages display in the **Validation Messages** dialog box (see [caAdapter Mapping Tool Validation](#) on page 28).

Saving a Map Specification

When you are finished working on the map specification, select **File > Save** or **File > Save As** from the menu bar or click the save icon on the tool bar to save the file. This file is portable and can be opened by the same or another user later.

Warning: The map specification has an internal reference to the full path name of the source and target specification files and those must be accurate to process the conversion or edit a map specification successfully. If you are sharing map specification files with other users, you must send all three files, the CSV specification (.scs), HL7 v3 specification (.h3s), and map specification (.map) and not just the map specification.

Generating a Map Specification Report

When a map specification tab is selected, you can generate a report on the status of the mapping specification by performing the following steps.

1. Select **Report > Generate Report** from the menu bar to display the **Select File to Save Generated Report** dialog box.
2. Enter a **File name** and click **Save**. A Report has been successfully generated message displays.

The report is an Excel spreadsheet containing the status of the mapping specification. The report contains up to six worksheets (tabs) within the generated report. Under the mapped category, it contains the mapping status between:

- Source and target - Mapped(Source_Target)
- Source and function - Mapped(Source_Function)
- Function and target - Mapped(Function_Target)
- Function and function - Mapped(Function_Function)

Under the unmapped category, it contains the unmapped:

- Source - Unmapped_Source
- Target - Unmapped_Target

HL7 v3 Message

The HL7 v3 message is the end goal in using the mapping tool. XML HL7 message instances are created using the map specification.

Business Rules

Following are the business rules for creating an HL7 v3 message:

- You must have data in a CSV format.
- The map specification must be valid.
- The source and target specifications used to create the map must be located in the same directory as they were when the map specification was created (or the map specification must have been edited to point to the new location of these files if they were moved). The map specification uses the references to these files as it converts the data into the new format.

Step-by-Step Instructions

This section contains the step-by-step instructions to generate the HL7 v3 message.

Note: There is no **File > Open** option that corresponds to HL7 v3 messages since you always want to generate fresh messages based on the current selection of source and map files.

Overview of the HL7 v3 Message Tab

The purpose of the HL7 v3 Message tab is to allow you to generate and view the XML instances and messages converted from a data file and map specification. Each data file may have one or more logical records which result in a corresponding number of XML instances (or more depending on the structure of the mapping). The user interface allows you to navigate between the instances. The HL7 v3 message tab (see [Figure 5.35](#)) contains the following four panels:

- Regenerate and navigable buttons
- Name of data file and map specification used
- Scrollable text fields for XML instances
- Scrollable text fields for validation messages

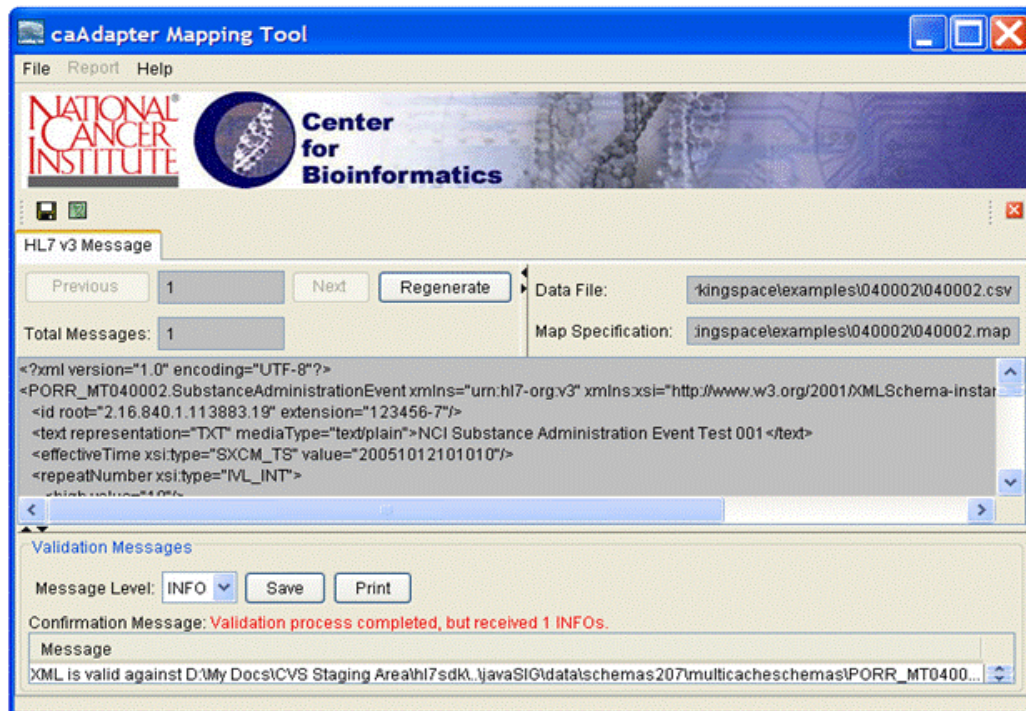


Figure 5.35 HL7 v3 Message tab

Starting the Conversion Process

Perform the following steps to convert a data file into an HL7 v3 message.

1. Select **File > New > HL7 v3 Message** from the menu bar to display the **HL7 v3 Message** dialog box (see [Figure 5.36](#)).



Figure 5.36 HL7 v3 Message dialog box

2. Click **Browse** next to **Data File** to display the **Open Data File** dialog box.
3. Select the data file you want to use in the conversion process and click **Open**.
4. Click **Browse** next to **Map Specification** to display the **Open Map Specification** dialog box.
5. Select the map specification file you want to use in the conversion process and click **Open**.
6. Click **OK** to generate HL7 v3 messages from the selected files.
7. Given the underlying data and mapping structure, it could take a long time to complete the HL7 v3 message generation task. If the system estimates that it will take longer than ten seconds (as defined and is configurable in the source distribution), then the **Question** dialog displays as shown in [Figure 5.37](#). Click **Yes** to start the process or click **No** to abort the process given the estimated time.

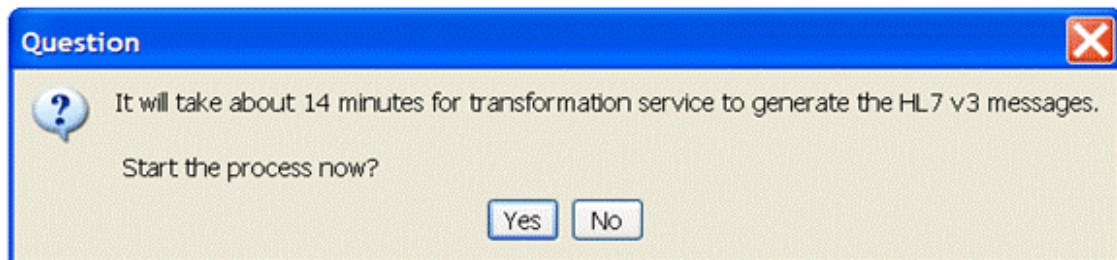


Figure 5.37 HL7 v3 message generation confirmation

8. After a **Yes** confirmation, the process starts and a progress dialog box displays (see [Figure 5.38](#)).

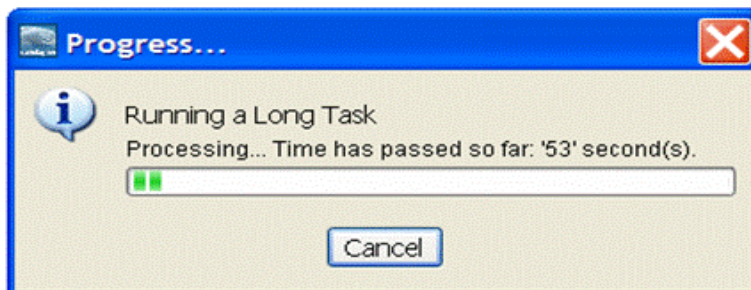


Figure 5.38 HL7 v3 message generation progress dialog

9. Once the process starts, you can cancel the process by clicking **Cancel**. If cancelled, the underlying generation process is terminated and an information dialog displays (see [Figure 5.39](#)).

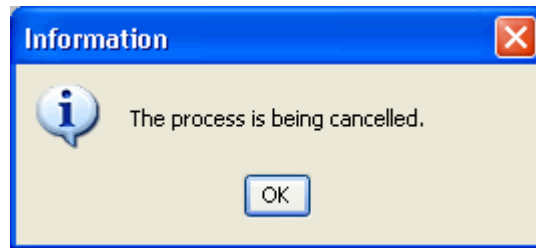


Figure 5.39 HL7 v3 message generation cancelled

10. A message displays (see [Figure 5.40](#)) after the overall process completes. Click **OK**.

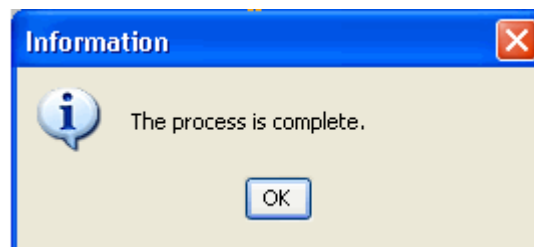


Figure 5.40 HL7 v3 message process complete

11. The **HL7 v3 Message** tab displays.

Using the Basic Features of the HL7 v3 Messages Tab

The two main features of the **HL7 v3 Message** tab (see [Figure 5.41](#)) are the two scrollable text fields containing an XML instance and the associated error, warning and/or informational messages generated during the conversion process.

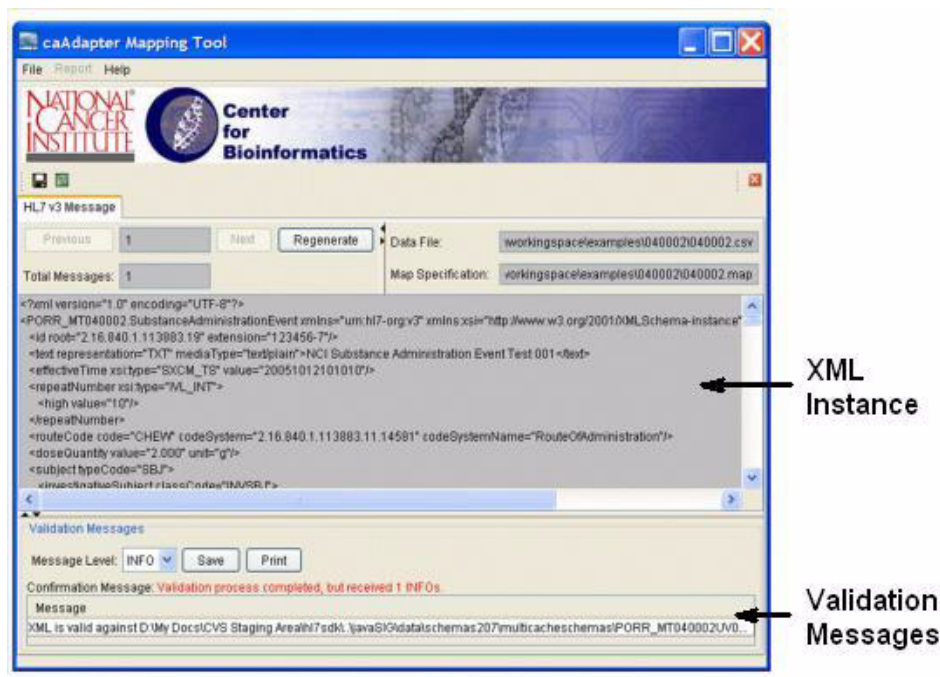


Figure 5.41 HL7 v3 Message tab

Click the **Previous** and **Next** buttons to cycle through the messages one at a time. As the messages change, the validation messages change. See [caAdapter Mapping Tool Validation](#) on page 28 for more information on the validation messages. Click the **Regenerate** button to regenerate the messages from scratch using the same data file and map specification.

Saving an HL7 v3 Message

Select **File > Save** or **File > Save As** from the menu bar or click the save icon on the tool bar to save the HL7 v3 message. If there is more than one instance of a message, then the files are saved with number extensions (for example, 090102_1.xml, 090102_2.xml, 090102_3.xml).

CHAPTER 6

CAADAPTER FILE TYPES

This chapter includes the different file types and their formats used by caAdapter.

Topics in this chapter include:

- [*caAdapter File Formats and Locations*](#) on this page
- [*CSV Data File*](#) on page 66
- [*CSV Specification*](#) on page 66
- [*HL7 v3 Specification*](#) on page 68
- [*Function Specification*](#) on page 69
- [*Map Specification*](#) on page 70
- [*HL7 v3 Message*](#) on page 71

caAdapter File Formats and Locations

caAdapter uses a variety of files in its APIs and mapping tool. [*Table 6.1*](#) contains the files and extensions used by caAdapter.

<i>File Type</i>	<i>Extension</i>
CSV specification	.scs
HL7 v3 specification	.h3s
HL7 v2 specification	.h2s
Database specification	.dbs
Function library specification	.fls
Map specification	.map
HL7 v3 message	.xml

Table 6.1 File extensions

Note: Manual editing of files is not supported and highly discouraged since they rely on UUID.

Warning: The map specification has an internal reference to the full path name of the source and target specification files and that must be accurate to process the conversion or to edit a map specification successfully. Though it is not recommended, the map specification can be manually edited to change the file path for the source and target specification if necessary. If you are sharing map specification files with other users, you must send all three files, the CSV specification (`.scs`), HL7 v3 specification (`.h3s`), and map specification (`.map`) and not just the map specification.

CSV Data File

It is an assumption for this version of the mapping tool that the source data systems provide data in CSV flat file formats with the following characteristics:

- File contents are organized into multi-line logical records.
- Each line, called a segment, begins with an identifier, called a segment name, and is terminated by a new-line character.
- Each segment has one or more data items, called fields, which follow the segment name and terminates by commas (except for the last field on the line that uses the segment terminator).
- Segments may occur more than once in the same logical record, except for the first or root segment, which always indicates the beginning of a new record.
- Segments are related to one another in a parent-child hierarchy that documents the one-to-many nature of the association between related data items.
- A CSV file may have one or more logical records. Each of these is terminated by the beginning of the next record (a new root segment) or the end of file.
- The intention is that each logical record will become one single HL7 v3 XML message instance.

CSV Specification

CSV specification describes the structure of a CSV instance. In essence, it is a CSV specification in the same way an XSD is a specification of an XML instance. The CSV specification is based on common concepts found in EDI, CSV and HL7 v2.x-related files. To document this structure, the CSV specification uses an XML format that has three main elements:

1. `<csvMetadata>`
2. `<segment>`
3. `<field>`

There can only be one root `<segment>`, but within it there can be any number of dependent `<segment>` elements and any number of `<field>` elements. All `<field>` elements have a column number assigned which corresponds to the second, third, etc. column in the CSV file (the first is the segment name which is considered column 1). The field names are informational and are not used in the mapping file where only the segment name and column number is referenced.

Following is a CSV specification file (090102.scs) example.

```
<?xml version="1.0" encoding="UTF-8"?>
<csvMetadata uuid="320c0770-4cfc-40f9-b21a-01536d7b9229" version="1.3">
  <segment name="PERS" uuid="ea1d418a-4b5a-4743-895d-7c8383a46fb7">
    <segment name="PERSID" uuid="505e2c3e-f129-4952-813f-dfccb1059148">
      <field column="1" name="Root" uuid="4d72fde8-aaed-4c68-b1b5-
e3ece792ae48"/>
      <field column="2" name="Extension" uuid="76db78b3-aaaa-46c5-953e-
c41eb0a06746"/>
    </segment>
    <segment name="PERSAD" uuid="73a04b7b-cd61-4f5e-b9c7-9e8aaa36ad56">
      <field column="1" name="Street 1" uuid="5a99239a-4523-4d38-bb47-
1d2e819d1a7d"/>
      <field column="2" name="Street 2" uuid="f40fcel1f-8783-44cb-b4da-
7f83e87e8f28"/>
      <field column="3" name="City" uuid="daf9eabc-8eed-4b46-8f4a-
9b8b701d041a"/>
      <field column="4" name="State" uuid="ad4e97a3-d53a-424e-9a55-
dddc52396883"/>
      <field column="5" name="Zip Code" uuid="c6fd917e-6cef-42ac-a951-
3d1a3e022522"/>
    </segment>
    <segment name="ORGID" uuid="4b94eb59-3213-4c6c-a3d8-1eddf1dc2751">
      <field column="1" name="Root" uuid="b0f0c6b1-5a42-450a-a258-
71bf6cf72c89"/>
      <field column="2" name="Extension" uuid="367fb803-40ed-4c07-9fab-
7b81f7bf9423"/>
    </segment>
    <segment name="ORGNM" uuid="bc595334-0cd9-47ae-8f53-7a6f769f6e79">
      <field column="1" name="Name" uuid="088d9fa5-943c-4913-a261-
51d2c77624b3"/>
    </segment>
    <segment name="ORGAD" uuid="2d1c2b66-e7da-4ff7-af49-e9acd87483bd">
      <field column="1" name="Street 1" uuid="7fe588e4-1a13-426e-82be-
82d85ee84e02"/>
      <field column="2" name="Street 2" uuid="c6f373dd-74ca-4a08-b88f-
2b32fa165ca8"/>
      <field column="3" name="City" uuid="9debfe68-acc6-4f96-b4d6-
128ab2c96860"/>
      <field column="4" name="State" uuid="5148784f-71c2-4b3b-be98-
e6b0aed0775d"/>
      <field column="5" name="Zip Code" uuid="c4576a69-cc7d-433a-9e4c-
8636be775214"/>
    </segment>
    <field column="1" name="First Name" uuid="74d508af-b486-40c9-b83e-
409bc582d10b"/>
    <field column="2" name="Last Name" uuid="7fe87cdf-2498-4108-bd0d-
01ede279b917"/>
    <field column="3" name="Middle Initial" uuid="27d9f78b-a6e8-40ca-ab33-
b271f60738c5"/>
    <field column="4" name="Job Code" uuid="1d31ec7b-6e33-4895-be9b-
0ba657258118"/>
    <field column="5" name="Department Id" uuid="162e800f-efa3-4364-9102-
e5e76b519409"/>
  </segment>
</csvMetadata>
```

HL7 v3 Specification

The HL7 v3 specification, used to define the HL7 v3 metadata information, is based largely on the MIF for the target HL7 message. It uses four main types of nested elements:

1. <hl7v3meta>
2. <clone>
3. <attribute>
4. <datatypeField>

Following is part of an HL7 v3 specification file (090102.h3s) example. See the {home directory}\workingspace\examples\090102 for the entire file.

```
<?xml version="1.0" encoding="UTF-8"?>
<hl7v3meta messageId="COCT_MT090102" version="V1.3">
  <clone cloneName="AssignedPerson" cardinality="1..1" uuid="63a0437d-784a-
44a0-8c34-6228e2ef041b">
    <attribute uuid="0ff560b0-5111-4802-9b06-c8879e285aaa" name="classCode"
cardinality="1..1" isMandatory="true" conformance="R" codingStrength="CNE"
domainName="RoleClass"/>
    <attribute uuid="557738e0-957d-4b54-a244-6903552d1b02" name="id" cardi-
nality="1..*" datatype="II" isMandatory="true" conformance="R">
      <datatypeField name="root" uuid="916fe083-eaee-48eb-8689-
b387247aa742"/>
      <datatypeField name="extension" uuid="5649498a-285f-4bdd-9bb4-
f17e64d7cd26"/>
      <datatypeField name="assigningAuthorityName" uuid="02428c15-c95d-40e3-
bfb5-d513e8195bcc"/>
      <datatypeField name="displayable" uuid="fe6422bb-b662-4dcc-a7d5-
78dfefe75cfd"/>
      <datatypeField name="inlineText" uuid="0a890545-b189-4b5d-b8dd-
c043b9756228"/>
      .
      .
      .

    </attribute>

    <attribute uuid="0e1a691c-1bad-424b-a859-d1ffe5df3cc7" name="city"
cardinality="0..*">
      <datatypeField name="inlineText" uuid="467f7189-188a-4777-a347-
93c491f626d4"/>
    </attribute>
    <attribute uuid="80deac68-d217-4713-9672-47a29f9f5d75" name="house-
Number" cardinality="0..*">
      <datatypeField name="inlineText" uuid="43fedd4e-445e-48d6-9224-
fea6dbf41e64"/>
    </attribute>
  </clone>
</clone>
</hl7v3meta>
```

Function Specification

Function Specification Overview

The function specification is used as a guide for function objects to read the function specification and determine what objects to call to execute a function (for example, concatenation). The function specification also stores data points for rendering by a function graphical representation within the mapping tool. It uses the following types of nested elements:

1. <function>
2. <group name>
3. <function name>
4. <inputs>
5. <datapoint>
6. <outputs>

Following is an example of a function specification file (`core.flx`). See the `{home directory}\map\functions` directory for the entire file.

```
<?xml version="1.0"?>
<functions>
  <group name="date" uuid="e44d7420-09db-11da-8cd6-0800200c9a66">
    <function name="changeFormat" uuid="ffd2bde0-09db-11da-8cd6-0800200c9a66">
      <inputs>
        <datapoint pos="0" name="fromFormat" datatype="string"
          uuid="ffd2bde1-09db-11da-8cd6-0800200c9a66"/>
        <!--<datapoint pos="1" name="toFormat" datatype="string"
          uuid="ffd2bde2-09db-11da-8cd6-0800200c9a66"/> -->
        <datapoint pos="1" name="dateIn" datatype="string"
          uuid="ffd2bde3-09db-11da-8cd6-0800200c9a66"/>
      </inputs>
      <outputs>
        <datapoint pos="0" name="dateOut" datatype="string"
          uuid="ffd2bde4-09db-11da-8cd6-0800200c9a66"/>
      </outputs>
      <implementation classname="gov.nih.nci.hl7.function.DateFunction"
        method="changeFormat"/>
    </function>
  </group>
  <group name="math" uuid="0b244070-fe0f-11d9-8cd6-0800200c9a66">
    <function name="Addition" uuid="376e87b0-fe11-11d9-8cd6-0800200c9a66">
      <inputs>
        <datapoint pos="0" name="Value1" datatype="int"
          uuid="ab38a100-fe0f-11d9-8cd6-0800200c9a66"/>
        <datapoint pos="1" name="Value2" datatype="int"
          uuid="ab38a101-fe0f-11d9-8cd6-0800200c9a66"/>
      </inputs>
      <outputs>
        <datapoint pos="0" name="Sum" datatype="int"
          uuid="ab38c810-fe0f-11d9-8cd6-0800200c9a66"/>
      </outputs>
    </function>
  </group>
</functions>
```

```
<implementation classname="gov.nih.nci.hl7.function.MathFunction" method="add"/>
</function>
<function name="Subtract" uuid="be9f2df1-fe1d-11d9-8cd6-0800200c9a66">
  <inputs>
    <datapoint pos="0" name="Value1" datatype="int"
    uuid="be9f2df0-fe1d-11d9-8cd6-0800200c9a66"/>
    <datapoint pos="1" name="Value2" datatype="int"
    uuid="be9f5500-fe1d-11d9-8cd6-0800200c9a66"/>
  </inputs>
  <outputs>
    <datapoint pos="0" name="Difference" datatype="int"
    uuid="be9f5501-fe1d-11d9-8cd6-0800200c9a66"/>
  </outputs>
  <implementation classname="gov.nih.nci.hl7.function.MathFunction" method="subtract"/>
</function>
</group>
.
.
</group>
</functions>
```

Adding Functions to the Function Library

The function library provides a list of system defined functions that facilitate the data transformation requirement. Functions are grouped by its functional categories (for example, math group, string group, etc). It is required that each group has to have a unique name across the whole function library, but the name of individual function is only required to be unique within its defined group.

The design of function library encompasses some extensibility on the support of user-customized functions in the definition of the function library's XML schema. In this version of release, no GUI utility is available to allow you to register custom function libraries to the mapping tool. However, advanced software engineers can update the function library definition file, named `core.flx`, located in the `{home directory}\etc` directory, to register or replace your own function implementations. After registration, the configuration engineer needs to make sure the corresponding customized Java library is available on the classpath, so that next time the mapping tool starts, it can secure the needed Java implementation classes during the generation of HL7 v3 messages.

Map Specification

A map specification describes the relationship between components via links and/or views. It has the following main elements:

1. <components>
2. <links>
3. <views>

A component is a reference to a resource that exists in the system prior to the mapping. A function component is an algorithm between two (or more) pieces of data.

Following is a part of a map specification file (090102.map) example. See the {home directory}\workspace\examples\090102 for the entire file.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="1.3">
  <components>
    <component kind="scs" location="090102.scs" type="source" uuid="84cf18e7-950b-429e-8405-10c8ab995520"/>
    <component kind="HL7v3" location="090102.h3s" type="target"
    uuid="99bde9dc-af0f-4d70-9f3e-bb727da9aafe"/>
    <component group="string" kind="core" name="Concatenate" type="function"
    uuid="60f44548-54ad-441c-a290-830e291fc3b4"/>
    <component group="string" kind="core" name="Concatenate" type="function"
    uuid="7784cbd5-0b5c-4761-9f2f-c57dd115570c"/>
  </components>
  <links>
    <link uuid="c88ab929-ae88-4faa-aae3-b5aeb6b1fbb2">
      <linkpointer component-uuid="84cf18e7-950b-429e-8405-10c8ab995520"
      data-uuid="74d508af-b486-40c9-b83e-409bc582d10b"/>
      <linkpointer component-uuid="99bde9dc-af0f-4d70-9f3e-bb727da9aafe"
      data-uuid="59c2d448-4ec2-4441-b873-67f1943646a4"/>
    </link>
    <link uuid="3218294a-c08c-4556-9df6-80a83215c2de">
      <linkpointer component-uuid="84cf18e7-950b-429e-8405-10c8ab995520"
      data-uuid="7fe87cdf-2498-4108-bd0d-01ede279b917"/>
      <linkpointer component-uuid="99bde9dc-af0f-4d70-9f3e-bb727da9aafe"
      data-uuid="551491ef-a7c5-42ae-be6a-9314af0833b3"/>
    </link>
    .
    .
    .
  </links>
  <views>
    <view component-uuid="84cf18e7-950b-429e-8405-10c8ab995520" height="0"
    width="0" x="0" y="0"/>
    <view component-uuid="99bde9dc-af0f-4d70-9f3e-bb727da9aafe" height="0"
    width="0" x="0" y="0"/>
    <view component-uuid="60f44548-54ad-441c-a290-830e291fc3b4" height="61"
    width="85" x="168" y="126"/>
    <view component-uuid="7784cbd5-0b5c-4761-9f2f-c57dd115570c" height="61"
    width="85" x="186" y="318"/>
  </views>
</mapping>
```

HL7 v3 Message

The HL7 v3 message is the end goal of using caAdapter. It is represented in XML. Following is an example HL7 v3 message file (ExampleOutput1.xml).

```
<?xml version="1.0" encoding="UTF-8" ?>
- <COCT_MT090102.AssignedPerson xmlns="urn:hl7-org:v3" classCode="ASSIGNED">
  <id root="2.16.840.1.113883.19.1" extension="12345" />
  <id root="2.16.840.1.113883.19.2" extension="23456" />
  <id root="2.16.840.1.113883.19.3" extension="34567" />
  <code code="NRS10" codeSystem="2.16.840.1.113883.19.1" />
- <addr use="WP">
```

```
<streetAddressLine>123 Main St.Suite 500</streetAddressLine>
<city>Rockville</city>
<state>MD</state>
<postalCode>20852</postalCode>
</addr>
- <addr>
  <streetAddressLine>456 Washington BlvdSuite 1000</streetAddressLine>
  <city>Washington</city>
  <state>DC</state>
  <postalCode>20002</postalCode>
</addr>
- <assignedPerson classCode="PSN" determinerCode="INSTANCE">
- <name use="L">
  <family>Shang</family>
  <given>Lee</given>
</name>
</assignedPerson>
- <representedOrganization classCode="ORG" determinerCode="INSTANCE">
  <id root="2.16.840.1.113883.19.4" extension="1111GHHMO" />
  <id root="2.16.840.1.113883.19.5" extension="2222" />
  <name>Good Health HMO</name>
  <name>Good Health Radiology</name>
  <name>GHHMOR</name>
- <addr use="WP">
  <streetAddressLine>456 Washington BlvdSuite 1000</streetAddressLine>
  <city>Washington</city>
  <state>DC</state>
  <postalCode>20002</postalCode>
</addr>
- <addr>
  <streetAddressLine>567 Empire Ave.Suite 10000</streetAddressLine>
  <city>New York</city>
  <state>NY</state>
  <postalCode>10118</postalCode>
</addr>
</representedOrganization>
</COCT_MT090102.AssignedPerson>
```


APPENDIX

A

CAADAPTER EXAMPLE DATA

Example data is included in the caAdapter distribution. You can use the example data to become acquainted with the mapping tool or APIs before using your own data. The example data is located at the `{home directory}\workspace\examples` directory (for example, `C:\caadapter\workspace\examples`). The example data structure is shown in [Figure A.1](#).

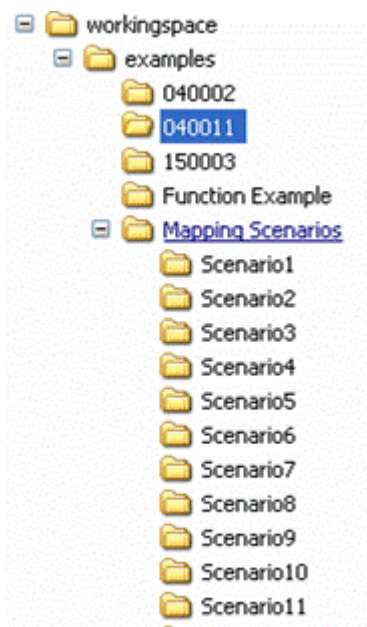


Figure A.1 Example data directory structure

The `examples` directory contains small (090102), medium (040002) and big (040011) sample data files. The big example is an ICSR example where the other directories contain a subset of this data. For more information on the mapping scenarios, see the caAdapter Mapping Rules documentation.

APPENDIX B REFERENCES

Technical Manuals/Articles

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

caBIG Material

1. caBIG: <http://cabig.nci.nih.gov/>
2. caBIG Compatibility Guidelines: http://cabig.nci.nih.gov/guidelines_documentation

caCORE Material

1. NCICB: <http://ncicb.nci.nih.gov>
2. caCORE: <http://ncicb.nci.nih.gov/core>
3. caBIO: <http://ncicb.nci.nih.gov/core/caBIO>
4. caDSR: <http://ncicb.nci.nih.gov/core/caDSR>
5. EVS: <http://ncicb.nci.nih.gov/core/EVS>

HL7 Concepts and Material

1. HL7: <http://www.hl7.org/>

2. HL7 Java Special Interest Group (JavaSIG): <http://www.hl7.org/Special/committees/java/index.cfm>
3. HL7 Tutorial: http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter/HL7_Tutorial
4. caAdapter: <http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter>
5. HL7 Reference Information Model: <https://www.hl7.org/library/data-model/RIM/C30202/rim.htm>
6. HL7 Vocabulary Domains: <http://www.hl7.org/library/data-model/RIM/C30123/vocabulary.htm>
7. HL7 Version 3 Standard: <http://www.hl7.org/v3ballot/html/welcome/environment/index.htm>
8. UCUM: <http://aurora.regenstrief.org/UCUM/ucum.html>

Software Products

1. Java: <http://java.sun.com>
2. Ant: <http://ant.apache.org/>

APPENDIX

C

CAADAPTER GLOSSARY

Acronyms, objects, tools and other terms referred to in the chapters or appendixes of this caAdapter guide are described in this glossary.

<i>Term</i>	<i>Definition</i>
.csv	Comma-Separated Values. This is the extension for a flat file with rows of data where data items on each line are separated from each other with commas.
.fls	Function Library Specification. This is the file extension for a file containing metadata describing the functions available for use in the mapping tab.
.h3s	HL7 v3 Specification. This is the extension for a file containing metadata describing the structure of the relevant portions of the HL7 v3 MIF that is used in the mapping.
.map	This is the file extension for the file containing the metadata describing what source elements (from the.scs file) map to the what target elements (from the h3s file) and what functions (if any) are used in the mappings.
.scs	Segmented CSV Specification. This is the extension for a file containing metadata describing the structure of the CSV file.
.xml	This is the file extension for the files that are generated by converting CSV data into HL7 v3 messages by the runtime engine.
AD	An abbreviation for the HL7 data type Postal Address.
ADXP	An abbreviation for the HL7 data type Address Part.
AE	Adverse Event
ANSI	American National Standards Institute

Term	Definition
ANY	An HL7 data type that defines the basic properties of every data value. This is an abstract type, meaning that no value can be just a data value without belonging to any concrete type. Every concrete type is a specialization of this general abstract DataValue type.
API	Application Programming Interface
BL	An abbreviation for the HL7 data type Boolean. The Boolean data type stands for the values of two-valued logic. A Boolean value can be either true or false, or, as any other value be NULL.
caBIG	cancer Biomedical Informatics Grid
caBIO	Cancer Bioinformatics Infrastructure Objects
caCORE	cancer Common Ontologic Representation Environment
cardinality	Cardinality describes the minimum and maximum number of associated objects within a set
CD	An abbreviation for the HL7 data type Concept Descriptor. A concept descriptor represents any kind of concept usually by giving a code defined in a code system. A concept descriptor can contain the original text or phrase that served as the basis of the coding and one or more translations into different coding systems. A concept descriptor can also contain qualifiers to describe, e.g., the concept of a "left foot" as a post coordinated term built from the primary code "FOOT" and the qualifier "LEFT". In exceptional cases, the concept descriptor need not contain a code but only the original text describing that concept.
CDMS	Clinical Trials Data Management System
CE	An abbreviation for the HL7 data type Coded with Equivalents.
clone	An instance of a RIM class as used in a DMIM or RMIM
CMET	Common Message Element Type
CR	An abbreviation for the HL7 data type Concept Role.
CS	An abbreviation for the HL7 data type Coded Simple Value. Coded data in its simplest form, consists of a code and original text. The code system and code system version is fixed by the context in which the CS value occurs. CS is used for coded attributes that have a single HL7-defined value set.
CSV	Comma-Separated Values
CTMS	Clinical Trials Management System
CTS	HL7 Common Terminology Service
CV	An abbreviation for the HL7 data type Coded Value.
DAM	Domain Analysis Module
DMIM	Domain Message Information Model. This is a model of a specific subject matter area within the universe of healthcare information. It uses an HL7-specific variation on the UML modeling notation.
ED	An abbreviation for the HL7 data type Encapsulated Data.
EN	An abbreviation for the HL7 data type Entity Name.

Term	Definition
ENXP	An abbreviation for the HL7 data type Entity Name Part.
EVS	Enterprise Vocabulary Services
FDA	Food and Drug Administration
GUI	Graphical User Interface
GUID	Globally Unique Identifier. A GUID is a pseudo-random number used in software applications. While each generated GUID is not guaranteed to be unique, the total number of unique keys is so large that the possibility of the same number being generated twice is very small.
HL7	Health Level Seven
home directory	Indicates the directory where caAdapter is installed
ICD	International Classification of Diseases
ICSR	Individual Case Safety Report. This is a name commonly used to refer to HL7's adverse event related message, also technically known as PORR_MT040011.
II	An abbreviation for the HL7 data type Instance Identifier.
INT	An abbreviation for the HL7 data type Integer. Integer numbers are precise numbers that are results of counting and enumerating. Integer numbers are discrete; the set of integers is infinite but countable. No arbitrary limit is imposed on the range of integer numbers. Two NULL flavors are defined for the positive and negative infinity.
ISO	International Standards Organization
IVL	An abbreviation for the HL7 data type Interval.
JAR	Java Archive
Javadoc	Tool for generating API documentation in HTML format from doc comments in source code (http://java.sun.com/j2se/javadoc/)
JavaSIG	Java Special Interest Group
JSP	JavaServer Pages
LOINC	Logical Observation Identifiers, Names and Codes
metadata	Definitional data that provides information about or documentation of other data; can describe anything such as file formats, databases, documents, web pages, etc.
MIF	Model Interchange Format
MO	An abbreviation for the HL7 data type Monetary Amount. A monetary amount is a quantity expressing the amount of money in some currency. Currencies are the units in which monetary amounts are denominated in different economic regions. While the monetary amount is a single kind of quantity (money) the exchange rates between the different units are variable. This is the principle difference between physical quantity and monetary amounts, and the reason why currency units are not physical units.
MT	Message Type

Term	Definition
multiplicity	Multiplicity of an association end indicates the number of objects of the class on that end may be associated with a single object of the class on the other end
NCI	National Cancer Institute
NCICB	National Cancer Institute Center for Bioinformatics
OID	Object Identifiers. Object identifiers are unique numeric values that are granted by various issuing authorities to identify data elements, syntaxes, and other parts of distributed applications. Object identifiers are, basically, strings of numbers. They are allocated in a hierarchical manner (for example, the authority for "1.2.3" is the only one that can say what "1.2.3.4" means). They are used in a variety of protocols. The formal definition of OIDs comes from ITU-T recommendation X.208 (ASN.1).
ON	An abbreviation for the HL7 data type Organization Name.
ONXP	An abbreviation for the HL7 data type Organization Name Part.
OSI	Open Systems Interconnection
PN	An abbreviation for the HL7 data type Person Name.
PQ	An abbreviation for the HL7 data type Physical Quantity. A dimensioned quantity expressing the result of a measurement act.
PQR	An abbreviation for the HL7 data type Physical Quantity Representation.
QTY	An abbreviation for the HL7 data type Quantity. The quantity data type is an abstract generalization for all data types whose value set has an order relation (less-or-equal) and where difference is defined in all of the data type's totally ordered value subsets. The quantity type abstraction is needed in defining certain other types, such as the interval and the probability distribution.
REAL	An abbreviation for the HL7 data type Real Numbers (fractional numbers). They are typically used whenever quantities are measured, estimated, or computed from other real numbers. The typical representation is decimal, where the number of significant decimal digits is known as the precision. Real numbers are needed beyond integers whenever quantities of the real world are measured, estimated, or computed from other real numbers.
RIM	Reference Information Model. The RIM is a generic, abstract UML model that expresses the information content of all areas of the health-care universe. It is the fundamental model from which all HL7 v3 messages are derived.
RMIM	Refined Message Information Model. This is a model of a specific message topic within a specific subject matter area in the universe of health-care information. It is derived from a DMIM and uses the same HL7-specific variation on the UML modeling notation.

Term	Definition
RTO	An abbreviation for the HL7 data type Ratio. A quantity constructed as the quotient of a numerator quantity divided by a denominator quantity. Common factors in the numerator and denominator are not automatically cancelled out. The RTO data type supports titers (e.g., “1:128”) and other quantities produced by laboratories that truly represent ratios. Ratios are not simply “structured numerics”, particularly blood pressure measurements (e.g. “120/60”) are not ratios. In many cases REAL should be used instead of RTO.
SAX	Simple API for XML
SC	An abbreviation for the HL7 data type Character String. It may have a code attached.
SDK	Software Development Kit
SDOs	Standards Developing Organizations
SNOMED	Systematized Nomenclature of Medicine
ST	An abbreviation for the HL7 data type Character String. The character string data type stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) It is used for names, symbols, and formal expressions.
structural attributes	Structural attributes are used to specify type and state of each RIM class and what it means when used in a message. They use a standard vocabulary defined and controlled by HL7.
TEL	An abbreviation for the HL7 data type Telephone number.
TS	An abbreviation for the HL7 data type Point in Time. A quantity specifying a point on the axis of natural time. A point in time is most often represented as a calendar expression.
UCUM	Unified Code for Units of Measure. The UCUM is a code system intended to include all units of measure being contemporarily used in international science, engineering, and business.
UML	Unified Modeling Language. UML is a modeling language that unifies the object-oriented modeling methods of Grady Booch, Jim Rumbaugh, Ivar Jacobson, and others. The UML is a rich, mainly graphical, means of expressing object-oriented concepts.
URL	Uniform Resource Locator
UUID	An abbreviation for the HL7 data type Universally Unique Identifier. The GUID is an implementation by Microsoft of a standard called UUID, specified by the Open Software Foundation (OSF).
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

INDEX

Symbols

.dbs extension 65
.fls extension 65
.h2s extension 65
.h3s extension 65
.map extension 65
.scs extension 65
.xml extension 65

A

Abstract data types
 HL7 v3 property 44
 updating 51
Add Clone option 48
Add Function option 57
Adding
 clones to HL7 v3 Specification 48
 fields in CSV 33
 functions 70
 functions to function library 70
 function to map specification 57
 input to a function 58
 multiple attributes on HL7 v3 specification 50
 multiple clones on HL7 v3 specification 49
 segments in CSV 32
Add Multiple Clone option 50
Adverse event
 message 11
 reporting 14
Analysts, using caAdapter 1
ant compile 24
ant launchui 24
ANY label 51
API, caAdapter 18
Architecture
 caAdapter core engine 9
 mapping tool 9
Artifacts, HL7 6
Assigning an OID 47
Attribute, HL7 v3 specification 36

Audience of user's guide 1

B

Binary distribution, starting 23
build directory 17
Building
 HL7 v3 ICSR message 11
 object graph 11, 14
 XML 13
Business rules
 CSV specification 30
 HL7 v3 message 60
 HL7 v3 specification 35
 map specification 52

C

caAdapter
 APIs 14
 mapping tool architecture 9
 overview 7
 tested data 7
caadapter.log file 13, 21
caadapter_ui.jar 24
caBIG solution 7
caBIO API 20
caCORE, caAdapter integrates 7
Cardinality 42, 49
CDMS, operational scenario 14
changeFormat date function 58
Changing logging properties 21
Choice
 boxes 51
 selected 51
 unselected 51
Clinical data 7
Clone
 adding 48
 Attribute Object Properties panel 56
 HL7 v3 specification 36
Clone List dialog box 49, 51

- Close all files [27](#)
- Close file [27](#)
- CMET
 - HL7 v3 property [45](#)
 - in HL7 v3 specification [36](#)
- COCT_MT090102 message type [37](#)
- codeSystem data type [47](#)
- Coding strength, HL7 v3 property [45](#)
- Component, defined [70](#)
- Components of caAdapter [7](#)
- Conformance, HL7 v3 property [43](#)
- Constant function [58](#)
- Converting data file into HL7 v3 message [61](#)
- core.fls file [70](#)
- Core engine architecture [9](#)
- Creating
 - CSV Specification [31](#)
 - HL7 v3 message [61](#)
 - HL7 v3 Specification [37](#)
 - mapping link [55](#)
 - map specification [54](#)
- CSV data file format [66](#)
- CSV Field Properties [56](#)
- CSV specification
 - business rules [30](#)
 - file example [67](#)
 - format [66](#)
 - tab overview [30](#)
 - updating [32](#)
- CTS, HL7 [20](#)

D

- Data exchange specifications [6](#)
- Data type
 - element [36](#)
 - field [51](#)
 - HL7 v3 property [45](#)
 - specification [6](#)
- Date function, changeFormat [58](#)
- Default values
 - behavior [46](#)
 - defining [46](#)
 - structural attributes [47](#)
- Defining
 - default data [46](#)
 - inlineText [46](#)
 - mappings [24](#)
 - object identifiers [47](#)
 - units of measure [46](#)
- Delete button [33](#)
- Deleting
 - fields in CSV [33](#)
 - map lines [56](#)

- segments in CSV [33](#)
- DMIM, defined [6](#)
- docs directory [17](#)
- Document conventions [2](#)
- Dragging-and-dropping elements in CSV [33](#)

E

- Editable values, HL7 v3 specification [42](#)
- Edit Constant option [58](#)
- Editing
 - constant function [58](#)
 - field name [33](#)
 - fields in CSV [33](#)
 - segment name [32](#)
 - segments in CSV [33](#)
- Element
 - types of HL7 v3 specification [36](#)
- Element name, HL7 v3 property [42](#)
- Element parent, HL7 v3 property [42](#)
- Element type, HL7 v3 property [42](#)
- ERROR message [29](#)
- etc directory [17](#)
- EVS, validation [7](#), [8](#), [9](#), [12](#), [14](#), [20](#)
- Example
 - CSV specification file [67](#)
 - data [17](#), [73](#)
 - data structure [73](#)
 - function specification file [69](#)
 - HL7 v3 message file [71](#)
 - HL7 v3 specification [68](#)
 - map specification file [71](#)
 - OIDs [47](#)

- examples directory [17](#), [68](#), [71](#), [73](#)
- Excel spreadsheet [60](#)
- Existing CSV specification [31](#)
- Exit caAdapter [27](#)
- Extensions, file descriptions [65](#)

F

- FATAL message [29](#)
- FDA, operational scenario [14](#)
- Field properties [33](#)
- File
 - New CSV Specification [31](#)
 - New HL7 v3 Message [62](#)
 - New HL7 v3 Specification [37](#)
 - New Map Specification [54](#)
 - Open CSV Specification [32](#)
 - Open HL7 v3 Specification [38](#)
 - Open Map Specification [54](#)
 - Save [34](#), [52](#), [59](#), [64](#)
 - Save As [34](#), [52](#), [59](#), [64](#)

- Validate [34](#), [52](#), [59](#)
- File extensions [65](#)
- File options [26](#)
- File types [65](#)
- Format
 - CSV data file [66](#)
 - CSV specification [66](#)
 - function specification [69](#)
 - HL7 v3 message [71](#)
 - HL7 v3 specification [68](#)
 - map specification [70](#)
 - of files [65](#)
- Four pillars of semantic interoperability [5](#)
- Function
 - component, defined [70](#)
 - group properties panel [56](#)
 - library [70](#)
 - panel, defined [57](#)
 - properties panel [56](#), [57](#)
 - requirements [70](#)
 - specification, example file [69](#)
 - specification format [69](#)

G

- Generating
 - CSV Report [34](#)
 - CSV specification [24](#)
 - HL7 specification [24](#)
 - HL7 v3 messages [61](#)
 - Map Report [60](#)
- Goal of HL7 [5](#)
- gov.nih.nci.hl7.map package [20](#)
- gov.nih.nci.hl7.validation package [20](#)
- GUID, structural attribute [47](#)

H

- Help option [27](#)
- HL7
 - artifacts [6](#)
 - assigned OIDs [47](#)
 - choice boxes [51](#)
 - controlled vocabulary [47](#)
 - default value, HL7 v3 property [45](#)
 - domain, HL7 v3 property [45](#)
 - key goal [5](#)
 - overview [5](#)
 - RIM object graphs [19](#)
 - supported message types [37](#)
 - v3 artifacts [9](#)
- HL7 v3 message
 - business rules [60](#)
 - creating [61](#)
 - defined [60](#)
 - dialog box [62](#)

- Example file [71](#)
- format [71](#)
- overview [61](#)
- tab features [64](#)
- HL7 v3 Message Builder [19](#)
- HL7 v3 Message Parser [18](#)
- HL7 v3 Property Descriptions [42](#)
- HL7 v3 specification
 - attribute properties panel [56](#)
 - data type field properties panel [56](#)
 - dialog box [37](#)
 - element options [37](#)
 - example file [68](#)
 - format [68](#)
 - properties panel [42](#), [48](#)
 - tab overview [35](#)
 - validating [52](#)

I

- ICSR
 - caAdapter tested [7](#)
 - described in process flow [11](#)
 - operational scenario [14](#)
- images directory [17](#)
- INFO message [29](#)
- inlineText data type field [46](#)

J

- javax.xml.transform.Transform [19](#)
- JdomMessageTypeLoader [18](#)

L

- lib directory [17](#)
- license directory [17](#)
- Link
 - defined [52](#)
 - properties panel [56](#)

- Log files [21](#)
- logging.properties file [21](#)

M

- Mandatory
 - HL7 v3 property [42](#)
 - values [46](#)
- MapGenerateResult class [20](#)
- Mapping
 - allowed symbol [55](#)
 - functions [10](#)
 - line [56](#)
 - not allowed symbol [55](#)
- Mapping tool
 - architecture [9](#)

- basic steps [24](#)
- defined [7](#)
- interface [25](#)
- operational scenario [14](#)
- Map specification
 - business rules [52](#)
 - creating [54](#)
 - example file [71](#)
 - format [70](#)
 - internal reference [66](#)
 - opening [54](#)
 - status [60](#)
 - tab overview [53](#)
 - updating [55](#)
 - validating [59](#)
- Menu bar [25, 26](#)
- Message builder [9, 19](#)
- MessageContentHandler [18](#)
- Message Level [28](#)
- Message parser [9, 18](#)
- Messages, types of errors [29](#)
- Message Service Integration, defined [9](#)
- messageType [19](#)
- MessageTypeLoader [18](#)
- Message types, supported [37](#)
- Meta Data Loader [9, 18](#)
- MIF
 - defined [6](#)
 - file [18](#)
 - format [18](#)
 - HL7 file [14](#)
 - HL7 v3 properties reflected [42](#)
 - schema derived from [11](#)
- Move Down button [32](#)
- Move Up button [32](#)
- Moving a segment in CSV [33](#)
- MT, defined [6](#)
- Multiples in HL7 v3 specification [49](#)

N

- NCICB [7](#)
 - clinical trials architecture vision [7](#)
 - training resources [6](#)
- New CSV specification dialog [31](#)
- New file [26](#)
- Next button [64](#)

O

- OID
 - defining [47](#)
 - registry page [47](#)
- Open CSV specification dialog [32](#)

- Open Data File dialog box [62](#)
- Open HL7 v3 Specification File dialog box [38](#)
- Opening

- CSV specification [31](#)
 - file [27](#)
 - HL7 v3 specification [37](#)
 - map specification [54](#)
 - new file [28](#)

- Open Map Specification dialog box [62](#)

- Open Source File dialog box [54](#)

- Open Target File dialog box [54](#)

- Operational scenario

- caAdapter API [14](#)
 - mapping tool [14](#)

- Optional associations [48](#)

- org.hl7.meta package [18](#)

- org.hl7.xml.builder package [19](#)

- org.hl7.xml.parser package [18](#)

- org.xml.sax.ContentHandler [19](#)

- Overview, chapters [2](#)

P

- Parser, HL7 v3 message [18](#)

- Parsing

- message [11, 14](#)
 - XML [11](#)

- XML to object graph [12](#)

- PORR_MT040002 message type [37](#)

- PORR_MT040011 [37](#)

- Previous button [64](#)

- Printing, validation messages [28](#)

- Process

- flow, caAdapter API [11](#)
 - to parse an AE [11](#)

- Properties

- HL7 v3 specification [42](#)
 - panel [56](#)

Q

- QTY label [51](#)

R

- Reading materials [1](#)

- Receiving an HL7 v3 message [11](#)

- Regenerate button [64](#)

- Registering custom function libraries [70](#)

- Remove Clone option [49](#)

- Remove Multiple Attribute option [50](#)

- Remove Multiple Clone option [50](#)

- Removing, multiple attributes from HL7 v3 specification [50](#)

Removing, multiple clones from HL7 v3 specification 50

Report

CSV example 34
generate map report 60
generate report 34
Map specification 60
option 27

Reset button 33

Resizing panels 26

Resources 1

RIM

classes 47
defined 6
file 18
object graph 18
object graphs 19
source, HL7 v3 property 43
used in four pillars 5

RimGraphXMLSpeaker 19

RMIM

defined 6

S

Save as file 27

Save file 27

Saving

CSV Specification 34
HL7 v3 Message 64
HL7 v3 Specification 52
map specification 59
validation messages 28

SAX events 18, 19

SAX parser 18

Scenarios, mapping 73

schema directory 17

Scroll bars 26

Segment

options 32
properties 32

Select Choice option 51

Selected Choice for label 51

Semantic interoperability 5

SimpleDateFormat class 58

Source distribution, starting 24

Source specification 30

Source specification, defined 10

src directory 17

Starting, mapping tool 23

Structural attributes, using 47

T

Tab

CSV specification 30
HL7 v3 message 61
HL7 v3 specification 35
map specification 53
open 25
types of 28

Target specification

types 35

Target specification, defined 10

Technical audience 1

Tool bar 25, 27

Training, online tutorials 6

Transformation Service 19

Transformation Service, defined 10

TransformationService class 20

Transforming, into RIM object graph 24

U

UCUM, units of measure 46

Units of measure properties 46

Updating

abstract data types 51
CSV specification 32
HL7 v3 specification 42
map specification 55

User-defined default value 46, 47, 48

User interface, defined 10

Using

date function 58

Using functions in map specifications 57

UUID, used in files 66

V

Validating

CSV 33
CSV data against specialization 34
CSV specification 34
defined 10
file option 27
HL7 structural attributes 20
HL7 v3 specification 52
map specification 59
object graph against EVS 12
purpose 28
structural attributes 12
structure 11
vocabulary using EVS 14

Validation Messages dialog box 59

Validation Messages panel 28, 52

Validation services, defined 9

Valid data types 51

Vocabulary domain [6](#)
Vocabulary validation [20](#)

W

WARNING message [29](#)
Windows distribution, starting [24](#)
Windows layout, mapping tool [25](#)
workspace directory [17](#)

X

XMLRimGraphSpeaker [19](#)
XMLSpeaker [19](#)