caADAPTER 4.3

User's Guide

NATIONAL® CANCER INSTITUTE

Center for Biomedical Informatics
and Information Technology

April 11, 2009

TABLE OF CONTENTS

About This Guide 1

Purpose 1
Audience 1
Typical User 1
Prerequisites 1
Organization of This Guide 2
Recommended Reading 3
Text Conventions Used 3
Credits and Resources 4

Chapter 1
Overview of caAdapter 5

About caAdapter 5
caAdapter Core Engine Architecture 6
caAdapter Architecture 7
About HL7 8
Prerequisites for Using caAdapter 10
Resources for Installing caAdapter 10
Starting caAdapter 10
Starting caAdapter from the Binary Distribution 10
Starting caAdapter from the Source Distribution 10
Starting caAdapter from the Windows Distribution 10
Starting the Mapping Tool on the Web (WebStart) 10
caAdapter Common Features 11
caAdapter Interface 11

Chapter 2
Using caAdapter 15

API Process Flow for CSV to HL7 V3 Transformation 15
API Operational Scenario for CSV to HL7 V3 16
Operational Scenario for CSV to HL7 V3 17
Operational Scenario for HL7 V2 to HL7 V3 Transformation 17

Chapter 3
CSV to HL7 V3 Mapping and Transformation 19

caAdapter Process Flow 20
caAdapter Validation 20
Source Specification 22
CSV Specification 22
Creating a New CSV Specification File 23
Opening an Existing CSV Specification File 24
Updating the CSV Specification 25
Validating the CSV Specification 27
Saving a CSV Specification 27
Generating a Report 28
Target Specification 28
HL7 V3 Specification 28
Overview of HL7 V3 Specification Tab 29
Defining Inline Text 31
Defining Units of Measure 31
Defining Default Data 32
Defining Object Identifiers (OIDs) 32
Adding Optional Clones to the HL7 V3 Specification 33
Adding and Removing Multiples in the HL7 V3 Specification 34
Updating Abstract Data Types in the HL7 V3 Specification 35
Enabling and Disabling Force XML with an Optional Clone 35
Map Specification 39
Map Specification Business Rules 39
Creating Mappings 39
HL7 V3 Message 48
HL7 V3 Message Business Rules 48
Generating HL7 V3 Messages 48
Transforming an HL7 Message into CSV Format 52
HL7 V3 to CSV Transformation Business Rules 52
Transforming HL7 V3 Messages to CSV Format 52

Chapter 4
HLV V2 to HL7 V3 Conversion 55

Converting HL7 V2 to HL7 V3 Messages 55
Understanding the Mapping and Transformation Processes 55
Step 1: Generating the H3S File 56
Step 2: Creating the `.map` file 56
Step 3: Transforming a Message from V2 to V3 60

Using APIs for HL7 V2 to HL7 V3 Transformation 61

Chapter 5
Using Functions in Mapping 63

Functions Provided by caAdapter 63
Function Specifications 65
Function Specification Overview 65
Vocabulary Mapping Specification Overview 73
Adding Functions to the Function Library 75

Chapter 6
Using the caAdapter APIs 77

caAdapter Directory Structure 77
caAdapter API Modules 78
Metadata Loader 78
Transformation Service 79
HL7 V2 to HL7 V3 Transformation 79
Vocabulary and MIF Schema Validation 80
caAdapter API Error Logs 81

Chapter 7
caAdapter Web Services Transformation Module 83

Introduction 83
Setup Mapping Scenarios Through the Web Portal 84
Programmatic Access to the caAdapter Web Services 85
Axis 1.x RPC Style Access to caAdapter Web Services 85
Axis 1.x DII Style Access to caAdapter Web Services 86
Axis 2.0 RPC Style Access to caAdapter Web Services 87

Chapter 8
caAdapter File Types 89

caAdapter File Formats and Locations 89
CSV Data File 90
CSV Specification 91
HL7 V3 Specifications 92
HL7 V2 Specifications 97
Message Structure 98
DataTypeSpec 98
Segment Attribute Table 99
Definition Table 99
SDTM Data Files 99
SDTM Metadata Files 101

HL7 V3 Message 102
CSV to HL7 V3 Map Specification 103
Object to Database Map Specification 104

Appendix A
caAdapter Example Data 109

Appendix B
References 111

Articles 111
caBIG Material 111
caCORE Material 111
HL7 Concepts and Material 112
Software Products 112

caAdapter Glossary 113

About This Guide

This section introduces you to the *caAdapter 4.3 User's Guide*.

Topics in this section:

- *Purpose* on this page
- *Audience* on this page
- *Organization of This Guide* on page 2
- *Recommended Reading* on page 3
- *Text Conventions Used* on page 3
- *Credits and Resources* on page 4

Purpose

This guide is the companion documentation to caAdapter 4.3. It includes information and instructions for using the two main caAdapter components: a set of Application Programming Interfaces (APIs) and a mapping tool graphical user interface (GUI).

Audience

Typical User

This guide is designed for the following types of users:

- Technical users (such as Java programmers and system architects) who want to use the major caAdapter APIs to parse, build, and validate Health Level Seven version 3 (HL7 V3) messages; and
- Analysts (such as HL7 analysts, database administrators, and business analysts) who need step-by-step procedures for creating HL7 V3 XML message instances using caAdapter.

Prerequisites

This guide assumes that you are familiar with the HL7 Reference Information Model (RIM).

Use of caAdapter requires additional prerequisites. For more information, see *Prerequisites for Using caAdapter* on page 10.

Organization of This Guide

The *caAdapter 4.3 User's Guide* includes the following chapters:

- *Chapter 1, Overview of caAdapter,* on page 5 discusses the caAdapter architecture and related data standards.
- *Chapter 2, Using caAdapter,* on page 15 provides a high-level overview of using caAdapter.
- *Chapter 3, CSV to HL7 V3 Mapping and Transformation,* on page 19 explains the procedures for using caAdapter to perform for CSV to HL7 V3 mapping and transformation.
- *Chapter 4, HLV V2 to HL7 V3 Conversion,* on page 55 provides detailed instructions for using the caAdapter GUI for HL7 V2 to HL7 V3 conversion.
- *Chapter 5, Using Functions in Mapping,* on page 63 provides detailed instructions for using and adding functions in caAdapter mappings.
- *Chapter 6, Using the caAdapter APIs,* on page 77 provides Java developers information required to use caAdapter APIs.
- *Chapter 7, caAdapter Web Services Transformation Module,* on page 83 provides detailed instructions for using the caAdapter Web Service.
- *Chapter 8, caAdapter File Types,* on page 89 provides an overview of the different types of files used by caAdapter and an example of each.
- *Appendix A, caAdapter Example Data*, on page 109 provides a description of the example data delivered with caAdapter.
- *Appendix B, References*, on page 111 provides a list of references used to produce this guide or referred to within the text.

Recommended Reading

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

Resource	*URL*
Health Level 7 (HL7)	http://www.hl7.org
National Cancer Institute Center for Bioinformatics (NCICB) HL7 tutorial	http://ncicb.nci.nih.gov/infrastructure/cacore_overview/caadapter/indexContent/HL7_Tutorial
Unified Modeling Language (UML)	http://www.cdisc.org/models/sds/v3.1/

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

Text Conventions Used

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

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

Credits and Resources

The following people contributed to the development of this document.

caAdapter Development and Management Teams		
Development	***Documentation***	***Program Management***
Eugene Wang [2]	Carolyn Kelley Klinger[3]	Sichen Liu [1]
Ki Sung Um [2]		Anand Basu [1]
Ye Wu [2]	***Quality Assurance***	Christo Andonyadis [1]
	Jyothsna Chilukuri[2]	Sharon Gaheen [2]
[1] National Cancer Institute Center for Bioinformatics (NCICB)		[2] Science Application International Corporation (SAIC)
[3] Lockheed Martin Management System Designers		

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

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

CHAPTER

1

OVERVIEW OF CAADAPTER

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

Topics in this chapter include:

- *About caAdapter* on page 5
- *About HL7* on page 8
- *Prerequisites for Using caAdapter* on page 10
- *Prerequisites for Using caAdapter* on page 10
- *Resources for Installing caAdapter* on page 10
- *Starting caAdapter* on page 10
- *caAdapter Common Features* on page 11

About caAdapter

The caAdapter (http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter) consists of several components that, via messaging standards, support data sharing at NCICB (http://ncicb.nci.nih.gov) and/or cancer centers as part of the cancer Biomedical Informatics Grid (caBIG) (http://caBIG.nci.nih.gov) solution. The components include a core engine for building, parsing, and validating HL7 V3 messages via an API or web service, and a mapping tool for providing mapping and transformation services using an assortment of messaging standards or formats such as HL7 V2 and V3 and object and data models.

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 structural,

vocabulary and schema validation. 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) and the caCORE Software Development Kit Programmer's Guide (ftp://ftp1.nci.nih.gov/pub/cacore/SDK) 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.

caAdapter is an open source application that supports several types of mapping and transformation. It enables analysts and database engineers, who are knowledgeable about HL7, to create a mapping from 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.

Using similar GUI and mapping features, caAdapter also enables HL7 V2 analysts to convert V2 messages to HL7 V3 mapping capabilities.

In addition, caAdapter supports object to data model mapping. This component allows you to parse and load data and object models from an `XMI` file, map the object model to the data model using drag-and-drop capabilities, add SDK-required tags and tag values into the `XMI` file, and generate a Hibernate mapping file.

caAdapter Core Engine Architecture

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

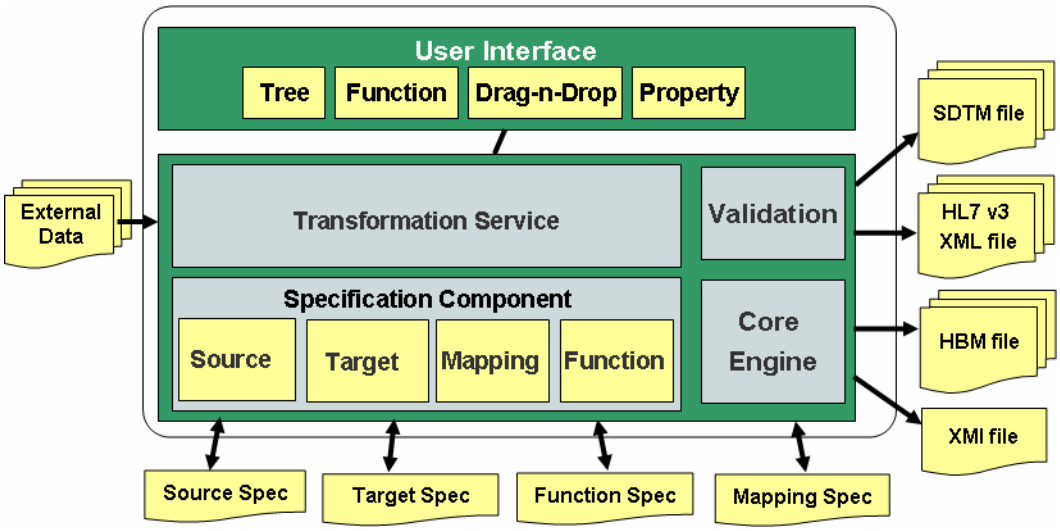


Figure 1.1 caAdapter Core Engine Architecture

The main features of the caAdapter core engine are:

- Metadata Loader – represents HL7 V3 metadata in-memory
- Parser – parses source data and validates against its specification
- Message Builder – builds messages from source data to the target object
- Validation Service – validates the structure and content of mapping files

caAdapter Architecture

caAdapter is a graphical application for mapping clinical data to an HL7 V3 message. *Figure 1.2* illustrates caAdapter architecture design depicting its subsystems and components.

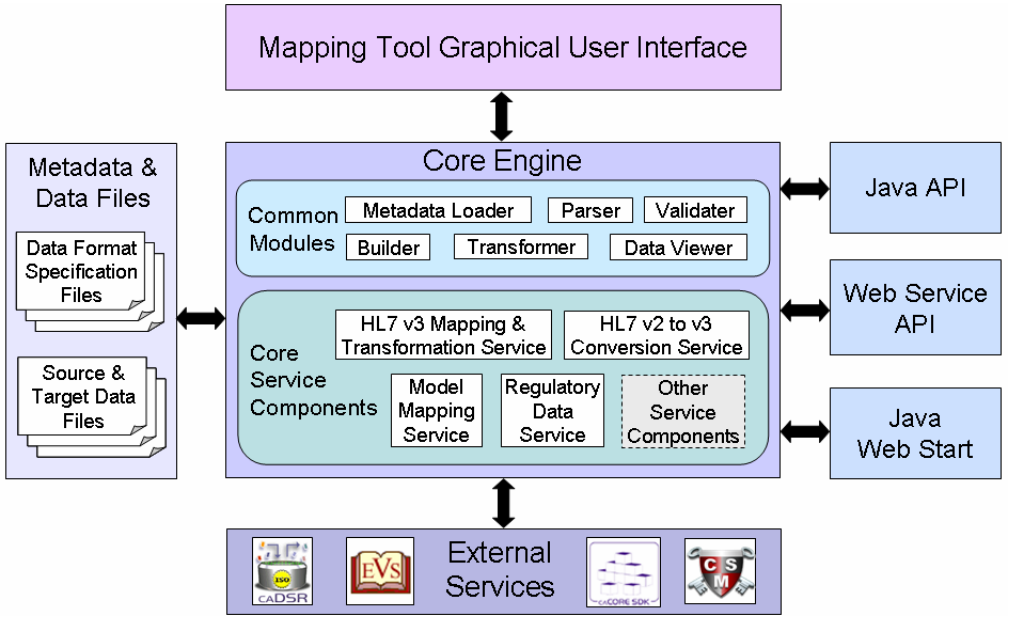


Figure 1.2 caAdapter Architecture

The mapping component of caAdapter has the following features:

- Source and target specification - graphical interface for defining input and output data formats
- User interface - simple mechanism for mapping source fields to target elements containing tree structure, drag-and-drop functionality, and functions and property definitions
- Mapping functions - capability to do simple-source data manipulation
- Validation - capability to validate the structure and content of mapping files

About HL7

Health Level Seven (HL7) (http://www.hl7.org/) is one of several American National Standards Institute (ANSI)-accredited Standards Developing Organizations (SDOs) operating in the 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.

HL7 version 2 (V2) is a messaging standard that focuses on syntactic data interchange. HL7 messaging (V2 or higher) has been recommended as a data exchange standard by the e-Government initiative. In fact, various releases of this version are in use in over 90% of U.S. hospitals, and V2 is considered the most widely implemented standard for healthcare information in the world. However, since it lacks an explicit methodology, conformance rules, and grouping of messages, it cannot be considered an interoperability standard.

HL7 V2 messages are composed of segments (individual lines in a message) which are composed of fields (data values) which may in turn be composed of components and sub-components. Several different delimiters or field separators are used to mark boundaries between the various elements. Specifications for messages using these structures are published in a text document format which does not easily lend itself to being computable. Furthermore, messages are often customized at local sites making it difficult to share messages between sites. caAdapter consequently includes a computable version of the message specifications which can be tailored to suit the needs of cancer centers and hospitals.

HL7 as an organization aimed to address some of the problems of V2 in its next major version, version 3 (V3). The key goal of the HL7 community is syntactic and semantic interoperability. This goal is supported in HL7 V3 by what are commonly called the four pillars of semantic interoperability:

1. A common Reference Information Model (RIM) spanning the entire clinical, administrative, and financial healthcare universe. The RIM is the cornerstone of the HL7 V3 development process. An object model created as part of the V3 methodology, the RIM is a large pictorial representation of the clinical data domains and identifies the life cycle of events that a message or groups of related messages will carry. It is a shared model between all the domains and is the model from which all domains create their messages. Explicitly representing the connections that exist between the information carried in the fields of HL7 messages, the RIM is essential to HL7's ongoing mission of increasing precision and reducing implementation costs.
2. A well-defined and tool-supported process for deriving data exchange specifications from the RIM. HL7 has defined a methodology and process for developing specifications, artifacts to document the models and specifications, tools to generate the artifacts and an organization for governing the overall process of standards development. Such structure avoids ambiguity common to many existing standards.

3. A formal and robust data type specification upon which to ground the RIM. Data types are the basic building blocks of attributes. They define the structural format of the data carried in the attribute and influence the set of allowable values an attribute may assume. HL7 defines an extensive set of complex data types which provide the structure and semantics needed to describe data in the healthcare arena.
4. A formal methodology for binding concept-based terminologies to RIM attributes. Within HL7, a vocabulary domain is the set of all concepts that can be taken as valid values in an instance of a coded field or attribute. HL7 has defined vocabulary domains for some attributes to support use of the RIM in messages. It also provides the ability to use, document, and translate externally coded vocabularies in HL7 messages.

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

- The RIM is the foundational Unified Modeling Language (UML) class diagram representing the universe of all healthcare data that may be exchanged between systems.
- A Domain Message Information Model (DMIM) is a subset of the RIM that includes RIM class clones, attributes, and associations that can be used to create messages for a particular domain (a particular area of interest in healthcare). DMIMs use HL7 modeling notation, terminology, and conventions.
- A Refined Message Information Model (RMIM) is a subset of a DMIM that is used to express the information content for an individual message or set of messages with annotations and refinements that are message specific.
- A Model Interchange Format (MIF) is an XML representation of the information contained in an HL7 specification, and is the format that all HL7 V3 specification authoring and manipulation tools will be expected to use.
- A Message Type (MT) is the specification of an individual message in a specific implementation technology.

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

CBIIT provides training resources to assist the caBIG community and other interested parties in implementing HL7 V3 messaging. These resources include online tutorials, self-paced training, and links to HL7 resources (http://ncicb.nci.nih.gov/infrastructure/cacore_overview/caadapter/indexContent/HL7_Tutorial).

Prerequisites for Using caAdapter

The following skills will ensure successful use of caAdapter:

- Thorough familiarity with source data
- HL7 artifacts, messages, and data types for HL7 V2 and HL7 V3
- Training on caAdapter
- Familiarity with caAdapter mapping rules

Resources for Installing caAdapter

Complete instructions for installing caAdapter are located in the *caAdapter Installation Guide* at http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caadapter/.

Starting caAdapter

Starting caAdapter from the Binary Distribution

To start the caAdapter binary distribution, follow these steps:

1. In a Command Prompt window, enter `cd {home directory}` to go to your home directory (Windows example: `C:\caadapter`).
2. Enter `run.bat`. The caAdapter application appears.

Starting caAdapter from the Source Distribution

To start the caAdapter source distribution, follow these steps:

1. In a command prompt window, enter `cd {home directory}` to go to your caAdapter home directory (Windows example: `C:\caadapter`).
2. Enter `ant`.
3. Enter `cd dist`.
4. Enter `run.bat`. The caAdapter application appears.

Starting caAdapter from the Windows Distribution

To start the caAdapter from Microsoft Windows:

- Select **Start** > **caAdapter**. The caAdapter application appears.

Starting the Mapping Tool on the Web (WebStart)

You can also use caAdapter on the web without having to install the software by clicking the link in the *Use caAdapter Online* section on the following page: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caadapter.

Once you click the link, caAdapter will be downloaded to and launched on your computer. It will run locally and your data will not be uploaded to the NCICB server.

caAdapter Common Features

caAdapter Interface

The caAdapter Mapping Tool interface is Windows-based and includes a main menu bar, a tool bar, and tabs located in the top of the window. You can resize the various panels by selecting the edge of the panel and dragging. Scroll bars appear when needed to display all of the information.

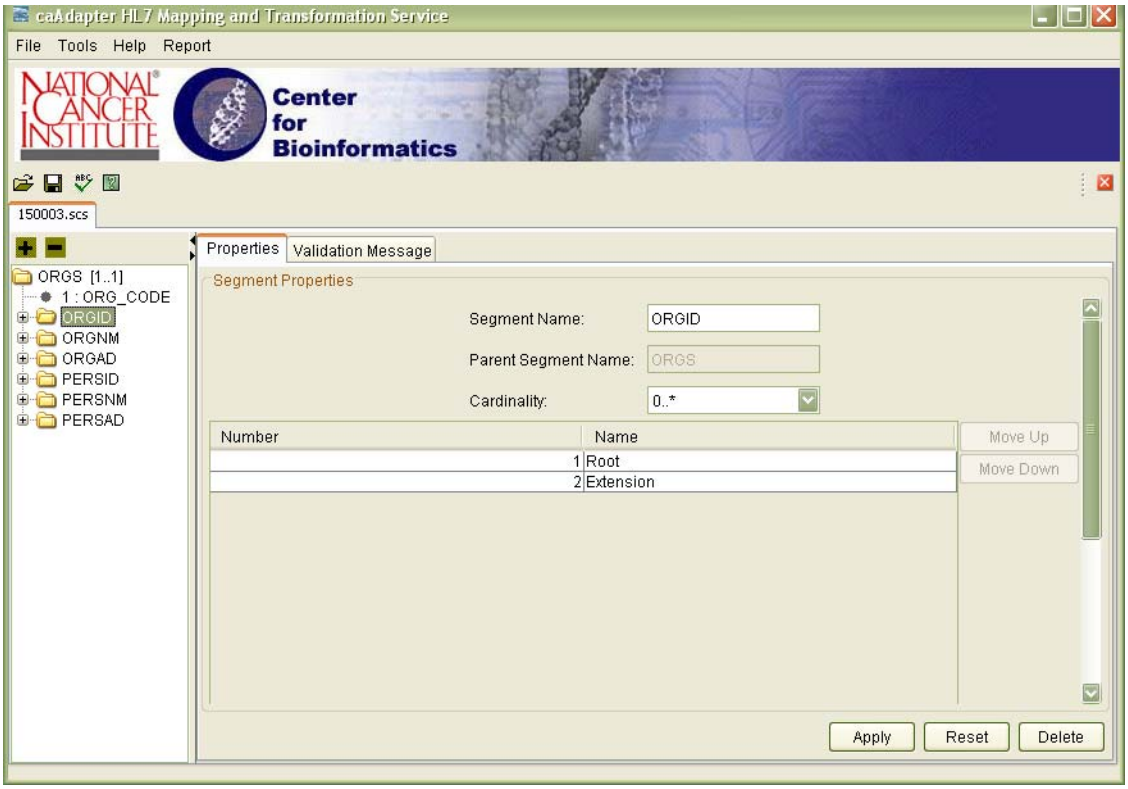


Figure 1.3 Mapping Tool Interface

Menu Bar

The menu bar is context-sensitive. The options that are available for your current window appear in a black font; unavailable options appear in a faded gray font.

The menu bar includes the File, Tools, Help, and Report commands. The following sections discuss each command.

File Menu

Select the File menu to perform the functions in *Table 1.1*.

File Option	*Description*

New	Creates a new file for the type of file you select including:
	• CSV to HL7 V3 Mapping and Transformation Service
	  ○ HL7 V3 Specification
	  ○ CSV Specification
	  ○ CSV to HL7 V3 Map Specification
	  ○ CSV to HL7 V3 Message
	• HL7 V2 to HL7 V3 Mapping and Transformation Service
	  ○ HL7 V3 Specification
	  ○ HLV V2 to HLV V3 Map Specification
	  ○ HLV V2 to HL7 V3 Message
	• HLV V3 to CSV Transformation Service
	  ○ HLV V3 to CSV

Open	Opens an existing file for the type of file you select including: • CSV Specification • HL7 V3 Specification • CSV to HL7 V3 Map Specification • HL7 V2 to HL7 V3 Map 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 message 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 open files.
Exit (ALT + F4)	Closes caAdapter.

Table 1.1 File menu commands

Tools Menu

Select the Tools menu to perform the options in *Table 1.2*.

File Option	*Description*
Preferences (CTRL + Q)	You can select options relevant to HL7 specification (Enable NullFlavor, Enable Complex Datatype, and Enable OID) and message validation (Structure, Structure & Vocabulary, and Structure, Vocabulary, and Schema (xsd)).

Table 1.2 File menu commands

File Option	Description
Load HL7 V3 Normative Edition Artifacts	Select this option to load the HL7 specification files and folders.

Table 1.2 File menu commands (Continued)

Help Menu

Select the Help menu learn more about caAdapter. The following table explains the menu options.

File Option	Description
About caAdapter	View a description of caAdapter and its license and copyright information.
Help - Contents and Index	Open a browser window that contains online help for caAdapter. Tabs appear on the left of the browser window that allow you to search for help topics in a traditional table of contents as well as an index.

Table 1.3 Help menu commands

Report Menu

The Report menu is available for files for which it is possible to generate a report. When it is not possible to create a report for a file, this menu is unavailable. The following table explains the menu option.

File Option	Description
Generate Report	Select this option to create a report of the file currently opened in caAdapter. Reports are saved in Microsoft Excel format.

Table 1.4 Report menu command

caAdapter Toolbar

The caAdapter toolbar is context-sensitive. The toolbar displays only the options that are available for your current window. *Table 1.5* describes each of the available toolbar buttons. These serve as shortcuts for specific menu commands.

Button	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.
	Refreshes the mapping panel. It is only visible if it is on the mapping panel.
	Opens the Help window.

Table 1.5 Toolbar buttons

Tabs

caAdapter 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:

- CSV specification (`.scs`)
- HL7 V3 Specification (`.h3s`, `.xml`)
- CSV to HL7 V3 Map Specification (`.map`)
- HL7 V2 to HL7 V3 Map Specification (`.map`)

In some cases, such as with `.map`, only one of each file type may be open at a time. If you open a new file of a file type that is restricted and already open, then the existing file will be replaced with the new file. The tab name is either the name of the file in the tab or `untitled.<ext>`, where `<ext>` is the appropriate file extension for that type of tab. 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.

CHAPTER

2

USING caADAPTER

Topics in this chapter include:

- *API Process Flow for CSV to HL7 V3 Transformation* on page 15
- *API Operational Scenario for CSV to HL7 V3* on page 16
- *Operational Scenario for CSV to HL7 V3* on page 17
- *Operational Scenario for HL7 V2 to HL7 V3 Transformation* on page 17

API Process Flow for CSV to HL7 V3 Transformation

This section describes the process for creating a validated HL7 V3 adverse event (AE) message, also known in HL7 as an ICSR, based on a given CSV file or set of files and a corresponding mapping specification. caAdapter uses the Transformation and Validation engine to perform different validation levels based on a user's selection: structural only, structural and vocabulary; or structural, vocabulary, and schema.

The basic steps to accomplish this workflow using caAdapter are:

1. caAdapter receives a CSV file, its meta file, an HL7 V3 message specification, and the mapping file that the user used to map the CSV schema to the HL7 V3 message.
2. The transformation process uses the files above to create a preliminary HL7 V3 message, an internal instance, which will be put through the validation process.
3. The validation process uses the internal instance of the message to perform the following validation sub-processes, (1) validation against the MIF specifications. (2) validation against HL7 V3 published vocabulary, and, (3) validation against the schema of that HL7 V3 message type.
4. caAdapter creates the final HL7 V3 message that corresponds to the source CSV file.

Figure 2.1 illustrates the transformation and validation processes.

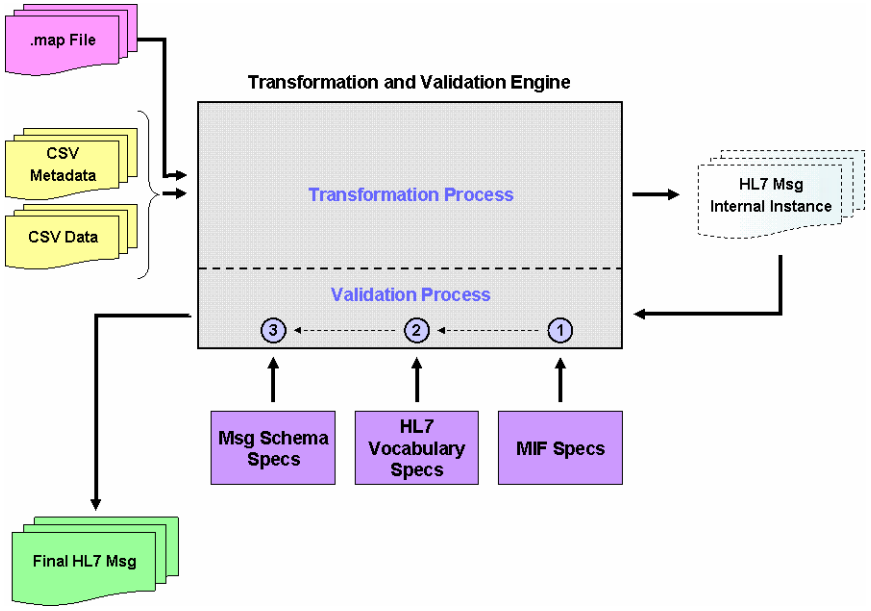


Figure 2.1 Transformation and Validation Processes

API Operational Scenario for CSV to HL7 V3

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 Enterprise Vocabulary Services (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.

Operational Scenario for CSV to HL7 V3

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 caAdapter to implement this plan.

The clinical systems analyst uses caAdapter 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 caAdapter 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 caAdapter. After the mapping is complete, the analyst then uses caAdapter 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.

Operational Scenario for HL7 V2 to HL7 V3 Transformation

A number of research institutes have been submitting daily electronic AE data to a clinical trials coordinating center which in turn consolidates and submits to the FDA. The data is being submitted in HL7 V2.5 format. The coordination center anticipates a new FDA requirement which mandates that all AE submissions be in HL7 V3 format. The coordination center decides that the best way to meet this requirement is to add an HL7 V2.5 to V3 data conversion step to its current FDA submission process.

Instead of manually implementing the conversion process, the coordinating center decided to use the caAdapter Mapping and Transformation tools to expedite the implementation.

The clinical systems analyst researched the HL7 V3 standards and identified the best message type to use for submitting AE data. The next step is to map the data elements from the HL7 V3 V2.5 message currently being used, to the identified target HL7 V3 message.

The first task of the conversion and transformation process is to use caAdapter to create a CSV file and CSV file specifications that match the HL7 V2.5 source message. The second step is to use the CSV to HL7 V3 mapping and transformation capability to map the CSV data elements to the target HL7 V3 message. For more information, see

the previous sections in this chapter. Once the map file has been created, caAdapter will use that to transform the data and create the HL7 V3 file.

Chapter 3

CSV to HL7 V3 Mapping and Transformation

This chapter explains the procedures for using caAdapter to perform CSV to HL7 V3 mapping and transformation.

Topics in this chapter include:

- *caAdapter Process Flow* on page 20
- *Source Specification* on page 22
- *Target Specification* on page 28
- *Map Specification* on page 39
- *HL7 V3 Message* on page 48
- *Transforming an HL7 Message into CSV Format* on page 52

caAdapter Process Flow

Follow the steps below to map and transform data in caAdapter. These steps are also depicted in the graphic below.

1. Generate a CSV specification file. For more information, see *Source Specification* on page 22.
2. Generate a target specification file from an HL7 MIF file. For more information, see *Target Specification* on page 28.
3. Map the source specification to the target specification, then generate a mapping file. For more information, see *Map Specification* on page 39.
4. Generate a HL7 V3 message from source data using the mapping file. For more information, see *HL7 V3 Message* on page 48.

.

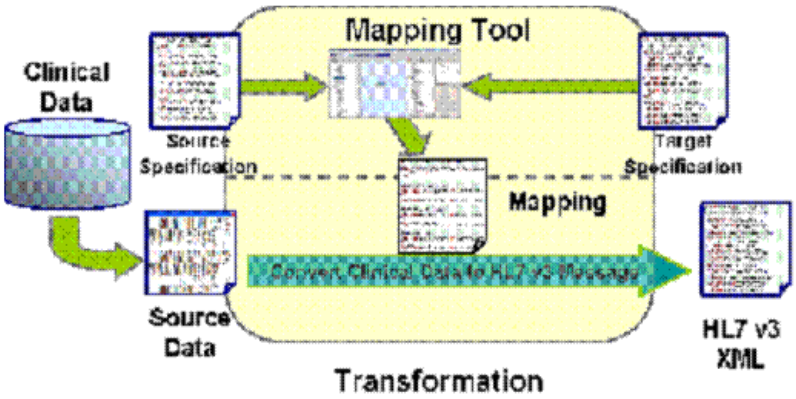


Figure 3.1 Mapping Tool Process Overview

caAdapter Validation

Validation is used to

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

The results of the validation appear in the Validation Messages panel (*Figure 3.2*). The panel displays only one level of message at a time.

From this panel you can do the following:

- 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 (*Figure 3.2*).

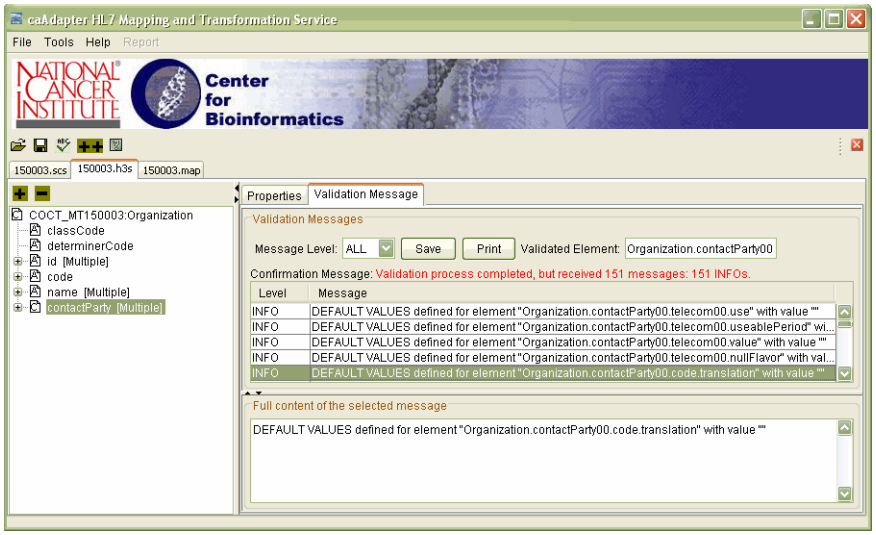


Figure 3.2 Validation Messages Panel

Table 3.1 lists and describes the different levels of messages produced during validation. It also gives examples.

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.	Not all segments and fields within the CSV specification have been mapped to the HL7 V3 specification.
INFO	Contains information, such as tips, suggestions, reminders, etc. You can simply ignore them if you want to.	Contains the choice selected for an element.

Table 3.1 Validation Messages

Source Specification

This section includes the following topics:

- *CSV Specification* on page 22
- *Creating a New CSV Specification File* on page 23
- *Opening an Existing CSV Specification File* on page 24
- *Updating the CSV Specification* on page 25
- *Validating the CSV Specification* on page 27
- *Saving a CSV Specification* on page 27
- *Generating a Report* on page 28

CSV Specification

Business Rules

Following are the business rules for a CSV specification. These rules are enforced during CSV validation:

- 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 CAPITAL letters, numbers, or the underscore character.
- Field names must be a combination of any letters (A-Z or a-z), numbers, or the underscore character.

About the CSV Specification Tab

The CSV Specification tab (*Figure 3.3*) enables you to identify the hierarchy of segments and fields that describe an incoming CSV data file that must be converted into one or more HL7 V3 XML messages. The tree structure appears in the left-hand panel, and the validation results and properties appear 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. Using the tree structure, you can drag and drop an element to another location in the tree. You can also expand and collapse a

branch of the tree using the plus (+) and minus (-) symbols. The Properties section in the right panel allows you to work with the metadata on the left.

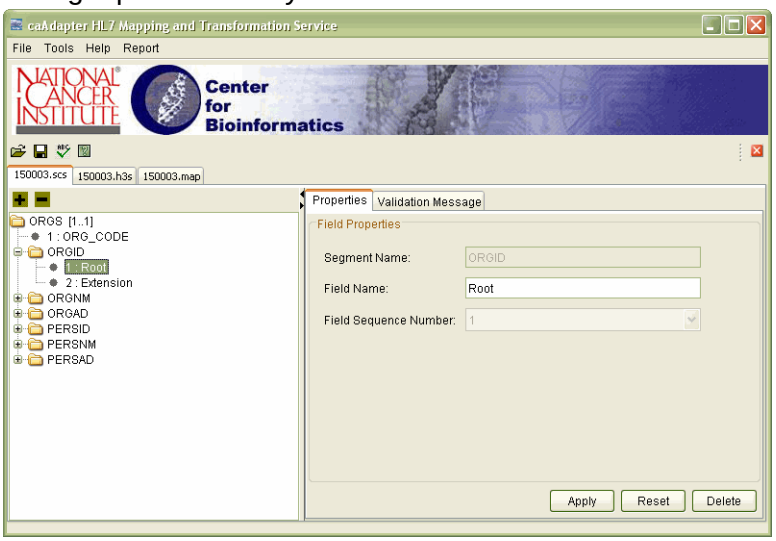


Figure 3.3 CSV Specification Tab

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

Creating a New CSV Specification File

To create a new CSV specification file, follow these steps:

1. Select the following menu commands:

   File > **New** > **CSV to HL7 V3 Mapping and Transformation Service** > **CSV Specification**

   The New CSV Specification dialog box appears.

2. Select one of the following sources, then follow the appropriate steps:

Source	***Steps***
Blank CSV Schema	Click **OK** to open the CSV Specification file named `Untitled_1.scs`. The file opens in a new tab with an empty tree, except for an initial root segment named ROOT (by default).
Generate from a CSV Instance	a. Click **Browse** to display the Open CSV File dialog box. b. Select the appropriate `.CSV` data file, then click **Open**. The New CSV Specification dialog box opens and displays the file. c. Click **OK**. A new tab named `Untitled_1.scs` opens and displays the contents of the selected file.
New from an Existing CSV Specification File	1. Click **Browse** to display the Open CSV Specification dialog box. 2. Select the appropriate `.scs` file, then click **Open**. The New CSV Specification dialog box opens and displays the file. 3. Click **OK**. A new tab named `Untitled_1.scs` opens and displays the contents of the selected file.

Table 3.2 Selecting a Source

Opening an Existing CSV Specification File

To open an existing CSV Specification file, follow these steps:

1. Select the following menu commands:

   File > **Open** > **CSV Specification**

   The Open CSV Specification dialog box appears.

2. Select the appropriate `.scs` file, then click **Open**.

   A new tab opens with the CSV Specification file displayed in the tree.

Updating the CSV Specification

Once you have a CSV specification file open, you can perform the following basic functions to update the tree hierarchy.

1. Update any default field names to have meaningful names.
   a. Click a segment in the tree structure to display the details of that element in the **Segment Properties** section (*Figure 3.4*).

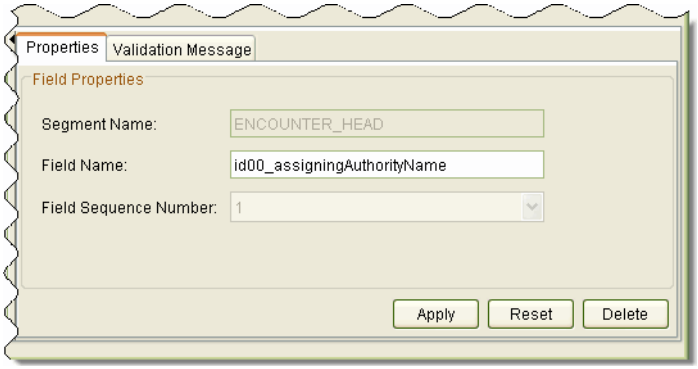


Figure 3.4 Segment Properties

   b. 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 3.4* 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.
   c. Edit the **Segment Name** and click **Apply** to update the tree in the left-hand panel.
   d. Right-click a segment to get the options available to perform on that segment (*Figure 3.5*).

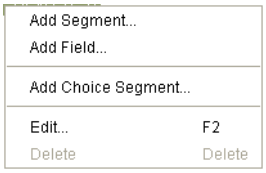


Figure 3.5 CSV segment right-click options

   — 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.
   — 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.
   — Right-click and select **Add Choice Segment** to display the Add Choice Segment dialog box. Name the new choice segment and select its cardinality type. Click **OK**. The segment is added to the tree structure.

— 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.

— 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.

2. Click a field in the tree structure to display the details of that element in the **Field Properties** section (*Figure 3.6*).

Figure 3.6 Field Name metadata properties

3. Edit the **Field Name** and click **Apply** to update the tree in the left-hand panel.
4. Right-click a field to get the options (**Edit**, **Delete**) available to perform on that field.
5. 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.
6. 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.
7. 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.
8. Use the **Reset** button to reset changes made before selecting **Apply**.
9. Use the **Delete** button 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 (*Figure 3.7*).

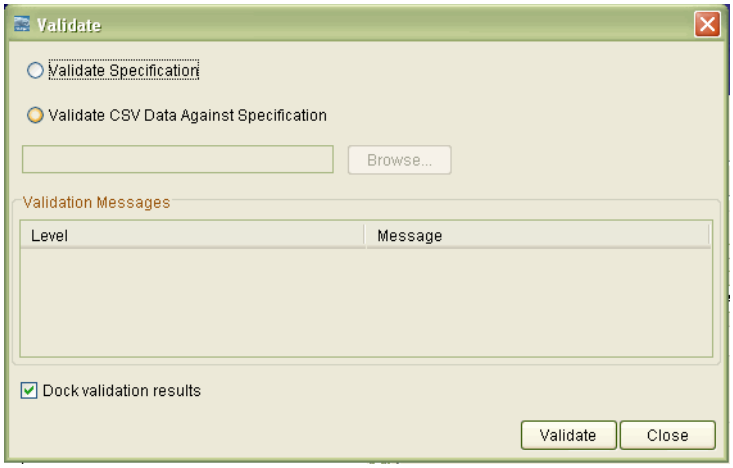


Figure 3.7 CSV Validate options

2. Select one of the following:
   - To validate the specification, click **Validate**.

     OR
   - Select **Validate CSV Data Against Specification** 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 appear. See *caAdapter Validation* on page 20 for more information on using the validation messages.

Saving a CSV Specification

When you are finished working on the CSV specification, do the following:

- From the menu bar, select **File > Save** or F**ile > Save As**.

  or
- Click the **Save** icon on the tool bar.

  This creates an XML-like file describing the tree structure. This file is portable and available for your or another user's future use.

Generating a Report

When the CSV Specification tab is selected, you can export the CSV specification to 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 appears.

A part of a generated CSV report is shown in *Figure 3.8*.

Segment Name	Field 1	Field 2	Field 3	Field 4
INVESTEVN	id_extension	code_code	code_displayName	text_inlineText
CASESERIOUSNESS_1	code_code	code_displayName	value_code	value_displayName
APPROVAL_2	id_extension	effectiveTime_low_value		
TERRITORY_21	code_code	code_displayName		
GOVERNINGAGENCY_22	id_extension	name_inlineText	name_suffix	
PLAYINGMANUFACTURER_23	id_extension	code_code	code_displayName	name_inlineText
PARTMEDICATION_3	code_code	code_displayName	name_inlineText	name_suffix
ASSIGNEDENTITY_4	code_code	code_displayName		
REPRESENTEDSCOPINGORGANIZATION_41	code_code	code_displayName	name_inlineText	name_suffix
ASSIGNEDORGANIZATION_42	id_extension	code_code	code_displayName	name_inlineText
TRIGGER_5	priorityNumber_value			
REACTION_51	code_code	code_displayName	text_inlineText	effectiveTime_value
INVESTIGATIVESUBJECT_511	id_extension			
SUBJECTAFFECTEDPERSON_5111	name_family_in	name_given_in	name_suffix_in	telecom_value

Figure 3.8 Part of a generated CSV report

Target Specification

This section includes the following topics:

- *HL7 V3 Specification* on page 28
- *Overview of HL7 V3 Specification Tab* on page 29
- *Defining Inline Text* on page 31
- *Defining Units of Measure* on page 31
- *Defining Default Data* on page 32
- *Defining Object Identifiers (OIDs)* on page 32
- *Adding Optional Clones to the HL7 V3 Specification* on page 33
- *Adding and Removing Multiples in the HL7 V3 Specification* on page 34
- *Updating Abstract Data Types in the HL7 V3 Specification* on page 35
- *Enabling and Disabling Force XML with an Optional Clone* on page 35

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 it must have either a default value or a mapping.

- If an element's cardinality is greater than one then you have a choice to add multiple fields.
- Only mandatory clones are included when a new HL7 V3 specification is first created. Optional clones may be added.
- nullFlavor field is optional.
- Address data types are only enabled with a pre-defined subset of its data fields. All other data fields can be optionally added or removed.

Overview of HL7 V3 Specification Tab

The HL7 V3 specification tab (*Figure 3.9*) 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.

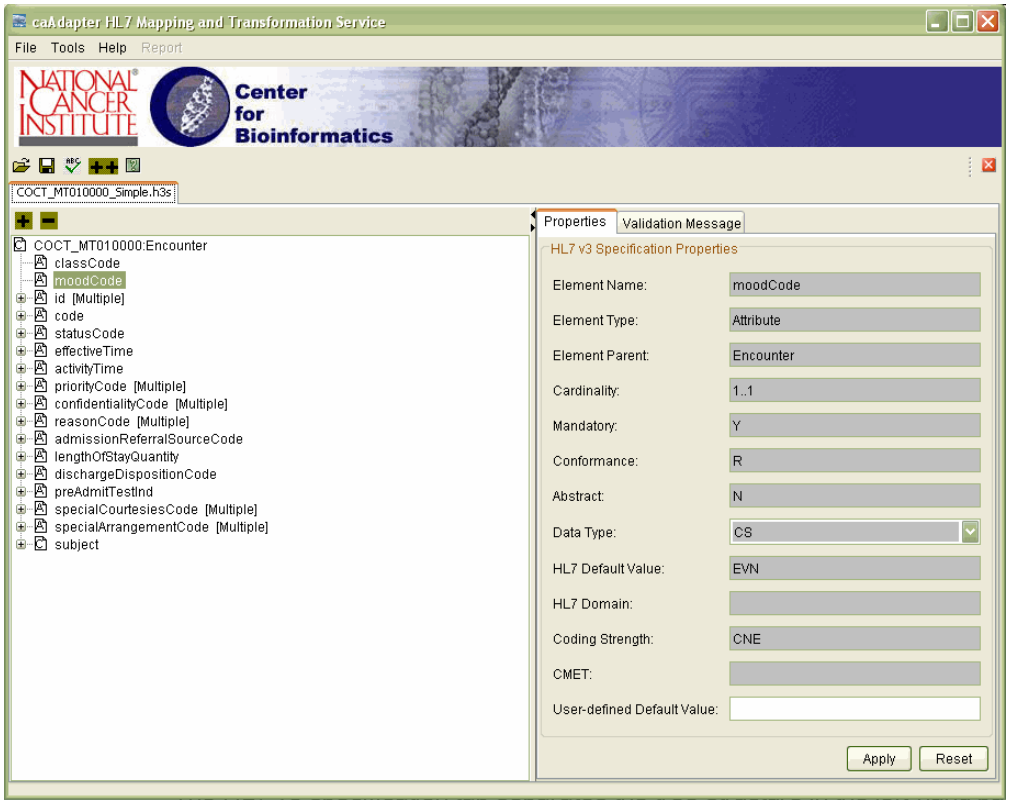


Figure 3.9 HL7 V3 Specification tab

The HL7 V3 specification tab 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

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 available actions as shown in the submenu in *Figure 3.10*. The available options are regular font.

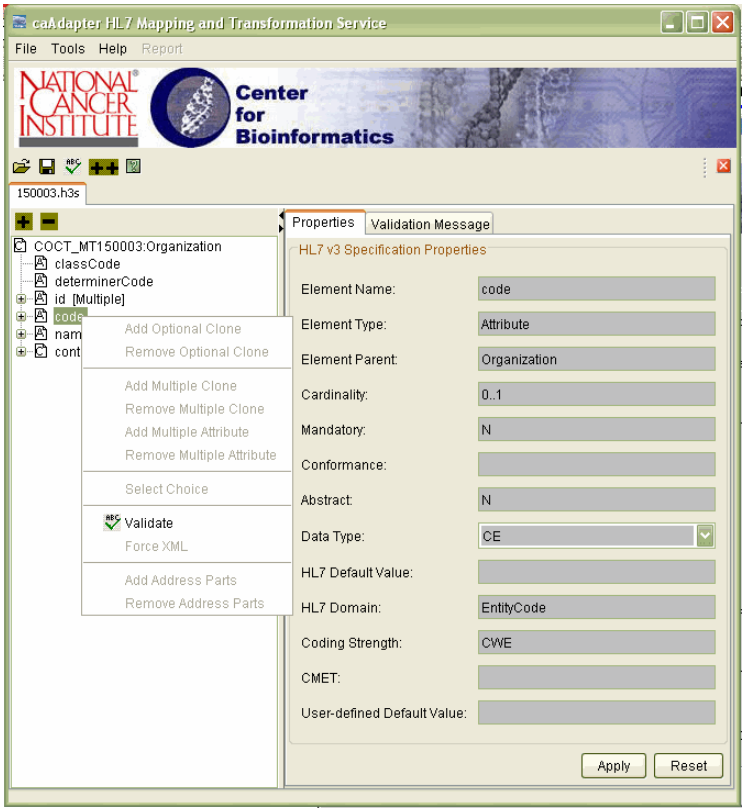


Figure 3.10 Options for HL7 V3 elements

The following sections describe how to create, update, validate, and save the HL7 V3 specification.

Creating an HL7 V3 Specification

The following steps describe how to create a new HL7 V3 specification.

1. Select **File > New > CSV to HL7 V3 Mapping and Transformation Service > HL7 V3 Specification** to display the **HL7 V3 Specification** dialog box with valid message types.

2. Select the appropriate HL7 Normative, message category, and message type from the drop-down lists (*Figure 3.11*) and click **OK**.

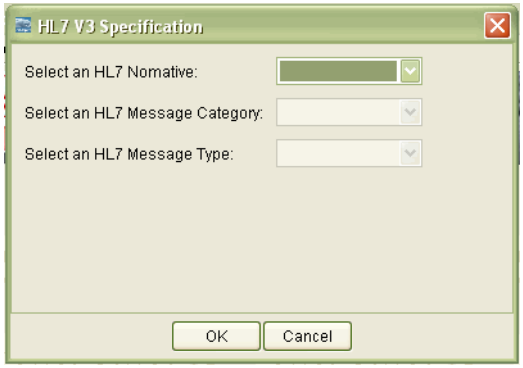


Figure 3.11 HL7 V3 Specification dialog box

Currently, the mapping tool supports all message types as defined by HL7 standards

A new HL7 V3 specification tab appears with the name `untitled_1.h3s`.

Opening an Existing HLV V3 Specification

To open an existing HLV V3 specification:

1. Select **File > Open > HL7 V3 Specification (.h3s) or File>Open >HL7 V3 Specification (.xml)**. The **Open HL7 V3 Specification (H3S) File** dialog box appears.
2. Select a **File name** to open and click **OK**. The HL7 V3 specification displays in a tab with the name of the file and its extension.

Defining Inline Text

The data type field **inline Text** is caAdapter's way of referring to text that appears between XML tags as opposed to being a value assigned to an XML attribute. The names of data types designed with inline text fields are configured within the caAdapter `.properties` file under the item: `caadapter.hl7.attribute.inlinetext.required`.

This attribute defines a list of data types that require inline text. If you want include more data types, reset the value of `caadapter.hl7.attribute.inlinetext.required`.

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 data type field values. These defaults allow you to assign values for data type fields 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 3.3* shows the expected behavior for attributes that are mapped with a CSV value, mapped with a null CSV value and unmapped data types.

	Mapped to Non-Null Field	***Mapped to Null Field***	***Unmapped***
Optional	CSV Value	Default Value	Element not created unless other sibling fields are mapped
Optional (Force XML)	CSV Value	Default Value	Default Value
Required	CSV Value	Default Value	Default Value
Mandatory	CSV Value	Default Value	Default Value

Table 3.3 Default value behavior

"Optional" means that the element is optional in the target message. It is not required if it has not been mapped.

"Optional (Force XML)" means that the element is optional in the HL7 MIF specification, but it is required by the user to create an empty element with a default value.

"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:

- HL7 OIDs
- Existing OIDs

In HL7, OIDs are assigned within the appropriate branch of the HL7 OID root (2.16.840.1.113883). If you are interested in assigning an OID to a scheme, be sure to check that the scheme you are assigning does not already have an OID assigned to it within the HL7 OID hierarchy. The process of registering an existing OID with HL7 involves adding an OID and its descriptive data to a central registry. The OID does not have to be within the HL7 root OID or any other specific 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 field, you must assign the OID in the User-defined Default Value field or you must have a map.

Adding Optional Clones to the HL7 V3 Specification

The ability to add or remove a clone is the way caAdapter 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 Optional Clone** and **Remove Optional Clone** options.

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

1. Right-click an element name with an optional or recursive relationship and select **Add Optional Clone**. The **Clone List** dialog box appears (*Figure 3.12*).

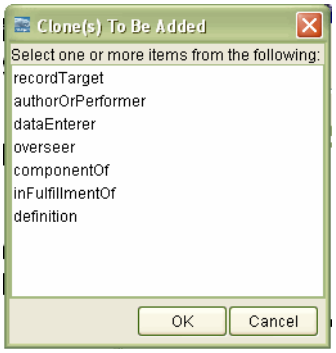


Figure 3.12 Clone List dialog box

2. In the Clone List dialog box, select one or more of the unused optional 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 clones you just added. This is the case with a recursive association, where you could continue adding recursive levels to an arbitrary level as needed by adding an optional clone.

Perform the following steps to remove optional clones.

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

Adding and Removing Multiples in 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.

Do the following to add multiple clones.

- Right-click a clone that contains a **[Multiple]** label (or a **[1]** label, which is the first of this group of multiple clones) and select **Add Multiple Clone**.

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

Do the following to remove multiple clones.

1. Right-click a clone with the **[xx]** label (where xx is a number greater than one), indicating that it is a replicated clone, the **Remove Multiple Clone** is enabled, select **Remove Multiple Clone**. One multiple of the element is removed from the tree structure.
2. Perform the following steps to add multiple attributes.
   a. Right-click an attribute with the **[1]** label, indicating that it is a replicated attribute. The **Remove Multiple Attribute** is enabled.
   b. Select Re**move Multiple Attribute**.

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

Do the following to add multiple attributes.

1. Right-click an attribute with the **[1]** label, indicating that it contains multiple numbered labels, and select **Add Multiple Attribute**.
2. One multiple of the attribute is added to the tree structure.

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

Do the following to remove multiple attributes.

1. Right-click an attribute with the **[1]** label, indicating that it contains multiple numbered labels, and select **Remove Multiple Attribute**.
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 to select the data type. Click **Apply**. After assigning a concrete data type with an abstract data type, the system retrieves the data fields of the assigned data type and attaches those to the original attributes accordingly.
3. Repeat steps 1 and 2 above if you need to change a different concrete data type.

Using Choice Boxes in the HL7 V3 Specification

You can only select one choice item at a time for a given element.

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

1. Right-click 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 (*Figure 3.13*).

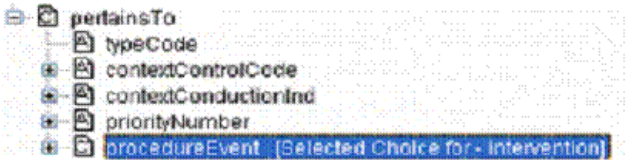


Figure 3.13 Selected choice

Note: Since a business rule for an HL7 V3 specification specifies a choice must be selected, there is no option to unselect a choice. However, if the parent association is optional, the association can be dropped and re-added.

3. Repeat the above steps if you want to select a different choice item.

Enabling and Disabling Force XML with an Optional Clone

If a clone is optional for the target message specification, right-click the tree node to make **Enable Force XML** active (*Figure 3.14*).

If a clone is optional for the target message specification, and it has been enabled, right-click the tree node to enable **Disable Force XML**. Enable Force XML or Disable

Force XML informs the HL7 message transformation engine whether or not to create an empty element if no mapping has been set (*Figure 3.15*).

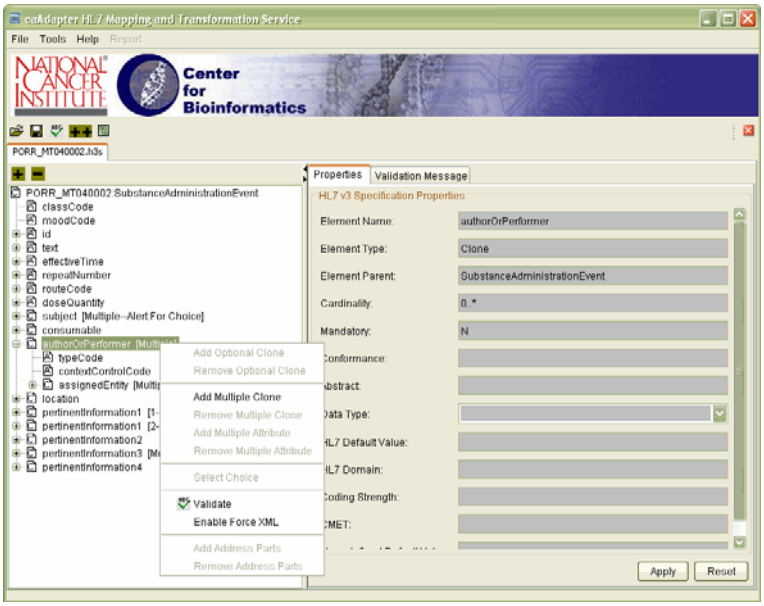


Figure 3.14 Enable Force XML

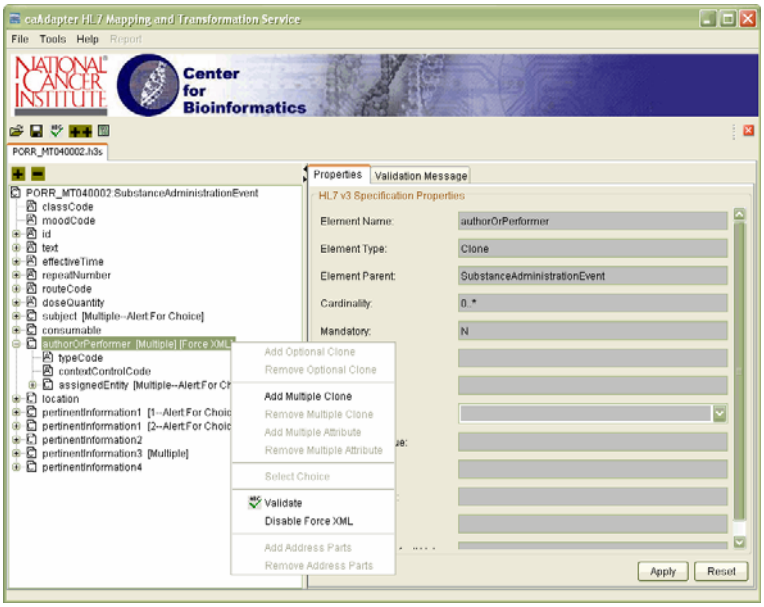


Figure 3.15 Disable Force XML

Adding and Removing Parts of an Address Data Type

If the system has predefined a subset of data fields for an attribute with an Address data type, other data fields can be added or removed (*Figure 3.16* and *Figure 3.17*).

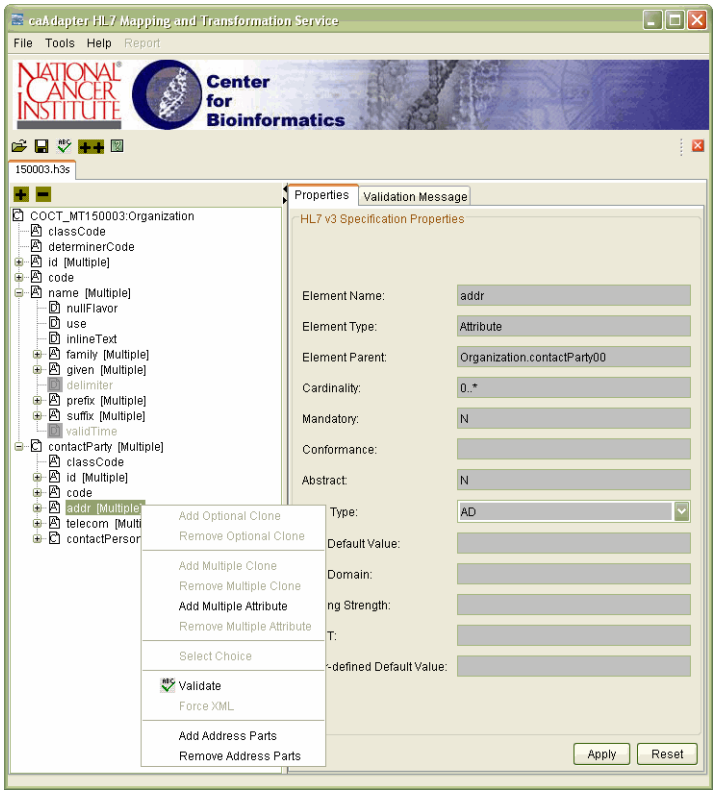


Figure 3.16 Adding or removing parts of an Address data type

Figure 3.17 Modifying an Address data type

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 any children and further descendants below it in the tree structure.

To validate the HL7 V3 specification

1. Choose one of the following ways to validate the HL7 V3 specification:
   - select **File > Validate**,
   - select the **Validate** icon from the tool bar, or
   - right-click a clone and select **Validate** to perform the validation.

   A Message dialog box appears (*Figure 3.18*), indicating the status of the validation.

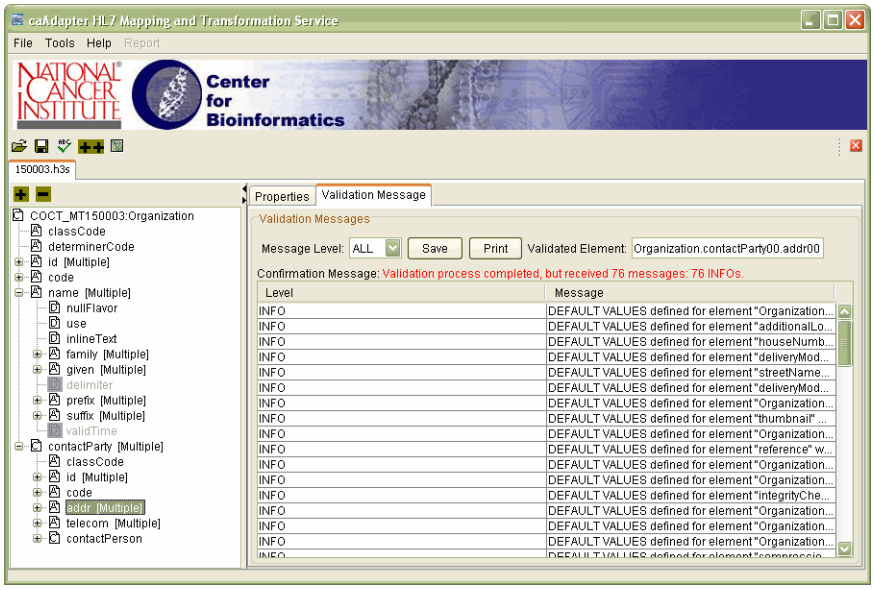


Figure 3.18 HL7 V3 specification validation

2. Click **OK**. The messages display in the **Validation Messages** panel (see *caAdapter Validation* on page 20).

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. If the specification is being saved for the first time, the system saves it in `.h3s` format (*Figure 3.19*).

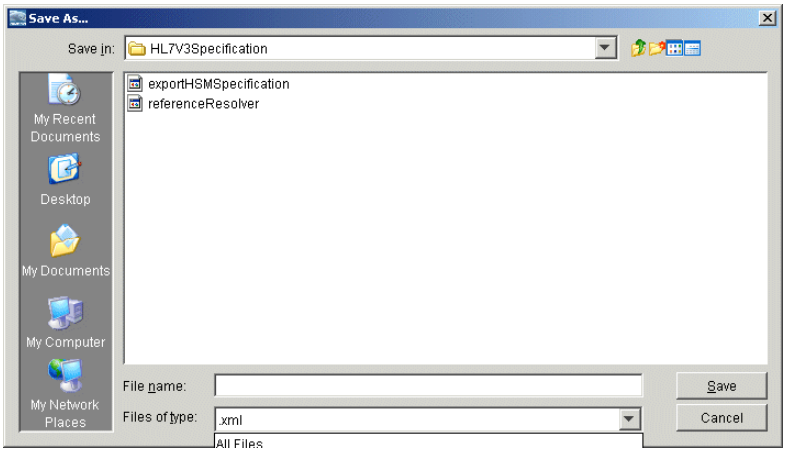


Figure 3.19 Saving HL7 specifications

Map Specification

A mapping defines the relationship between two specification elements. caAdapter provides

- a direct link from the source element to a target element, or
- a link from a source element to a function's input port, or
- a link from a function's output port to a target element.

Map Specification Business Rules

Following are the business rules for a map specification:

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

Creating Mappings

This section contains the step-by-step instructions to create the mappings. See *Appendix A, caAdapter Example Data*, on page 109 for detailed information on mapping scenario 8 included with the example data.

This section includes the following topics:

- *Overview of the Map Specification Tab* on page 39
- *Creating and Opening a Map Specification* on page 40
- *Creating a Mapping Link* on page 41
- *Deleting a Mapping Link* on page 44
- *Using Functions in Map Specifications* on page 45
- *Validating the Map Specification* on page 47
- *Saving a Map Specification* on page 47
- *Generating a Map Specification Report* on page 48

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 target specification must be `.h3s` format.

The Map Specification tab (*Figure 3.20*) consists of the following:

- **Two tree panels** - contain 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 and target elements 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.)

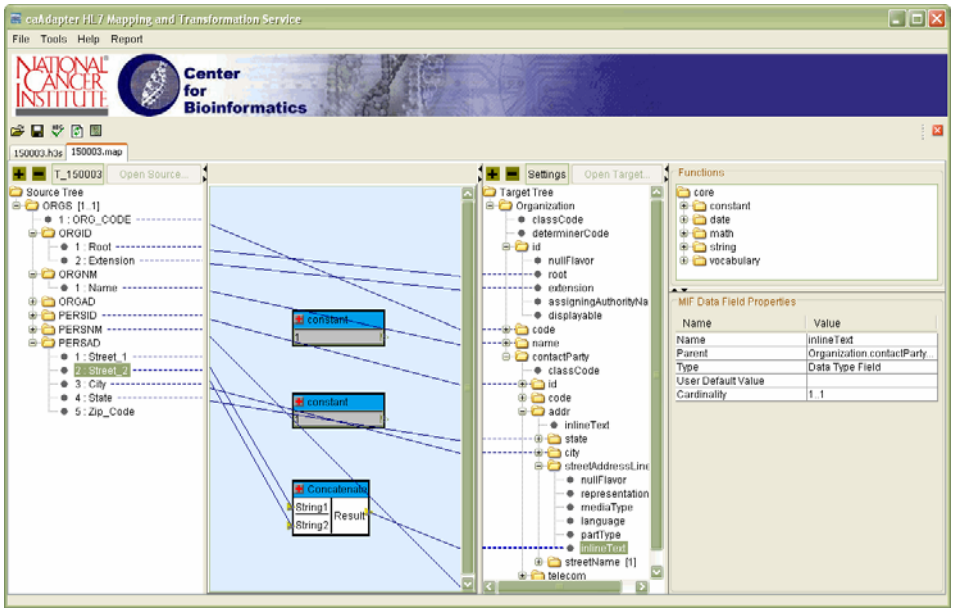


Figure 3.20 Map Specification tab

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.

Caution: 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 or producing other unpredictable behavior.

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

Creating and Opening a Map Specification

Perform the following steps to create a new map specification.

1. Select **File > New > CSV to HL7 V3 Mapping and Transformation > 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 panel 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 panel with its tree structure.

Perform the following steps to open an existing map specification.

1. Select **File > Open > CSV to HL7 V3 Map Specification**. The Open Map File dialog box appears.
2. Select the map specification file and click **Open** to display the source and target trees along with any existing mappings.

Creating a Mapping Link

Perform the following steps to create a mapping link.

1. Select a source element and drag it to the appropriate target element. The cursor indicates if the source is not allowed to be mapped to the target element (*Figure 3.21* and *Figure 3.22*).

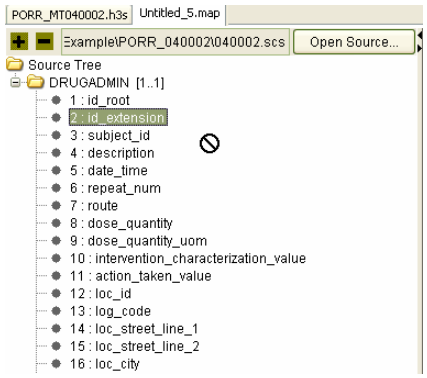


Figure 3.21 Mapping is not allowed

The cursor indicates when the source can be mapped to the target element.

2. Drop the source on the target element.

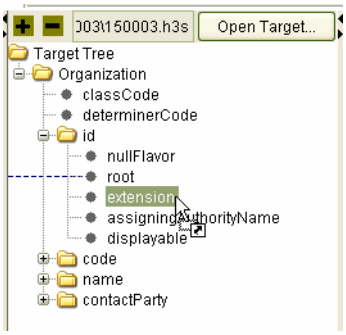


Figure 3.22 Mapping is allowed

Once a source field is mapped to a target element, a mapping line appears between them in the mapping panel. *Figure 3.23* shows a mapping line between **Root** and **root.**

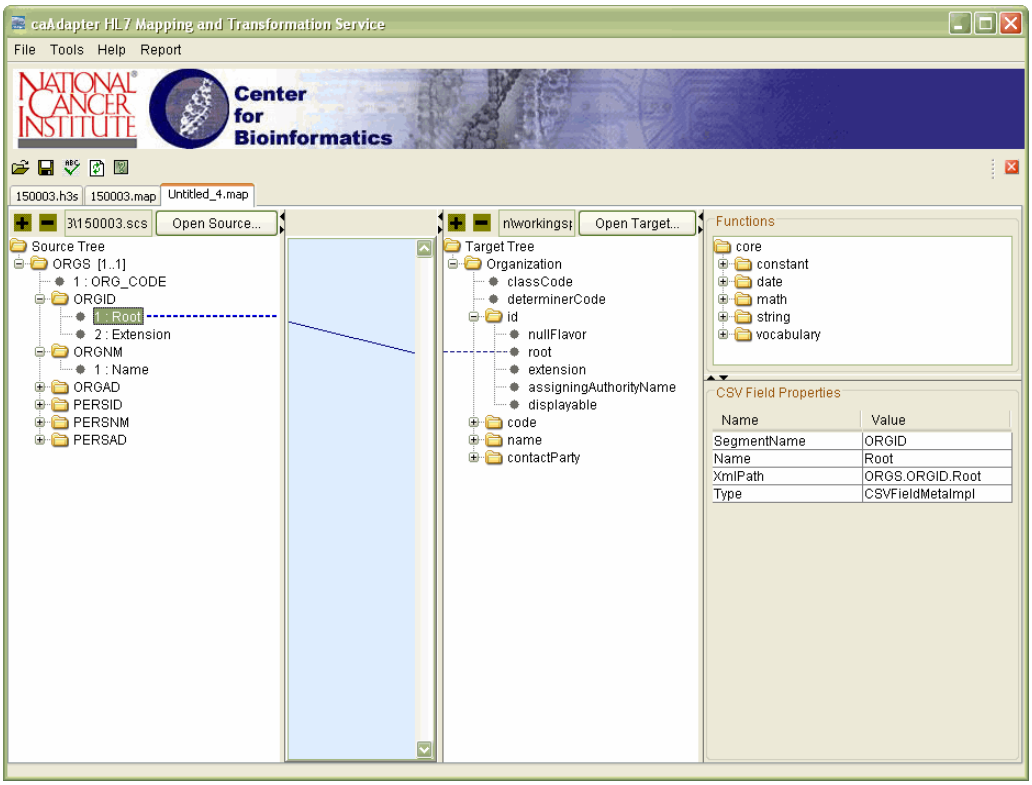


Figure 3.23 Mapping line between a source field and target element

Tip: To delete a mapped line or a function in the center panel, select the item you want to delete, right-click it, and select **Delete** or **Delete All**. Click **Yes** to confirm the deletion. The selected item is deleted from the mapping.

The **Properties** panel displays information about the selected element.

Note that when you select a source element, the **CSV Field Properties** appear (see *Figure 3.23*).

When you select a mapping line, the **Link Properties** appear (*Figure 3.24*).

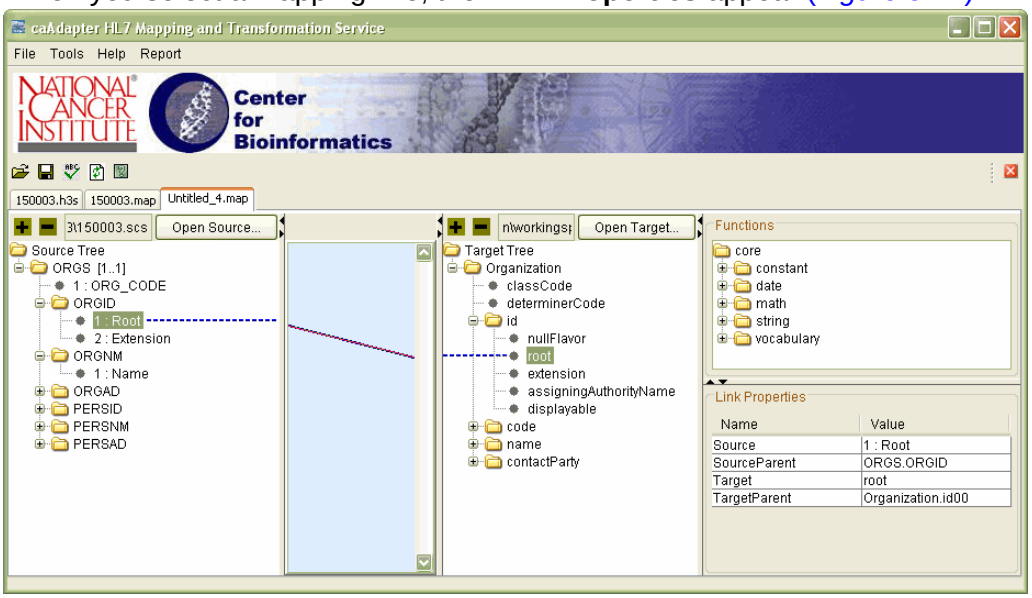


Figure 3.24 Link Properties pane

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** appear (*Figure 3.25*).

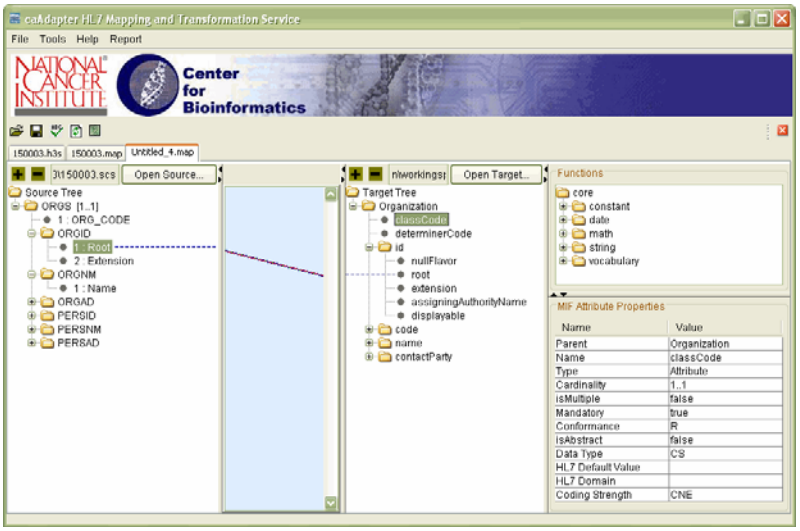


Figure 3.25 Attribute properties

When you select a function in the **Mapping** panel or in the **Functions** panel, the **Function Properties** appear. When you select a function group in the **Functions** panel, the **Function Group Properties** appear (*Figure 3.26*).

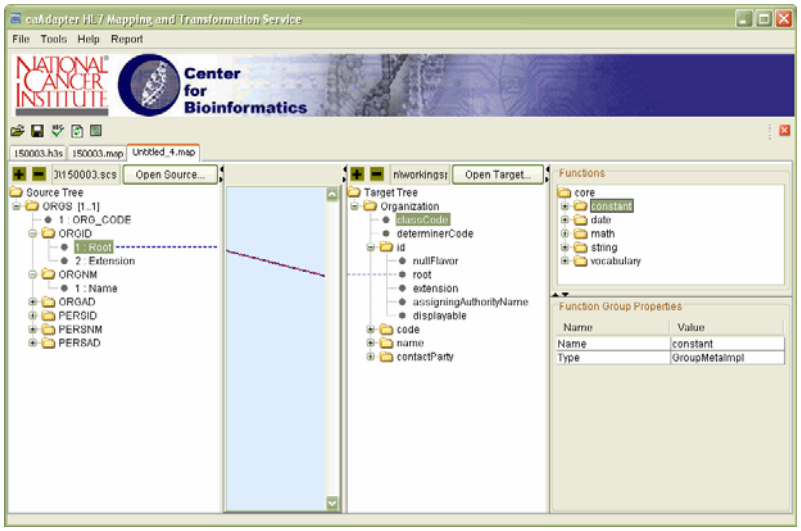
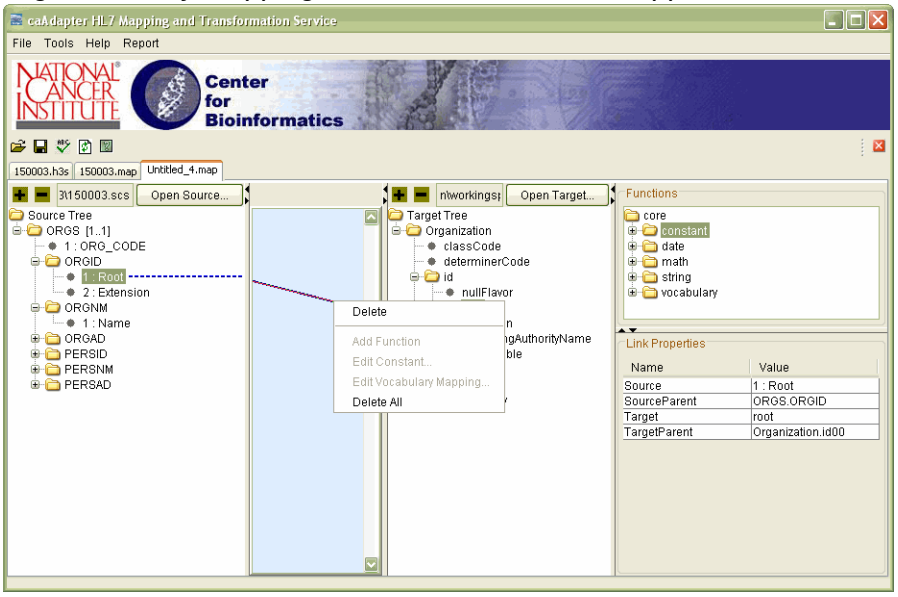


Figure 3.26 Function Group Properties

Deleting a Mapping Link

To delete a mapping link

1. Right-click any mapping line to select it. A menu appears.



2. Select **Delete** to delete only the selected mapping line. Select **Delete All** to delete all mapping lines (including those you have not selected). A message appears asking you to confirm the deletion(s). If you select Yes, the mapping line or lines are deleted.

Using Functions in Map Specifications

The **Functions** panel (*Figure 3.27*) 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.

When a function is selected in the function library, its properties information appears, such as name and number of input and output parameters, in the **Function Properties** panel (*Figure 3.27*).

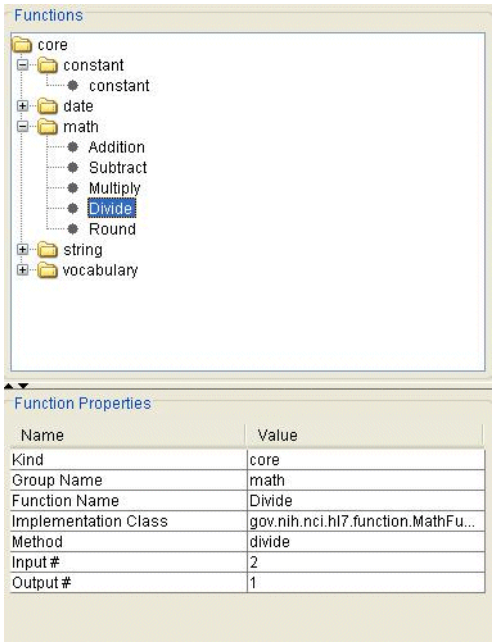


Figure 3.27 Functions in mapping specification

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 map.
2. Drag-and-drop the source field(s) onto the input parameters. *Figure 3.28* shows the selected field **text** being dropped as an input to the **Initcap** function.

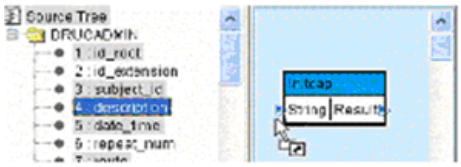


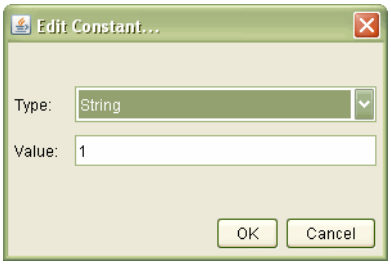
Figure 3.28 Adding input to a function

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**. The Edit Constant dialog box appears.



2. 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 3.4* 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 3.4 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 (*Figure 3.29*) appears.

Figure 3.29 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 (*Figure 3.30*) appears, notifying 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.

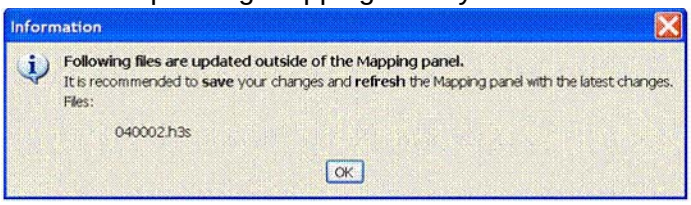


Figure 3.30 Refresh mapping panel recommendation

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. A **Message** dialog box appears indicating the status of the validation.
2. Click **OK**. The detailed messages appear in the **Validation Messages** dialog box (see *caAdapter Validation* on page 20).

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.

Caution: 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`); not just the map specification. Furthermore, the CSV and HL7 V3 specification files must be in the same path locations as they were on the machine where they were created. Alternatively, the path name can be manually removed by editing the `.map` file however this is dangerous and unpredictable results may occur if the file is changed improperly.

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 message window appears when the report has been successfully generated.

   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 elements:
   - Source - Unmapped_Source
   - Target - Unmapped_Target

HL7 V3 Message

Generating the HL7 V3 message is the end goal in using the mapping tool. XML HL7 message instances are created using the map specification and a corresponding CSV data file.

HL7 V3 Message 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.

Generating HL7 V3 Messages

This section includes the following topics:

- *About the HL7 V3 Message Tab* on page 49
- *Starting the Conversion Process* on page 49
- *Using the HL7 V3 Messages Tab* on page 51
- *Saving an HL7 V3 Message* on page 52

- *Reusing the HL7 V3 Message Tab* on page 52
- *Starting the Conversion Process* on page 53
- *Saving the CSV Data File* on page 54

About 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 (*Figure 3.31*) contains the following four panels:

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

Figure 3.31 HL7 V3 Message tab

Starting the Conversion Process

Perform the following steps to convert a data file into an HL7 V3 message.

Note: There is no **File > Open** option that corresponds to HL7 V3 messages since you should always generate fresh messages based on the current selection of source and map files.

1. Select **File > CSV to HL7 V3 Mapping and Transformation > HL7 V3 Message** from the menu bar to display the **HL7 V3 Message** dialog box (*Figure 3.32*).

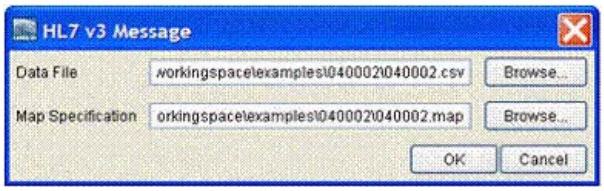


Figure 3.32 HL7 V3 Message dialog box

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 is defined and configurable in the source distribution), then the **Question** dialog displays as shown in *Figure 3.33*. Click **Yes** to start the process or click **No** to abort the process given the estimated time.

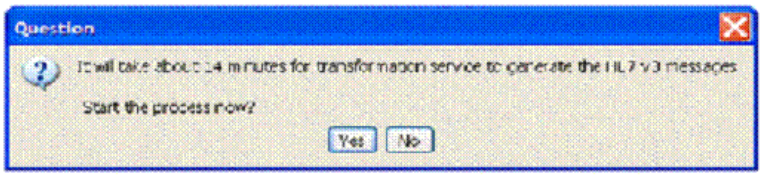


Figure 3.33 HL7 V3 message generation confirmation

8. After a **Yes** confirmation, the process starts and a progress dialog box appears (*Figure 3.34*). The system monitors the transformation progress for both loading the data, which includes reading the map file, source and target data specification; and the count of messages generated.

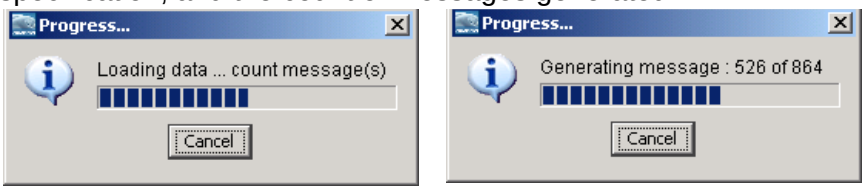


Figure 3.34 HL7 V3 message generation progress dialog

9. Once the process starts, you can cancel the process by clicking **Cancel**. If cancelled, the underlying generation process ends and a dialog box appears (*Figure 3.35*).

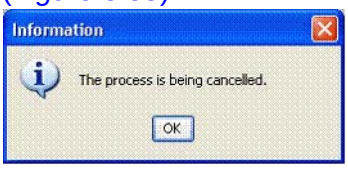


Figure 3.35 HL7 V3 message generation cancelled

A message appears (*Figure 3.36*) when the process ends.

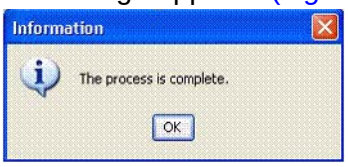


Figure 3.36 HL7 V3 message process complete

10. Click **OK**. The **HL7 V3 Message** tab appears.

Using the HL7 V3 Messages Tab

The two main features of the HL7 V3 Message tab (*Figure 3.37*) are the two scrollable text fields containing an XML instance and the associated error, warning and/or informational messages generated during the conversion process.

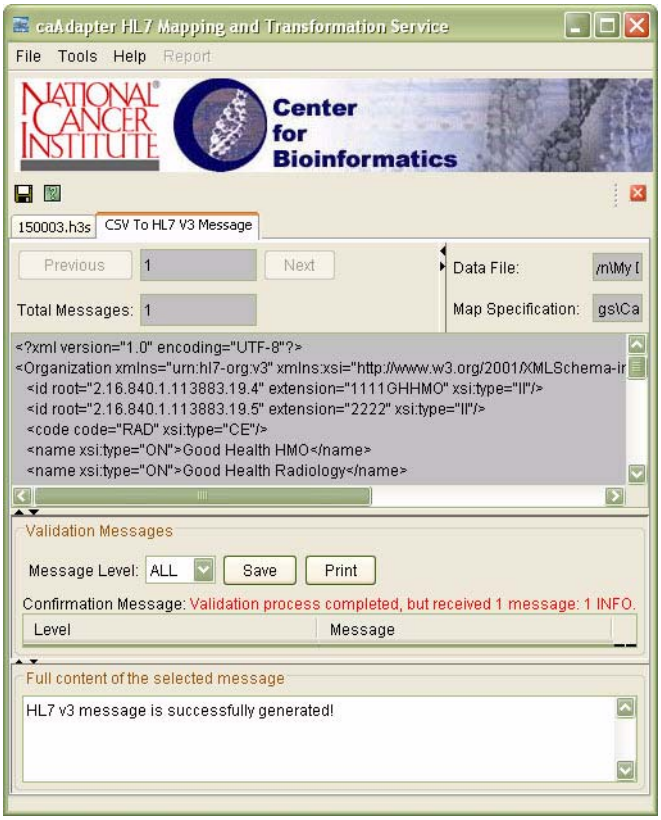


Figure 3.37 HL7 V3 Message tab

Click the **Previous** and **Next** buttons to cycle through the XML messages one at a time. As the messages change, the validation messages change. See *caAdapter Validation* on page 20 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, `example_message_1.xml`, `example_message_2.xml`, `example_message_3.xml`).

Note: Validation messages are not saved with their corresponding XML message and must be saved separately using the Save button in the Validation Messages panel.

Transforming an HL7 Message into CSV Format

This version of caAdapter allows you to map and transform HL7 messages into CSV files.

HL7 V3 to CSV Transformation Business Rules

The business rules are similar to those that apply to transforming CSV data into HL7 V3 messages with the source and target reversed, that is, transforming HL7 V3 into CSV file.

Transforming HL7 V3 Messages to CSV Format

This section includes procedures for transforming an HL7 V3 message to a CSV file.

It includes the following topics:

- *Reusing the HL7 V3 Message Tab* on page 52
- *Starting the Conversion Process* on page 53
- *Saving the CSV Data File* on page 54

Reusing the HL7 V3 Message Tab

You reuse the HL7 V3 Message tab when you want to generate and view the CSV data generated from the HL7 V3 message data. Each data file may have one or more logical records that result in a corresponding number of CSV meta instances (or more depending on the structure of the mapping). The user interface allows navigating

between the various instances. The reused HL7 V3 message tab (*Figure 3.38*) contains four panels:

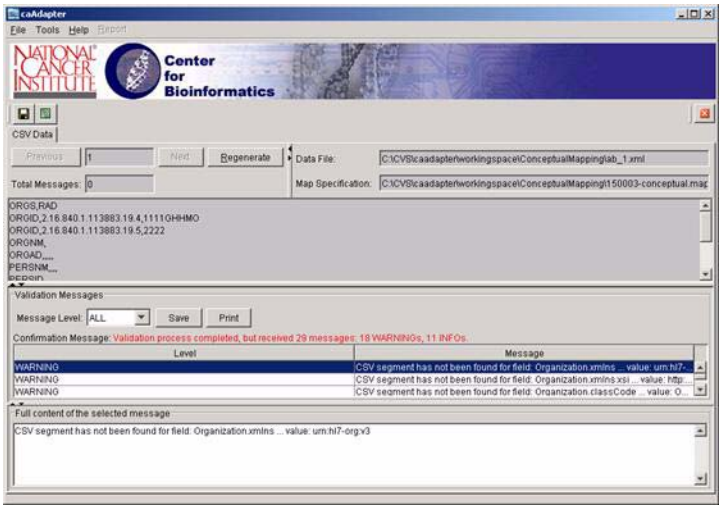


Figure 3.38 Displaying CSV Data with HL7 V3 Message Tab[can't figure out how to recreate this]

Starting the Conversion Process

Do the following to convert an HL7 V3 message into a CSV data file.

Note: There is no **File > Open** option that corresponds to CSV data file since you should always generate fresh CSV data based on the current selection of the source HL7 V3 message and the map files.

1. Select **File > New > HL7 V3 To CSV Transformation Service> New HL7 V3 To CSV** from the menu bar to display the **HL7 V3 To CSV** dialog box (*Figure 3.39*).

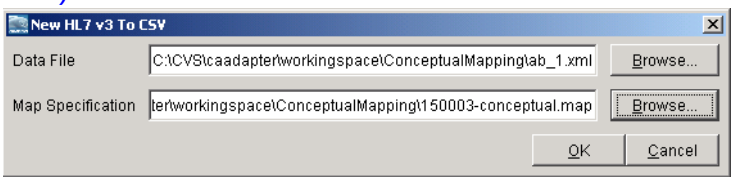


Figure 3.39 HL7 V3 to CSV Dialog Box

2. Click **Browse** next to **Data File**. The **Open Data File** dialog box appears.
3. Select the data file to use in the conversion process and click **Open**.
4. Click **Browse** next to **Map Specification**. The **Open Map Specification** dialog box appears.
5. Select the map specification file to use in the conversion process and click **Open**.
6. Click **OK** to generate CSV data file from the selected files.
7. Click the **Previous** and **Next** buttons to cycle through the CSV data one at a time. As the data change, the validation messages change as well. See *caAdapter Validation* on page 20 for more information on the validation

messages. Click the **Regenerate** button to regenerate the data from scratch using the same data file and map specification.

Saving the CSV Data File

To save the data file, select **File > Save** or **File > Save As** or click the Save icon on the toolbar.

CHAPTER 4

HLV V2 TO HL7 V3 CONVERSION

This chapter provides instructions on using caAdapter to map and convert an HL7 V2 message to an HL7 V3 message.

Topics in this chapter include:

- *Converting HL7 V2 to HL7 V3 Messages* on page 55
- *Understanding the Mapping and Transformation Processes* on page 55

Converting HL7 V2 to HL7 V3 Messages

You can convert an HL7 V2 message to an HL7 V3 message in the following two ways:

Using XML formatted V2 resources, which can be freely downloaded at the HL7 home page, V2 messages can be directly transformed to V3 message without any pre-processing. Since this method does not include V2 syntax and vocabulary validation, it is strongly recommended that you use another tool to validate and filter the input V2 message. Otherwise, the result may be unpredictable.

The current V2 related functions in the caAdapter GUI menus and all the explanations in this chapter are applying the later method. If you need to use the legacy method, refer to earlier versions of the *caAdapter User's Guide* or contact NCICB Application Support.

Understanding the Mapping and Transformation Processes

When you transform an HL7 V2 file to HL7 V3 format, an SCS file is not required. You must only select the V2 version and message type in the mapping panel.

The HL7 V2 to HL7 V3 transformation process has the following three steps.

- Step 1: Generate an H3S file for the target HL7 V3 message (see *Step 1: Generating the H3S File* on page 56)

- Step 2: Create a `.map` file between the source HL7 V2 message type and the target H3S file (see *Step 2: Creating the .map file* on page 56)
- Step 3: Transform and generate a V3 message from the source HL7 V2 message file using the `.map` file (see *Step 3: Transforming a Message from V2 to V3* on page 60)

Step 1: Generating the H3S File

Generating an H3S file for HL7 V2 to HL7 V3 mapping follows the same process as explained in *Transforming an HL7 Message into CSV Format* on page 52.

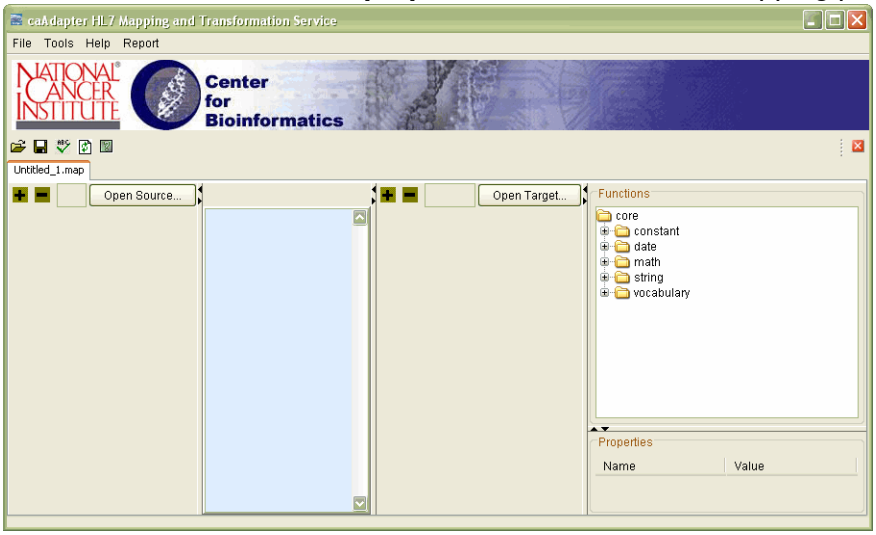
Note: Knowing which V3 message type is appropriate for a given a V2 message type is extremely important.

Proceed to *Step 2: Creating the .map file*, below.

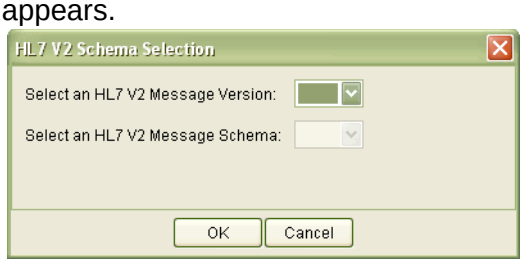
Step 2: Creating the `.map` file

Do the following to create the `.map` file.

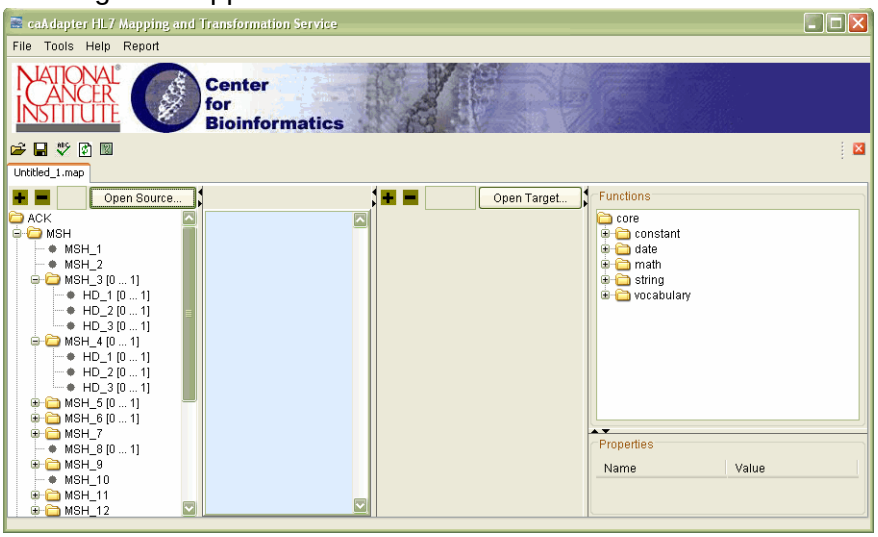
1. Select **New** > **HL7 V2 To HL7 V3 Mapping and Transformation Service** > **New HL7 V2 to HL7 V3 Map Specification**. The main mapping panel appears.



2. Click the **Open Source** button. The HL7 V2 Schema Selection dialog box appears.



3. Select the HL7 V2 message version and schema. Click **OK**. The HL7 V2 message tree appears on the left side of the window.



4. Select a source element and drag it to the appropriate target element. The cursor indicates if the source is not allowed to be mapped to the target element (*Figure 4.1* and *Figure 4.2*).

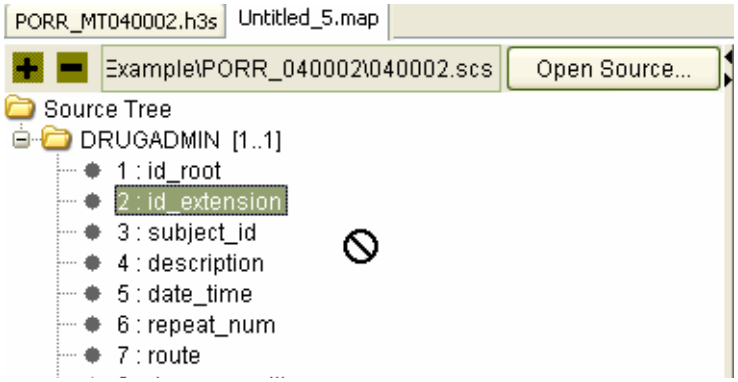


Figure 4.1 Mapping is not allowed

The cursor indicates when the source can be mapped to the target element.

5. Drop the source on the target element.

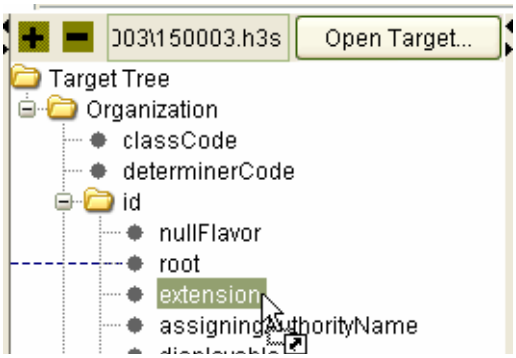


Figure 4.2 Mapping is allowed

Once a source field is mapped to a target element, a mapping line appears between them in the mapping panel. *Figure 4.3* shows a mapping line between **id_root** and **root.**

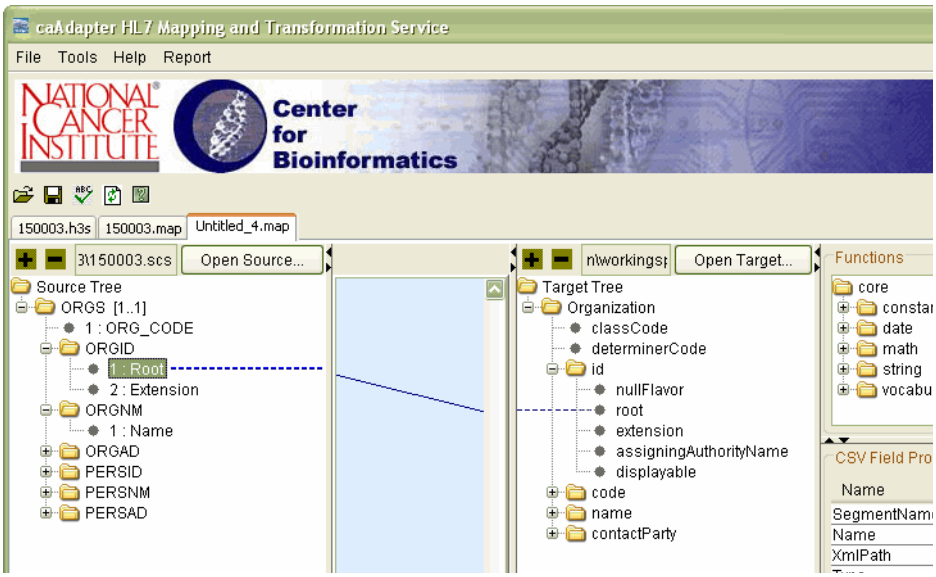


Figure 4.3 Mapping line between a source field and target element

Tip: To delete a mapped line or a function in the center panel, select the item you want to delete, right-click and select **Delete**. 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** appears (*Figure 4.4*).

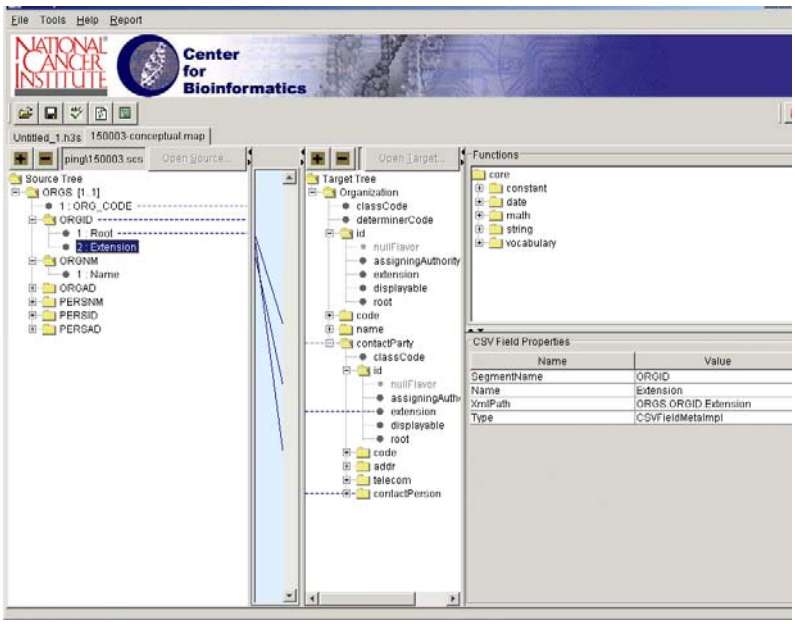


Figure 4.4 CSV Field Properties panel

When you select a mapping line, the **Link Properties** appears (*Figure 4.5*).

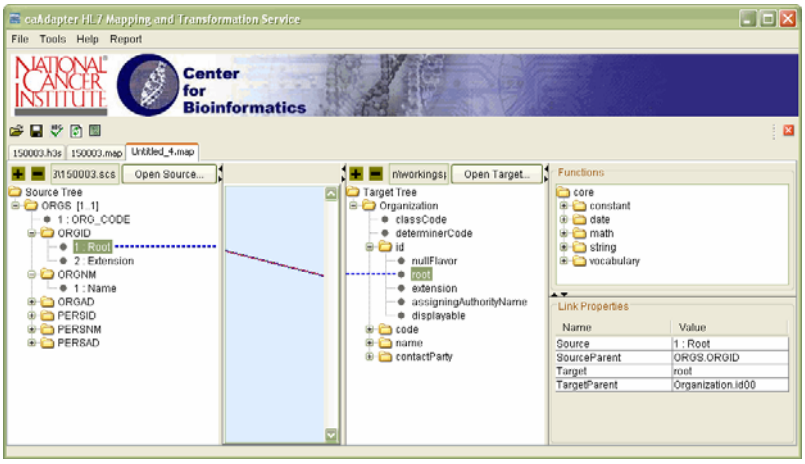


Figure 4.5 Link Properties pane

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** appears (*Figure 4.6*).

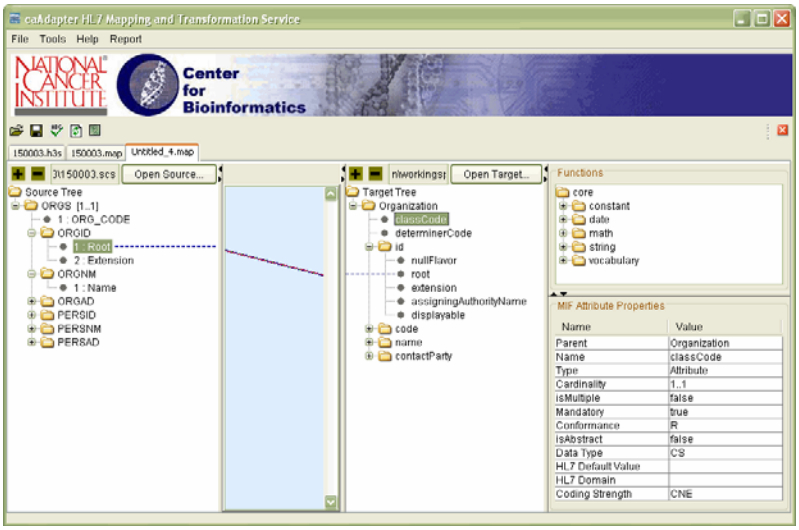


Figure 4.6 Attribute properties

When you select a function either in the **Mapping** panel or in the **Functions** panel, the **Function Properties** appear. When you select a function group in the **Functions** panel, the **Function Group Properties** appear (*Figure 4.7*).

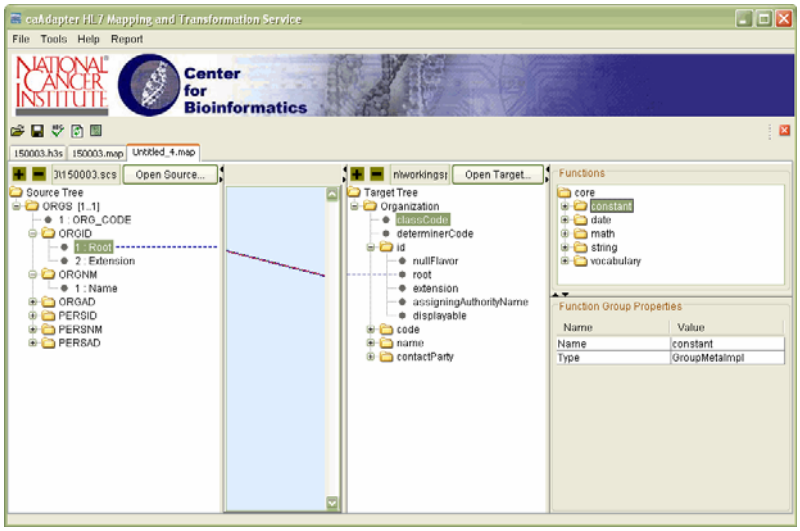


Figure 4.7 Function Group Properties

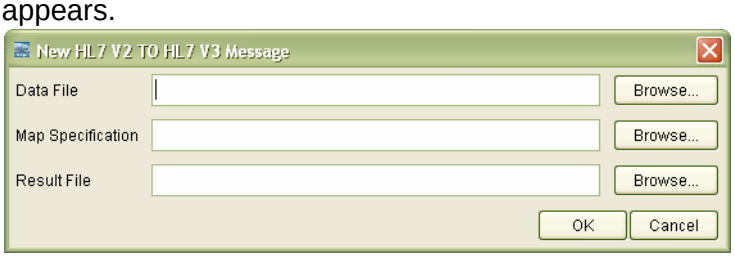
6. Proceed to *Step 3: Transforming a Message from V2 to V3*, below.

Step 3: Transforming a Message from V2 to V3

When you transform a message from HL7 V2 to HL7 V3, your input file type is .hl7. You must save your source HL7 V2 message as an .hl7 file before transforming it.

To transform a message from HL7 V2 to HL7 V3, do the following:

1. Select New > HL7 V2 to HL7 V3 Mapping and Transformation Service > New HL7 V2 to HL7 V3 Message. The New HL7 V2 to HL7 V3 Message dialog box appears.



2. In the Data File field, click **Browse**. The Open dialog box appears. Locate an `.hl7` file and then click **OK**. The New HL7 V2 to HL7 V3 Message dialog box appears.
3. In the Map Specification field, click **Browse**. The Open dialog box appears. Locate a `.map` file and then click **OK**. The New HL7 V2 to HL7 V3 Message dialog box appears.
4. In the Result File field, click **Browse**. The Open dialog box appears. Locate a `.zip` file and then click **OK**. The New HL7 V2 to HL7 V3 Message dialog box appears.

5. Click **OK**.

Using APIs for HL7 V2 to HL7 V3 Transformation

You can use APIs to transform HL7 V2 files to HL7 V3 format. Refer to *HL7 V2 to HL7 V3 Transformation* on page 79 for more information, keeping in mind that the source data file type for an HL7 V2 to HL7 V3 transformation is `.hl7`.

Note: Earlier versions of caAdapter required two steps to transfer HL7 V2 messages to HL7 V3. First the messages had to be converted to CSV using KNU (Kyungpook National University) V2 parser engine. This process includes V2 message syntax and vocabulary validating, and can process segments such as 'OBX'. In the second step, caAdapter transformed the CSV file to an HL7 V3 message. If you have a legacy application using this earlier process, legacy APIs are included in the caAdapter distribution.

CHAPTER

5

USING FUNCTIONS IN MAPPING

This chapter describes the different functions provided by caAdapter. Topics in this chapter include:

- *Functions Provided by caAdapter* on this page
- *Function Specifications* on page 65
- *Function Specification Overview* on page 65
- *Adding Functions to the Function Library* on page 75

Functions Provided by caAdapter

caAdapter provides a variety of basic functions as part of the initial installation. These functions may be used in any mapping where the function panel is available. There are five groups of functions:

- constant – There is one function in this group that allows the user to define a value that can be used as input with other functions
- date – There is one function in this group that allows the user to convert any date format into the HL7 V3 required date format.
- math – Five basic math functions are provided in this group.
- string – Ten commonly used functions in this group allows users to do basic data manipulation.
- vocabulary – These functions were new in the 3.2 release of caAdapter and allow a user to translate values in incoming data into a different value in the outgoing format.

The following table provides a simple overview of the functions that reside in each of these groups.

Function Group Name	Function Name	Function Description
constant	Constant	Allows the user to define a string or integer for use as an input value to another function or to a target field.
Date	changeFormat	Requires the user to define the incoming date format (using either a constant function or a source field mapping) and the date field to be converted. Only transforms to the HL7 V3 required date format, but does handle varying levels of specificity (e.g. with or without time).
math	Addition	Takes in two values and provides the sum.
math	Subtract	Takes in two values and provides the difference.
math	Multiply	Takes in two values and provides the product.
math	Divide	Takes in two values and provides the quotient.
math	Round	Takes in two values, a value to be rounded, and the digit number to which to round.
string	Concatenate	Takes in two strings and provides a single value have the first string appended with the second.
string	Split	Takes in a string and a position number and breaks the string into two strings at the given position.
string	Length	Takes in a single string and provides the number of characters present.
string	Substring	Takes in a string and a starting and ending position, returning a portion of the string.
string	Trim	Takes in a single string and provides the same basic value with leading and trailing blanks removed.
string	Replace	Takes in three strings, one containing the value to be operated on, one containing the "from" characters to search for, and the last containing the "to" characters to substitute, producing a single string with "from" characters substituted with "to" characters.
string	Instring	Takes in a string on which to operate and a pattern to search for, returning the position within the string where the pattern is found, or 0.

Table 5.1 caAdapter Functions

Function Group Name	Function Name	Function Description
string	Upper	Takes in a single string and returns the same string only with all alphabetic characters in uppercase.
string	Lower	Takes in a single string and returns the same string only with all alphabetic characters in lowercase.
string	Initcap	Takes in a single string and returns the same string only with all alphabetic characters in lowercase except the first which is in uppercase.
vocabulary	translateValue	Requires the user to select either a vocabulary mapping file (`.vom`) or a URL to use as the basis of the conversion. Also may require a domain to be specified if the `.vom` file has more than one translation set in it. Takes in a single string and returns a converted string based on the “from” and “to” values and business rules defined in the vocabulary mapping file or the URL-based function.
vocabulary	translateInverseValue	Behaves the same way as the translateValue function only in reverse, matching the input value to the “to” side of the vocabulary mappings and returning the value from the “from” side.

Table 5.1 caAdapter Functions

Function Specifications

There are two function-related specifications. The first one describes the function groups and functions, and the inputs, outputs and implementation for each function. The second one describes the vocabulary mappings used by the vocabulary functions.

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:

- `<function>`
- `<group name>`
- `<function name>`
- `<inputs>`
- `<datapoint>`

- <outputs>

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

```
<?xml version="1.0"?>
<functions>
    <group name="constant" xmlPath="constant">
                <function name="constant" xmlPath="constant.constant">
                        <outputs>
                                <datapoint pos="0" name="constant"
datatype="string" xmlPath="constant.constant.outputs.0"/>
                        </outputs>
                </function>
    </group>
    <group name="date" xmlPath="date">
                <function name="changeFormat"
xmlPath="date.changeFormat">
                        <inputs>
                                <datapoint pos="0" name="fromFormat"
datatype="string" xmlPath="date.changeFormat.inputs.0"/>
                                <datapoint pos="1" name="dateIn"
datatype="string" xmlPath="date.changeFormat.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="dateOut"
datatype="string" xmlPath="date.changeFormat.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.DateFunction"
method="changeFormat"/>
                </function>
        <function name="countDays" xmlPath="date.countDays">
                        <inputs>
                                <datapoint pos="0" name="fromDate"
datatype="string" xmlPath="date.countDays.inputs.0"/>
                                <datapoint pos="1" name="toDate"
datatype="string" xmlPath="date.countDays.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="dayNumber"
datatype="int" xmlPath="date.countDays.outputs.0"/>
                        </outputs>
```

```
                        <implementation
classname="gov.nih.nci.caadapter.common.function.DateFunction"
method="countDays"/>
                </function>
    </group>
        <group name="math" xmlPath="math">
                <function name="Addition" xmlPath="math.Addition">
                        <inputs>
                                <datapoint pos="0" name="Value1"
datatype="double" xmlPath="math.Addition.inputs.0"/>
                                <datapoint pos="1" name="Value2"
datatype="double" xmlPath="math.Addition.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Sum"
datatype="double" xmlPath="math.Addition.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.MathFunction"
method="add"/>
                </function>
                <function name="Subtract" xmlPath="math.Subtract">
                        <inputs>
                                <datapoint pos="0" name="Value1"
datatype="double" xmlPath="math.Subtract.inputs.0"/>
                                <datapoint pos="1" name="Value2"
datatype="double" xmlPath="math.Subtract.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Difference"
datatype="double" xmlPath="math.Subtract.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.MathFunction"
method="subtract"/>
                </function>
                <function name="Multiply" xmlPath="math.Multiply">
                        <inputs>
                                <datapoint pos="0" name="Value1"
datatype="double" xmlPath="math.Multiply.inputs.0"/>
                                <datapoint pos="1" name="Value2"
datatype="double" xmlPath="math.Multiply.inputs.1"/>
                        </inputs>
```

```
                        <outputs>
                                <datapoint pos="0" name="Product"
datatype="double" xmlPath="math.Multiply.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.MathFunction"
method="multiply"/>
                </function>
                <function name="Divide" xmlPath="math.Divide">
                        <inputs>
                                <datapoint pos="0" name="Dividend"
datatype="double" xmlPath="math.Divide.inputs.0"/>
                                <datapoint pos="1" name="Divisor"
datatype="double" xmlPath="math.Divide.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Quotient"
datatype="double" xmlPath="math.Divide.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.MathFunction"
method="divide"/>
                </function>
                <function name="Round" xmlPath="math.Round">
                        <inputs>
                                <datapoint pos="0" name="Input"
datatype="double" xmlPath="math.Round.inputs.0"/>
                <datapoint pos="1" name="roundDigit" datatype="int"
xmlPath="math.Round.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Output"
datatype="double" xmlPath="math.Round.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.MathFunction"
method="round"/>
                </function>
        </group>
        <group name="string" xmlPath="string">
                <function name="Concatenate"
xmlPath="string.Concatenate">
                        <inputs>
```

```
                                <datapoint pos="0" name="String1"
datatype="string" xmlPath="string.Concatenate.inputs.0"/>

                                <datapoint pos="1" name="String2"
datatype="string" xmlPath="string.Concatenate.inputs.1"/>

                        </inputs>

                        <outputs>

                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Concatenate.outputs.0"/>

                        </outputs>

                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="concat"/>

                </function>

                <function name="Split" xmlPath="string.Split">

                        <inputs>

                                <datapoint pos="0" name="String1"
datatype="string" xmlPath="string.Split.inputs.0"/>

                                <datapoint pos="1" name="Pos"
datatype="int" xmlPath="string.Split.inputs.1"/>

                        </inputs>

                        <outputs>

                                <datapoint pos="0" name="Result1"
datatype="string" xmlPath="string.Split.outputs.0"/>

                                <datapoint pos="1" name="Result2"
datatype="string" xmlPath="string.Split.outputs.1"/>

                        </outputs>

                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="split"/>

                </function>

                <function name="Length" xmlPath="string.Length">

                        <inputs>

                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Length.inputs.0"/>

                        </inputs>

                        <outputs>

                                <datapoint pos="0" name="Length"
datatype="int" xmlPath="string.Length.outputs.0"/>

                        </outputs>

                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="length"/>

                </function>
```

```
                <function name="Substring" xmlPath="string.Substring">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Substring.inputs.0"/>
                                <datapoint pos="1" name="StartPos"
datatype="int" xmlPath="string.Substring.inputs.1"/>
                                <datapoint pos="2" name="EndPos"
datatype="int" xmlPath="string.Substring.inputs.2"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Substring.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="substring"/>
                </function>
                <!-- function name="Trim" xmlPath="string.Trim">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Trim.inputs.0"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Trim.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="trim"/>
                </function -->
                <function name="Replace" xmlPath="string.Replace">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Replace.inputs.0"/>
                                <datapoint pos="1" name="FromStr"
datatype="string" xmlPath="string.Replace.inputs.1"/>
                <datapoint pos="2" name="ToStr" datatype="string"
xmlPath="string.Replace.inputs.2"/>
            </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Replace.outputs.0"/>
                        </outputs>
```

```
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="replace"/>
                </function>
                <function name="Instring" xmlPath="string.Instring">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Instring.inputs.0"/>
                                <datapoint pos="1" name="Pattern"
datatype="string" xmlPath="string.Instring.inputs.1"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="int" xmlPath="string.Instring.outputs.1"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="instring"/>
                </function>
                <function name="Upper" xmlPath="string.Upper">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Upper.inputs.0"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Upper.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="upper"/>
                </function>
                <function name="Lower" xmlPath="string.Lower">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Lower.inputs.0"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Lower.outputs.0"/>
                        </outputs>
```

```
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="lower"/>
                </function>
                <function name="Initcap" xmlPath="string.Initcap">
                        <inputs>
                                <datapoint pos="0" name="String"
datatype="string" xmlPath="string.Initcap.inputs.0"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="Result"
datatype="string" xmlPath="string.Initcap.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.StringFunction"
method="initcap"/>
                </function>
        </group>
    <group name="vocabulary" xmlPath="vocabulary">
                <function name="translateValue"
xmlPath="vocabulary.translateValue">
                        <inputs>
                                <datapoint pos="0" name="dataIn"
datatype="string" xmlPath="vocabulary.translateValue.inputs.0"/>
                        </inputs>
                        <outputs>
                                <datapoint pos="0" name="dataOut"
datatype="string" xmlPath="vocabulary.translateValue.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.FunctionVocabularyMap
ping" method="translateValue"/>
                </function>
        <function name="translateInverseValue"
xmlPath="vocabulary.translateInverseValue">
                        <inputs>
                                <datapoint pos="0" name="dataIn"
datatype="string" xmlPath="vocabulary.translateInverseValue.inputs.0"/
>
                        </inputs>
                        <outputs>
```

```
                                    <datapoint pos="0" name="dataOut"
datatype="string"
xmlPath="vocabulary.translateInverseValue.outputs.0"/>
                        </outputs>
                        <implementation
classname="gov.nih.nci.caadapter.common.function.FunctionVocabularyMap
ping" method="inverseTranslateValue"/>
                </function>
    </group>
</functions>
```

Vocabulary Mapping Specification Overview

The vocabulary mapping specification is used as a guide for translating values from one vocabulary set to another. It includes one or more vocabulary domain names with associated translations (source and target values) and a mechanism for handling cases where the incoming value does not match any of the mapped values.

The vocabulary mapping specification uses the following types of nested elements:

- <VocabularyMapping>
- <comment>
- <domain>
- <translation>
- <source>
- <target>
- <elseCase>
- <inverseElseCase>

The `elsecase` and `inverseElseCase` elements can have several types which govern what happens when an incoming value doesn't match any of the maps. Some of the flavors also include a value that the mapping can define for that case. The types include the following:

Else Case Type	***Description***	***Includes a Value?***
keepValue	Returns the incoming value without any change	No
null	Returns a null	No
assignValue	Returns the value provided in the value attribute	Yes
makeAnError	Returns an error status to cause caAdapter to report a vocabulary mapping error	No

Table 5.2 Else Case Types

Following is an example of a vocabulary mapping specification file (using the designated file extension, `.vom`). See the `{home`

directory}\workingspace\examples\V2V3 Mapping Examples\ADT_A03_to_402003 file for a soft copy of this code and see the {home directory}\etc functions file for the vom.xsd file that governs the structure of the .vom file.

```
<?xml version="1.0" encoding="UTF-8"?>
<VocabularyMapping name="Test_Example01>
    <comment>
        This vom file was made for test instance of V2-V3
mapping
        which is between ADT^A03 and PRPA_MT402003
    </comment>
    <domain name="AdministrativeGender">
 <comment>
         Source:HL70001(Administrative Sex),
Target:2.16.840.1.113883.11.1(AdministrativeGender)

 </comment>
         <translation name="Male">
                <source value="M" remark="Male"/>
                <target value="M" remark="Male"/>
         </translation>
         <translation name="Female">
                <source value="F" remark="Female"/>
                <target value="F" remark="Female"/>
         </translation>
         <translation name="unknown1">
                <source value="U" remark="Unknown"/>
                <target value="UN" remark="Undifferentiated"/
>
         </translation>
         <translation name="unknown2">
                <source value="O" remark="Other"/>
                <target value="UN" remark="Undifferentiated"/
>
         </translation>
         <translation name="unknown3">
                <source value="A" remark="Ambiguous"/>
                <target value="UN" remark="Undifferentiated"/
>
         </translation>
         <translation name="unknown4">
                <source value="N" remark="Not applicable"/>
                <target value="UN" remark="Undifferentiated"/
>
         </translation>
         <elseCase type="keepValue"/>
         <inverseElseCase type="assignValue" value="UN"/>

    </domain>
```

```
        <domain name="DiseaseCodingSystemOID">
            <translation name="ICD-10">
                <source value="I10"/>
                <target value="2.16.840.1.113883.6.3"/>
            </translation>
            <translation name="ICD-9CM">
                <source value="I9C"/>
                <target value="2.16.840.1.113883.6.2"/>
            </translation>
            <translation name="SNOMED">
                <source value="SNM"/>
                <target value="2.16.840.1.113883.6.5"/>
            </translation>
            <elseCase type="keepValue"/>
            <inverseElseCase type="keepValue"/>
        </domain>
    </VocabularyMapping>
```

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.fls`, 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.

CHAPTER

6

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 API Modules* on page 78
- *caAdapter API Error Logs* on page 81

caAdapter Directory Structure

Depending on the type of distribution of caAdapter, the directory structure will vary. *Table 6.1* contains the directories under your `{home directory}`for the binary distribution.

Directory	***Contents***
conf	Component level configuration
docs	Javadocs and other useful information
lib	Java libraries and dependencies; and the MIF.zip file
schema	HL7 V3 Schema files
workingspace	Default directory where you can save project files. It contains log files and HL7 V3 XML instances. It also contains an `examples` directory with example data (see *caAdapter Example Data* on page 109).

Table 6.1 Directory Structure for caAdapter (Binary Distribution)

Table 6.2 contains the directories under your `{home directory}` for source distribution.

Directory	*Contents*
components	Different caAdapter components. Each component has its own build script, required libraries, and configurations. caAdapter has common, hl7, RDS, UI, and web service components.
conf	Component level configuration
docs	Javadocs and other useful information
lib	Java libraries and dependencies; and the MIF.zip file
etc	Important supplementary files
workingspace	Default directory where you can save project files. It contains log files and HL7 V3 XML instances. It also contains an examples directory with example data (see *caAdapter Example Data* on page 109).

Table 6.2 Directory Structure for caAdapter (Source Distribution)

caAdapter API Modules

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

- Metadata Loader
- Transformation Service
- HL7 V2 to HL7 V3 Transformation
- Vocabulary and MIF Schema Validation

The following sections provide a description of each.

Metadata 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 Metadata Loader drives how the internal HL7 message instance is built.

  Note: The Metadata Loader supports both format types: a java object of the serialized MIF file, a XML based file. The following example demonstrates how to use the Metadata Loader.

- Load Serialized MIF file from resource.zip - located at `lib` directory.

```
InputStream is = this.getClass()
.getResourceAsStream("/mif/" + mifFileName);
objectInputStream ois = new ObjectInputStream(is);
MIFClass mifClass = (MIFClass)ois.readObject();
ois.close();
is.close();
```

- Load Serialized MIF file from a XML file.

```
XmlToMIFImporter xmlToMIFImporter = new XmlToMIFImporter();
   MIFClass mifClass = xmlToMIFImporter
   .importMifFromXml(new File(filepath));
```

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.caadapter.hl7.transformation` 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.

```
TransformationService ts = new TransformationService
("data/Transformation/COCT_MT010000_MAP1-1.map",
   "data/Transformation/COCT_MT01000_Person.CSV");

List<XMLElement> xmlElements = ts.process();
   if (xmlElements==null)
   {
      //if failed in processing the source data
      //file,it returns error messages
      ValidatorResults rs=ts.getValidatorResults();
      String errorMsg= rs.getAllMessages().toString();
                                              }
   else  {
//return a  list of generated messages
for(XMLElement rootElement: xmlElements) {
               String hl7MessageXml= rootElement. toXML().
                        toString();
     }
```

HL7 V2 to HL7 V3 Transformation

The first step in mapping an HL7 V2 to an HL7 V3 is to create a CSV specification file, or `.scs` file, equivalent to the HL7 V2 message structure. The user can then use caAdapter to transform the HL7 V2 message into a CSV file based on the CSV specification file created in this step.

The second step is to map the elements of the CSV file to the appropriate HL7 V3 message. See *Updating the CSV Specification* on page 25 for more iinformation about this step.

Alternatively, you may use caAdapter's APIs to automatically transform the HL7 V2 data to create the corresponding CSV file (reference the second part of the first step above).

Following is sample code that shows how to accomplish this task.

```
V2Converter con = new
V2Converter(FileUtil.getV2DataDirPath());
con.convertV2ToCSV(hl7FileName, csvFileName, scsFileName);
if (!con.isCSVValid())
   List<String> errList = con.getValidationMessages();
```

This sample code must be caught by the HL7MessageTreeException.

Vocabulary and MIF Schema Validation

Vocabulary validation provides the ability to validate HL7 structural attributes against the HL7 published vocabulary. MIF schema validation validates an XML format HL7 message against a MIF schema file provided by the user (calling program).

The following example demonstrates how to invoke the two validation processes:

```
ValidatorResults validatorsToShow=new ValidatorResults();
String level=CaadapterUtil.readPrefParams(
Config.CAADAPTER_COMPONENT_HL7_TRANSFORMATION_VALIDATION_LEVEL);
//always process the structure validation ... level_0
validatorsToShow.addValidatorResults(xmlMsg.getValidatorRe-
sults());
if(level!=null&&! level.equals( CaAdapter-
Pref.VALIDATION_PERFORMANCE_LEVLE_0))
{
//add vocabulary validation ... level_1
validatorsToShow.addValidatorResults(xmlMsg.validate());
if(level.equals(CaAdapterPref.VALIDATION_PERFORMANCE_LEVLE_2))
{//add xsd validation
try {
String xsdFile= FileUtil.searchMessageTypeSchemaFileName(
xmlMsg.getMessageType(),"xsd");
 HL7V3MessageValidator h7v3Validator=new
HL7V3MessageValidator();
//add xsd validation ... level_2 validatorsToShow.addValidator-
Results(h7v3Validator.validate(xmlMsg.toXML().toString(), xsd-
File);
} catch (Exception e)
{
e.printStackTrace();
}
}
```

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}\workingspace` 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

7

caAdapter Web Services Transformation Module

This chapter contains information on using caAdapter's Web Services.

Topics in this chapter include:

- *Introduction* on this page
- *Setup Mapping Scenarios Through the Web Portal* on page 84
- *Programmatic Access to the caAdapter Web Services* on page 85

Introduction

A Web service is a software application identified by a URI, whose interface and bindings are capable of being identified, described and discovered by XML artifacts. The web service also supports direct interactions with other software applications using XML based messages via Internet-based protocols (by World Wide Web Consortium).

caAdapter's CSV to HL7 V3 Message Transformation Service API is a JAVA API and can only be directly integrated with a JAVA-based application. This web service provides a powerful mechanism to integrate caAdapter's CSV to HL7 V3 Transformation Service into a variety of systems that are developed under different platforms and software environment.

caAdapter Web Service Model includes the following two sub-components:

- Web Portal – provides basic mapping scenario management.
- Web Service API – provides CSV to HL7 V3 transformation service.

The Web Portal provide a mechanism to upload all the mapping files including the actual .map file, CSV specification file, and HL7 V3 specification file. Once uploaded, the files can be used by subsequent transformation services. This is typically a one-time effort.

Figure 7.1 illustrates the Web Service Module architecture.

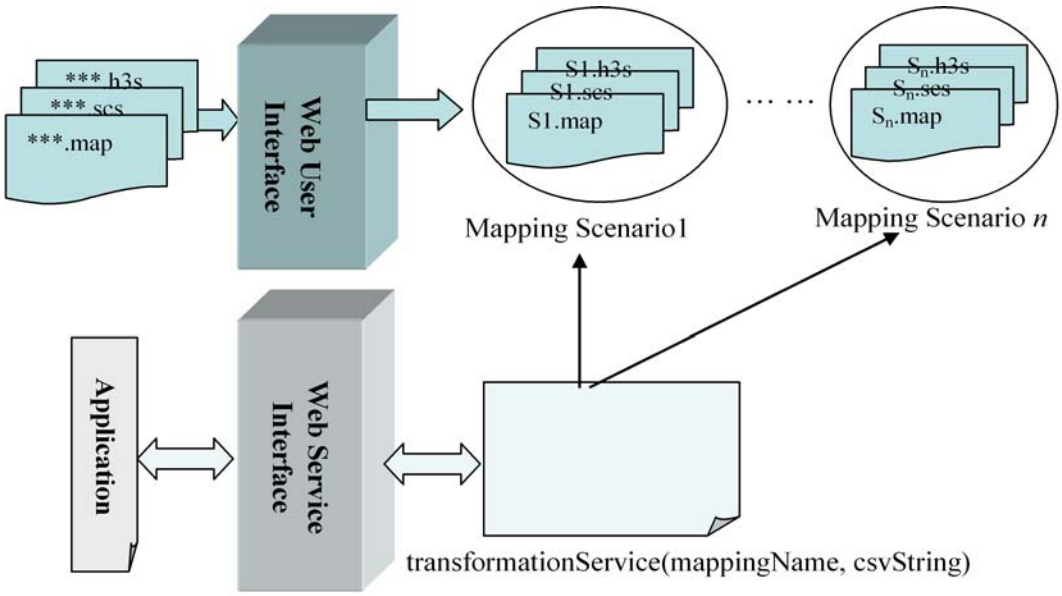


Figure 7.1 caAdapter Web Service Module Architecture

Setup Mapping Scenarios Through the Web Portal

This section contains the step-by-step instructions to upload mapping, CSV, and HL7 V3 specification.

1. Open an IE/Firefox browser and enter the following link:
   http://caadapter.nci.nih.gov

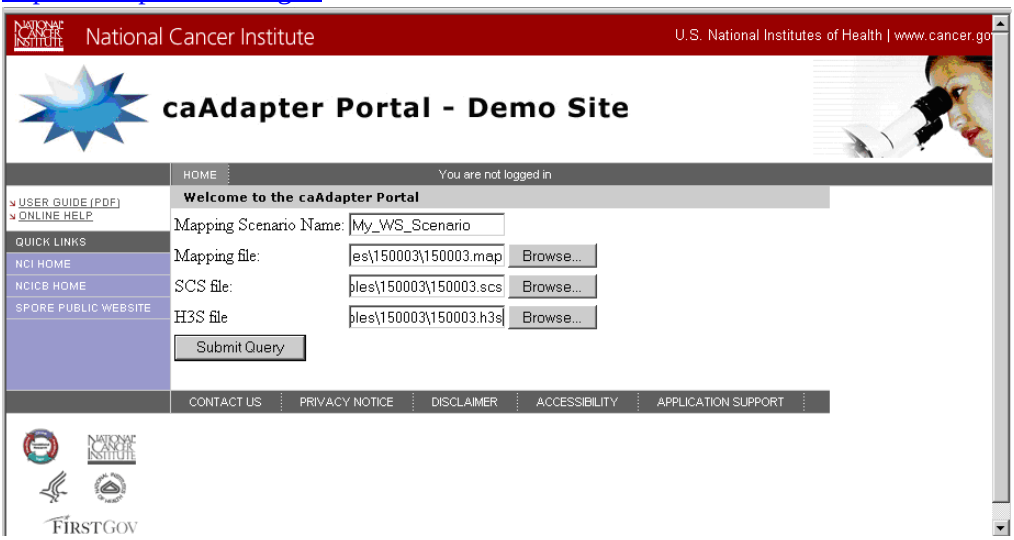


Figure 7.2 caAdapter Portal

2. In the "Mapping Scenario Name" field, specify the name for the set of mapping files you are going to upload, and use this name in the later web services clients.

3. In the “Mapping file” field, specify the name and path to the mapping file, usually with `.map` suffix.
4. In the “scs file” field, specify the name and path to the CSV specification file, usually with `.scs` suffix.
5. In the “H3S file” field, specify the name and path to the HL7 V3 metadata file, usually with `.h3s` suffix.

Once the mapping scenario is created successfully, a confirmation message appears.

Programmatic Access to the caAdapter Web Services

See the following sections for information about accessing the caAdapter Web Services:

- *Axis 1.x RPC Style Access to caAdapter Web Services* on page 85
- *Axis 1.x DII Style Access to caAdapter Web Services* on page 86
- *Axis 2.0 RPC Style Access to caAdapter Web Services* on page 87

Axis 1.x RPC Style Access to caAdapter Web Services

1. Download Axis 1.x (`axis-bin-1_4.zip`) from the following URL: http://www.apache.org/dyn/closer.cgi/ws/axis/1_4
2. Unzip the `axis-bin-1_4.zip` file.
3. Add the following files for the `axis-1_4/lib` directory to your classpath.
   - `axis.jar`
   - `axis-ant.jar`
   - `commons-discovery-0.2.jar`
   - `commons-logging-1.0.4.jar`
   - `jaxrpc.jar`
   - `log4j-1.2.8.jar`
   - `saaj.jar`
   - `wsdl4j-1.5.1.jar`
4. Run the following command to generate all the stubs:
   ```
   java org.apache.axis.wsdl.WSDL2Java http://caadapter.nci.nih.gov/caAdapterWS/ws/caAdapterTransformationService?wsdl
   ```
5. Use the following code to access the caAdapter Web Services.

```
import java.util.*;
import
gov.nih.nci.caadapter.caAdapterWS.ws.caAdapterTransformationS
ervice.*;
public class AxisRPCClient {
  public static void main(String[] args) {
    try {
      String csvString = "ORGS,RAD\nORGID,2.1 ... ...";
```

```
CaAdapterTransformationServiceService service
= new CaAdapterTransformationServiceServiceLocator();
CaAdapterTransformationService caAdapterService
= service.getcaAdapterTransformationService();
Object[] res  =
(Object[])caAdapterService.transformationService(
" My_WS_Scenario",csvString);
         for(int i=0;i<res.length;i++)
         System.out.println((String)res[i]);
     }catch(Exception e) {
        e.printStackTrace();
   }
 }
```

Axis 1.x DII Style Access to caAdapter Web Services

1. Download Axis 1.x (`axis-bin-1_4.zip`) from the following URL: http://www.apache.org/dyn/closer.cgi/ws/axis/1_4
2. Unzip `axis-bin-1_4.zip` file.
3. Add the following files for the `axis-1_4/lib directory` to your classpath.
   a. `axis.jar`
   b. `axis-ant.jar`
   c. `commons-discovery-0.2.jar`
   d. `commons-logging-1.0.4.jar`
   e. `jaxrpc.jar`
   f. `log4j-1.2.8.jar`
   g. `saaj.jar`
   h. `wsdl4j-1.5.1.jar`
4. Use the following code to access the caAdapter web services.

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import javax.xml.rpc.ParameterMode;
import javax.xml.namespace.QName;
import org.apache.axis.utils.Options;
import java.util.*;

public class AxisClient {

  public static void main(String[] args) {
    try {
      String endpointURL = " http://caadapter.nci.nih.gov/
caAdapterWS/ws/caAdapterTransformationService";
       String methodName = "transformationService";
```

```
      String csvString = "ORGS,RAD\nORGID,2.1 ... ...";
      Service service = new Service();
      Call call = (Call)service.createCall();
      call.setTargetEndpointAddress(new
java.net.URL(endpointURL));
      call.setOperationName(methodName);
      call.addParameter("parameter_name",
XMLType.XSD_STRING,
ParameterMode.IN );
      call.addParameter("csvstringname",
   XMLType.XSD_STRING,
 ParameterMode.IN );
      call.setReturnClass(java.util.ArrayList.class);
      ArrayList res = (ArrayList)call.invoke(
new Object[]{"My_WS_Scenario",csvString});
      System.out.println(res);
    }catch(Exception e) {
      e.printStackTrace();
    }
 }
```

In the above code, "My_WS_Scenario" is the "Mapping Scenario Name" you used in the caAdapter Web Service Management Portal. CSV String is the actual data that needs to be transformed. The result is an XML message of the result HL7 V3 messages.

Axis 2.0 RPC Style Access to caAdapter Web Services

1. Download Axis 2.0 (`axis2-1.1.zip`) from the following URL: http://ws.apache.org/axis2/
2. Unzip `axis2-1.1.zip`.
3. Add the following files for the `axis-1_4/lib` directory to your classpath.
   a. `axis.jar`
   b. `axis-ant.jar`
   c. `commons-discovery-0.2.jar`
   d. `commons-logging-1.0.4.jar`
   e. `jaxrpc.jar`
   f. `log4j-1.2.8.jar`
   g. `saaj.jar`
   h. `wsdl4j-1.5.1.jar`
4. Use the following code to access the caAdapter Web Services

```
package swe645;
import java.util.ArrayList;
import javax.xml.namespace.QName;
import org.apache.axis2.AxisFault;
import org.apache.axis2.addressing.EndpointReference;
```

```
import org.apache.axis2.client.Options;
import org.apache.axis2.rpc.client.RPCServiceClient;
import org.apache.axiom.om.impl.llom.OMTextImpl;
import org.apache.axiom.om.impl.llom.OMElementImpl

public class AxisClient {

   public static void main(String[] args1) throws AxisFault {
      String csvString = "ORGS,RAD\nORGID,2.1 ... ...";

  RPCServiceClient serviceClient = new RPCServiceClient();
        Options options = serviceClient.getOptions();
       EndpointReference targetEPR = new EndpointReference("
http://caadapter.nci.nih.gov/caAdapterWS/ws/
caAdapterTransformationService");
        options.setTo(targetEPR);
        // QName of the target method
        QName opAddEntry = new QName("caAdapter",
"transformationService");
        Object[] opAddEntryArgs = new Object[] {
"My_WS_Scenario",
csvString };
        Class[] returnTypes = new Class[] { ArrayList.class
};
        // Invoking the method
        Object[] res =
serviceClient.invokeBlocking(opAddEntry,
        opAddEntryArgs, returnTypes);

        ArrayList resultArrayList = (ArrayList) res[0];
for(int i=0;i< resultArrayList.size();i++) {
   OMElementImpl omE = (OMElementImpl)resultArrayList.get(i);
   OMTextImpl textOM = (OMTextImpl)omE.getFirstOMChild();
   System.out.println(textOM.getText());
 }
    }
}
```

CHAPTER

8

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 90
- *CSV Specification* on page 91
- *HL7 V3 Specifications* on page 92
- *HL7 V2 Specifications* on page 97
- *SDTM Data Files* on page 99
- *SDTM Metadata Files* on page 101
- *HL7 V3 Message* on page 102
- *CSV to HL7 V3 Map Specification* on page 103
- *Object to Database Map Specification* on page 104

caAdapter File Formats and Locations

caAdapter uses a variety of files in its APIs and mapping tool. *Table 8.1* contains the files and extensions used by caAdapter.

File Type	***Extension***
CSV Specification	`.scs`
HL7 V3 Specification	`.h3s` and `.xml`
HL7 V2 Message Structure	`.dat`
HL7 V3 DataTypeSpec	`.dat`

Table 8.1 File Extensions

File Type	Extension
HL7 V3 Segment Attribute Table	.dat
HL7 V3 Definition Table	.dat
Function Library Specification	.fls
SDTM Data File	.txt
SDTM Metadata File	.xml
Map Specification	.map
HL7 V3 Message	.xml

Table 8.1 File Extensions

Note: Manual editing of those files is not supported and is highly discouraged.

Caution: The map specification has an internal reference to the full path name of the source and target specification files. This must be accurate in order to process the conversion or to edit a map specification successfully. Though it is not recommended, the map specification file 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, or .xml), 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-related files. To document this structure, the CSV specification uses an XML format that has three main elements:

- <csvMetadata>
- <segment>
- <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; only the segment name and column number are referenced.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<csvMetadata xmlPath="csvMetaData" version="1.2">
   &lt;segment name="ORGS" xmlPath="ORGS" cardinality="1..1">
      &lt;segment name="ORGID" xmlPath="ORGS.ORGID"
cardinality="0..*">
         <field column="1" name="Root" datatype="String"
xmlPath="ORGS.ORGID.Root"/>
        <field column="2" name="Extension" datatype="String"
xmlPath="ORGS.ORGID.Extension"/>
      &lt;/segment>
      &lt;segment name="ORGNM" xmlPath="ORGS.ORGNM"
cardinality="0..*">
         <field column="1" name="Name" datatype="String"
xmlPath="ORGS.ORGNM.Name"/>
      &lt;/segment>
      &lt;segment name="ORGAD" xmlPath="ORGS.ORGAD"
cardinality="0..*">
         <field column="1" name="Street_1" datatype="String"
xmlPath="ORGS.ORGAD.Street_1"/>
         <field column="2" name="Street_2" datatype="String"
xmlPath="ORGS.ORGAD.Street_2"/>
         <field column="3" name="City" datatype="String"
xmlPath="ORGS.ORGAD.City"/>
         <field column="4" name="State" datatype="String"
xmlPath="ORGS.ORGAD.State"/>
         <field column="5" name="Zip_Code" datatype="String"
xmlPath="ORGS.ORGAD.Zip_Code"/>
      &lt;/segment>
      &lt;segment name="PERSNM" xmlPath="ORGS.PERSNM"
cardinality="0..*">
```

```
         <field column="1" name="First_Name"
datatype="String" xmlPath="ORGS.PERSNM.First_Name"/>
        <field column="2" name="Last_Name" datatype="String"
xmlPath="ORGS.PERSNM.Last_Name"/>
         <field column="3" name="Middle_Initial"
datatype="String" xmlPath="ORGS.PERSNM.Middle_Initial"/>
         <field column="4" name="Job_Code" datatype="String"
xmlPath="ORGS.PERSNM.Job_Code"/>
      &lt;/segment&gt;
      &lt;segment name="PERSID" xmlPath="ORGS.PERSID"
cardinality="0..*"&gt;
         <field column="1" name="Root" datatype="String"
xmlPath="ORGS.PERSID.Root"/>
        <field column="2" name="Extension" datatype="String"
xmlPath="ORGS.PERSID.Extension"/>
      &lt;/segment&gt;
      &lt;segment name="PERSAD" xmlPath="ORGS.PERSAD"
cardinality="0..*"&gt;
         <field column="1" name="Street_1" datatype="String"
xmlPath="ORGS.PERSAD.Street_1"/>
         <field column="2" name="Street_2" datatype="String"
xmlPath="ORGS.PERSAD.Street_2"/>
         <field column="3" name="City" datatype="String"
xmlPath="ORGS.PERSAD.City"/>
         <field column="4" name="State" datatype="String"
xmlPath="ORGS.PERSAD.State"/>
         <field column="5" name="Zip_Code" datatype="String"
xmlPath="ORGS.PERSAD.Zip_Code"/>
      &lt;/segment&gt;
      <field column="1" name="ORG_CODE" datatype="String"
xmlPath="ORGS.ORG_CODE"/>
   &lt;/segment&gt;
</csvMetadata>
```

HL7 V3 Specifications

The HL7 V3 specification, used to define the HL7 V3 metadata information, is based largely on the MIF for the target HL7 V3 message. An HL7 V3 specification may be saved either as a binary .h3s file or as an .xml file. The .h3s file is not readable. The .xml file uses four main types of nested elements:

- < class>
- < association>
- <attribute>
- < type>
- < dataField>

Following is part of an HL7 V3 specification file (150003.h3s) example. See the {home directory}\workingspace\examples\150003 for the entire file.

```
<class name="ContactParty" isEnabled="true" title="MIF Clone
Properties" referenceName="" sortKey="">

<packageLocation /

<attribute name="classCode" type="CS" defaultValue="CON"
isEnabled="true" title="MIF Attribute Properties"
mnemonic="CON" sortKey="1" minimumMultiplicity="1"
isStrutural="true"
parentXmlPath="Organization.contactParty00"
maximumMultiplicity="1" isMandatory="true" conformance="R"
dDefaultValueProperty="CON"
dDomainNameOidProperty="RoleClassContact
(2.16.840.1.113883.11.12205)" codingStrength="CNE"
multiplicityIndex="0" minimumSupportedLength="0"
domainName="RoleClassContact" />

<attribute name="id" type="II" isEnabled="true" title="MIF
Attribute Properties" sortKey="2" minimumMultiplicity="0"
parentXmlPath="Organization.contactParty00"
maximumMultiplicity="-1" multiplicityIndex="0"
minimumSupportedLength="0">

<type name="II" isEnabled="true" parents="ANY">

<dataField name="nullFlavor" type="NullFlavor" max="-2"
isValid="true" title="MIF Data Field Properties"
isSimple="true"
parentXmlPath="Organization.contactParty00.id00" min="-2"
isOptional="true" isAttribute="true" />

<dataField name="assigningAuthorityName" type="st" max="-2"
isValid="true" isEnabled="true" title="MIF Data Field
Properties" isSimple="true"
parentXmlPath="Organization.contactParty00.id00" min="-2"
isOptional="true" isAttribute="true" />
... ...
  </type>

</attribute>

<attribute name="addr" type="AD" isEnabled="true" title="MIF
Attribute Properties" sortKey="4" minimumMultiplicity="0"
parentXmlPath="Organization.contactParty00"
maximumMultiplicity="-1" multiplicityIndex="0"
minimumSupportedLength="0">

<type name="AD" isEnabled="true" parents="ANY">
```

```
<dataField name="direction" type="adxp.direction" max="-2"
isValid="true" title="MIF Data Field Properties" min="-2" />

<dataField name="city" type="adxp.city" max="-2"
isValid="true" isEnabled="true" title="MIF Data Field
Properties" isOptionChosen="true"
parentXmlPath="Organization.contactParty00.addr00" min="-2">

<type name="adxp.city" isEnabled="true" parents="ADXP">

<dataField name="reference" type="TEL" max="0" title="MIF
Data Field Properties" min="0" />

<dataField name="mediaType" type="cs" max="-2" isValid="true"
isEnabled="true" title="MIF Data Field Properties"
isSimple="true"
parentXmlPath="Organization.contactParty00.addr00.city"
min="-2" isAttribute="true" />

... ...

</type>

</dataField>

<dataField name="streetNameBase" type="adxp.streetNameBase"
max="-2" isValid="true" title="MIF Data Field Properties"
min="-2" />

<dataField name="precinct" type="adxp.precinct" max="-2"
isValid="true" title="MIF Data Field Properties" min="-2" />

<dataField name="unitType" type="adxp.unitType" max="-2"
isValid="true" title="MIF Data Field Properties" min="-2" />

... ...

</attribute>

<association name="contactPerson" isEnabled="true" title="MIF
Association Properties" sortKey="1" minimumMultiplicity="0"
isOptionChosen="true"
parentXmlPath="Organization.contactParty00"
maximumMultiplicity="1" multiplicityIndex="0">

<class name="Person" isEnabled="true" title="MIF Clone
Properties" referenceName="" sortKey="">

<packageLocation />
```

```
<attribute name="classCode" type="CS" isEnabled="true"
title="MIF Attribute Properties" mnemonic="PSN" sortKey="1"
minimumMultiplicity="1" isStrutural="true"
parentXmlPath="Organization.contactParty00.contactPerson"
maximumMultiplicity="1" isMandatory="true" conformance="R"
dDefaultValueProperty="PSN"
dDomainNameOidProperty="EntityClass
(2.16.840.1.113883.11.10882)" codingStrength="CNE"
multiplicityIndex="0" fixedValue="PSN"
minimumSupportedLength="0" domainName="EntityClass" />

<attribute name="determinerCode" type="CS" isEnabled="true"
title="MIF Attribute Properties" mnemonic="INSTANCE"
sortKey="2" minimumMultiplicity="1" isStrutural="true"
parentXmlPath="Organization.contactParty00.contactPerson"
maximumMultiplicity="1" isMandatory="true" conformance="R"
dDefaultValueProperty="INSTANCE"
dDomainNameOidProperty="EntityDeterminer
(2.16.840.1.113883.11.10878)" codingStrength="CNE"
multiplicityIndex="0" fixedValue="INSTANCE"
minimumSupportedLength="0" domainName="EntityDeterminer" />

<attribute name="name" type="EN" isEnabled="true" title="MIF
Attribute Properties" sortKey="3" minimumMultiplicity="1"
parentXmlPath="Organization.contactParty00.contactPerson"
maximumMultiplicity="-1" conformance="R"
multiplicityIndex="0" minimumSupportedLength="0">

<type name="EN" isEnabled="true" parents="ANY">

<dataField name="suffix" type="en.suffix" max="-2"
isValid="true" isEnabled="true" title="MIF Data Field
Properties" isOptionChosen="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00" min="-2">

<type name="en.suffix" isEnabled="true" parents="ENXP">

<dataField name="mediaType" type="cs" max="-2" isValid="true"
isEnabled="true" title="MIF Data Field Properties"
isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="-2" isAttribute="true" />

<dataField name="representation" type="BinaryDataEncoding"
max="-2" isValid="true" isEnabled="true" title="MIF Data
Field Properties" isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="-2" isAttribute="true" />
```

```
<dataField name="integrityCheckAlgorithm"
type="IntegrityCheckAlgorithm" max="-2" isEnabled="true"
title="MIF Data Field Properties" isProhibited="true"
isSimple="true" min="-2" isAttribute="true" />

<dataField name="language" type="cs" max="-2" isValid="true"
isEnabled="true" title="MIF Data Field Properties"
isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="-2" isOptional="true" isAttribute="true" />

<dataField name="thumbnail" type="ED" max="0" title="MIF Data
Field Properties" min="0" />

<dataField name="compression" type="CompressionAlgorithm"
max="-2" isEnabled="true" title="MIF Data Field Properties"
isProhibited="true" isSimple="true" min="-2"
isAttribute="true" />

<dataField name="nullFlavor" type="NullFlavor" max="-2"
isValid="true" isEnabled="true" title="MIF Data Field
Properties" isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="-2" isOptional="true" isAttribute="true" />

<dataField name="partType" type="EntityNamePartType" max="-2"
isValid="true" isEnabled="true" title="MIF Data Field
Properties" isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="-2" isAttribute="true" />

<dataField name="integrityCheck" type="bin" max="-2"
isEnabled="true" title="MIF Data Field Properties"
isProhibited="true" isSimple="true" min="-2"
isAttribute="true" />

<dataField name="reference" type="TEL" max="0" title="MIF
Data Field Properties" min="0" />

<dataField name="qualifier"
type="set_EntityNamePartQualifier" max="-2" isValid="true"
isEnabled="true" title="MIF Data Field Properties"
isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="-2" isOptional="true" isAttribute="true" />

<dataField name="inlineText" max="1" isValid="true"
isEnabled="true" title="MIF Data Field Properties"
isOptionChosen="true" isSimple="true"
```

```
parentXmlPath="Organization.contactParty00.contactPerson.name
00.suffix" min="1" />

</type>

</dataField>

<dataField name="nullFlavor" type="NullFlavor" max="-2"
isValid="true" isEnabled="true" title="MIF Data Field
Properties" isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00" min="-2" isOptional="true" isAttribute="true" />

<dataField name="inlineText" max="1" isValid="true"
isEnabled="true" title="MIF Data Field Properties"
isOptionChosen="true" isSimple="true"
parentXmlPath="Organization.contactParty00.contactPerson.name
00" min="1" />

<dataField name="delimiter" type="en.delimiter" max="-2"
isValid="true" title="MIF Data Field Properties"
parentXmlPath="Organization.contactParty00.contactPerson.name
00" min="-2" />

<dataField name="validTime" type="IVL_TS" max="-2"
isValid="true" title="MIF Data Field Properties"
parentXmlPath="Organization.contactParty00.contactPerson.name
00" min="0" />

</type>

</attribute>

</class>

</association>
... ...

</class>
```

HL7 V2 Specifications

The HL7 V2 message specification is described in four kinds of resource files, i.e. Message Structure, DataTypeSpec, DefinitionTable, SegmentAttributeTable. caAdapter

requires all four file collections to be able to parse HL7 V2 messages. *Figure 8.1* shows the directory structure where the resources files are stored.

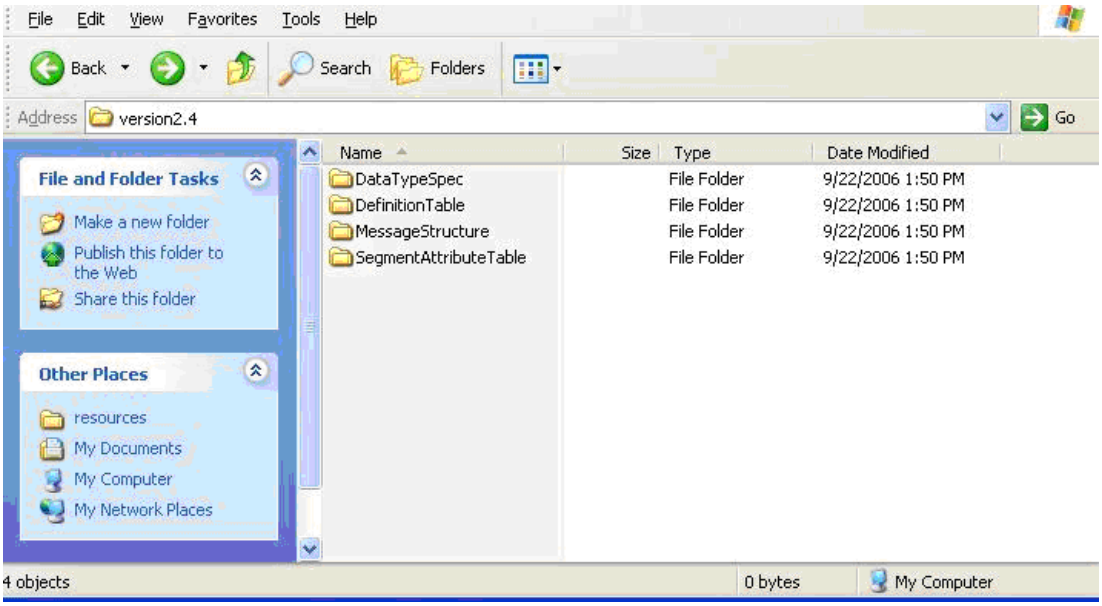


Figure 8.1 Resource Directory Structure

Message Structure

The Message Structure directory contains the information of the HL7 V2 message. The directory is organized by a collection of DAT files with file names corresponding to message type of the HL7 V2 message. 'ADT_A03' is a message type and the 'ADT_A03.DAT' is the data file. This DAT file represents the order of segments and represents the required and optional segments.

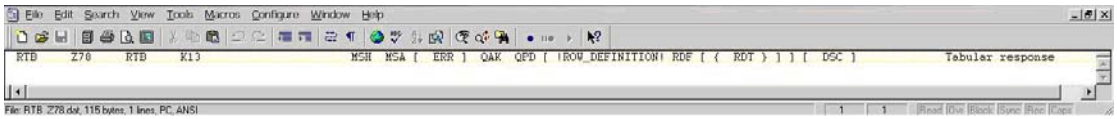


Figure 8.2 Contents of "RTB_Z78.DAT" message structure

DataTypeSpec

This directory contains DAT files with the file names corresponding to the data type. For example: AD is a datatype for representing the address object. The corresponding file in the directory has a physical file with the name "AD.DAT". The content of "AD.DAT" is shown below in *Figure 8.3*. The position, datatype (e.g. ST for String and ID for Identification), and description of each field are listed.

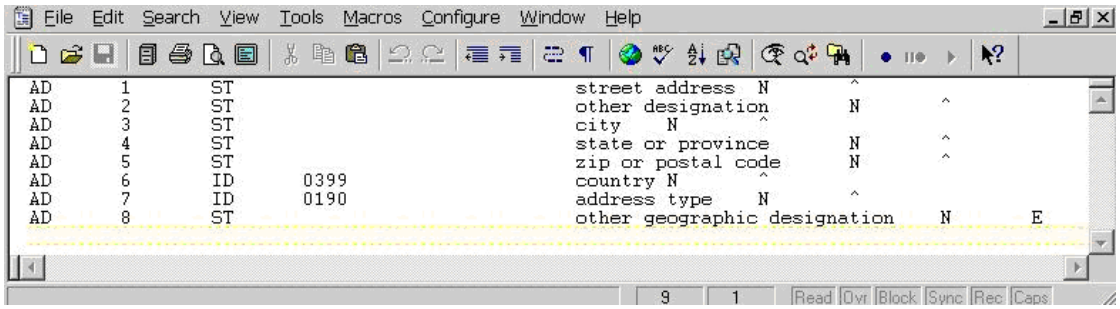


Figure 8.3 Contents of "AD.DAT"data type

Segment Attribute Table

The segment attribute table represents the structure of the Message Header (MSH) segment. It shows the fields, data types, positions, repeating fields, and index of each field for the MSH segment. *Figure 8.4* shows an example of a Segment Attribute Table.

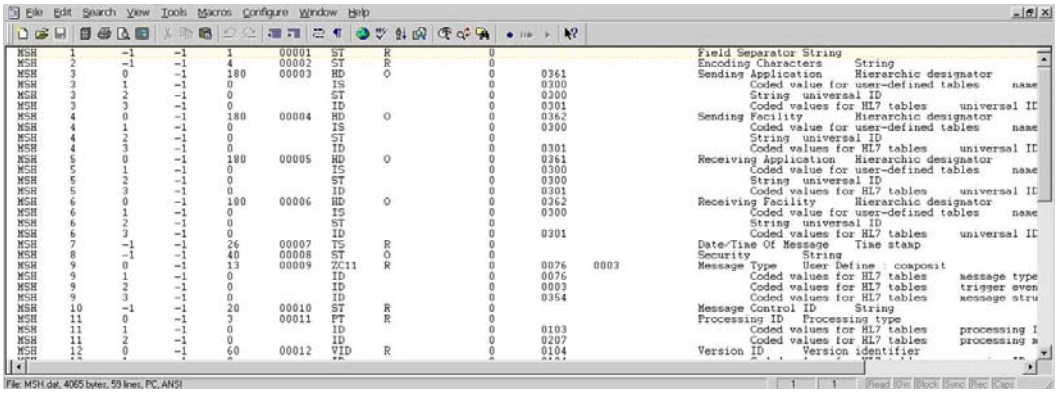
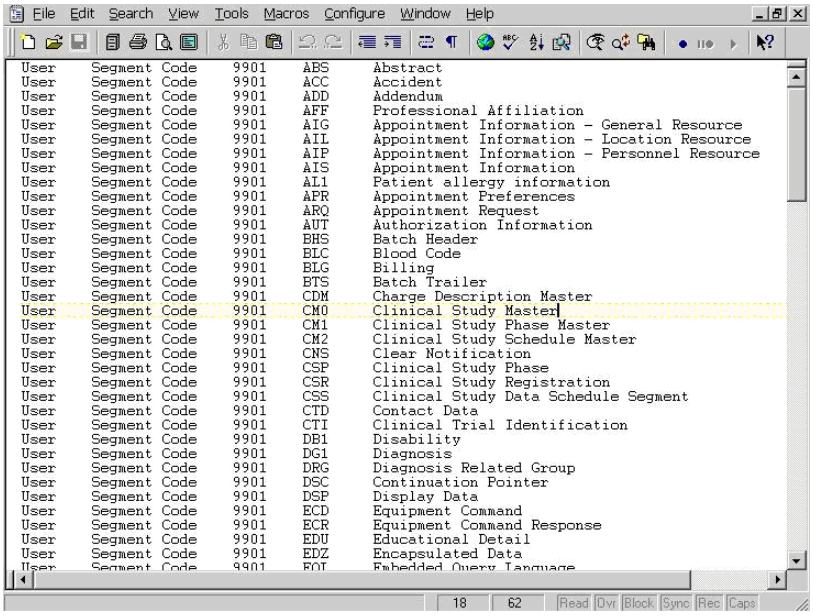


Figure 8.4 Contents of "MSH.DAT"segment information

Definition Table

The definition table stores the HL7 V2 vocabulary information for each segment in the message. For example, in the 9901.DAT file, shown in *Figure 8.5*, 'ABS' segment is represented as 'Abstract' and ''DB1' as 'Disability'.



User	Segment Code	9901	ABS	Abstract
User	Segment Code	9901	ACC	Accident
User	Segment Code	9901	ADD	Addendum
User	Segment Code	9901	AFF	Professional Affiliation
User	Segment Code	9901	AIG	Appointment Information - General Resource
User	Segment Code	9901	AIL	Appointment Information - Location Resource
User	Segment Code	9901	AIP	Appointment Information - Personnel Resource
User	Segment Code	9901	AIS	Appointment Information
User	Segment Code	9901	AL1	Patient allergy information
User	Segment Code	9901	APR	Appointment Preferences
User	Segment Code	9901	ARQ	Appointment Request
User	Segment Code	9901	AUT	Authorization Information
User	Segment Code	9901	BHS	Batch Header
User	Segment Code	9901	BLC	Blood Code
User	Segment Code	9901	BLG	Billing
User	Segment Code	9901	BTS	Batch Trailer
User	Segment Code	9901	CDM	Charge Description Master
User	Segment Code	9901	CM0	Clinical Study Master
User	Segment Code	9901	CM1	Clinical Study Phase Master
User	Segment Code	9901	CM2	Clinical Study Schedule Master
User	Segment Code	9901	CNS	Clear Notification
User	Segment Code	9901	CSP	Clinical Study Phase
User	Segment Code	9901	CSR	Clinical Study Registration
User	Segment Code	9901	CSS	Clinical Study Data Schedule Segment
User	Segment Code	9901	CTD	Contact Data
User	Segment Code	9901	CTI	Clinical Trial Identification
User	Segment Code	9901	DB1	Disability
User	Segment Code	9901	DG1	Diagnosis
User	Segment Code	9901	DRG	Diagnosis Related Group
User	Segment Code	9901	DSC	Continuation Pointer
User	Segment Code	9901	DSP	Display Data
User	Segment Code	9901	ECD	Equipment Command
User	Segment Code	9901	ECR	Equipment Command Response
User	Segment Code	9901	EDU	Educational Detail
User	Segment Code	9901	EDZ	Encapsulated Data

Figure 8.5 Contents of "9901.DAT"segment information

SDTM Data Files

A Study Data Tabulation Module (SDTM) text file consists of the mapped data elements from the CSV file. The file has a .txt extension. This text file is created by the SDTM transformation service. For each mapped source field in a segment in the scs file, a record will be created keeping the parent-child relationship intact. This is accomplished

by prefixing the path information to each row in the CSV file. The transformation service engine will fetch values for all the fields in the specified path.

For example, the converted CSV file is transformed by the transformation service as shown below.

"\SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\INVESTIGATIVESUBJECT_511\SUPPLY_5112\AUTHOR_51123\ASGNDENTT090000_511231\ASSIGNEDPERSON_5112311^Doeighty,Conrard,D."

The field name is 'ASSIGNEDPERSON_5112311' and the value is 'Doeighty,Conrard,D.' but the parent segment for this particular record are as listed below:

1. \SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\INVESTIGATIVESUBJECT_511\SUPPLY_5112\AUTHOR_51123\ASGNDENTT090000_511231\ASSIGNEDPERSON_5112311^Doeighty,Conrard,D.
2. \SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\INVESTIGATIVESUBJECT_511\SUPPLY_5112\AUTHOR_51123\ASGNDENTT090000_511231
3. \SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\INVESTIGATIVESUBJECT_511\SUPPLY_5112\AUTHOR_51123\
4. \SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\INVESTIGATIVESUBJECT_511\SUPPLY_5112\
5. \SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\INVESTIGATIVESUBJECT_511\
6. \SourceTree\INVESTEVN\TRIGGER_5\REACTION_51\
7. \SourceTree\INVESTEVN\TRIGGER_5\
8. \SourceTree\INVESTEVN\
9. \SourceTree\

The transformation service checks for mapped fields in any of the parent segments. If a mapping segment exists, the corresponding value from the CSV file will be set in the same record in the resulting SDTM .txt file.

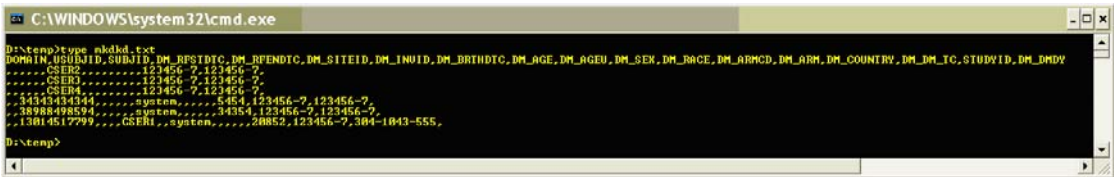


Figure 8.6 Contents of SDTM Text File

SDTM Metadata Files

SDTM metadata file, also called Case Report Tabulation Data Definition Specification (`define.xml`), describes the data exchange structure for the different domains. Sample `define.xml` can be found at CDISC website:http://www.cdisc.org/models/def/v1.0/index.html. The following is a sample section of the `define.xml` file downloaded form CDISC.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
************************************************************
**** -->
<!-- File: defineexample1.xml
-->
<!-- Date: 28-01-2005
-->
<!-- Author: Clinical Data Interchange Standards Consortium
(CDISC)             -->
<!-- Description: This is an example define.xml document
which … the Case  -->
<!--   Report Tabulation Data Definition Specification
Version 1.0.0              -->
<!--
************************************************************
**** -->
<ODM
  xmlns="http://www.cdisc.org/ns/odm/v1.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:def="http://www.cdisc.org/ns/def/v1.0"
  xsi:schemaLocation="http://www.cdisc.org/ns/odm/v1.2
define1-0-0.xsd"
  FileOID="Study1234"
  ODMVersion="1.2"
  FileType="Snapshot"
  CreationDateTime="2004-07-28T12:34:13-06:00">
<Study OID="1234">
  <GlobalVariables>
    <StudyName>1234</StudyName>
    <StudyDescription>1234 Data Definition</StudyDescription>
    <ProtocolName>1234</ProtocolName>
  </GlobalVariables>
  <MetaDataVersion OID="CDISC.SDTM.3.1.0"
```

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>
```

CSV to HL7 V3 Map Specification

A CSV to HL7 V3 map specification describes the relationship between components via links and/or views. It has the following main elements:

1. <components>
2. <links>
3. <source>
4. <target>
5. <linkpointer>
6. <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 (`150003.map`) example. See the `{home directory}\workingspace\examples\150003` for the entire file.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="1.2">
   <components>
      <component kind="scs" location="150003.scs" type="source"/>
      <component kind="h3s" location="150003.h3s" type="target"/>
   </components>
   <links>
      <link>
         <source>
            <linkpointer kind="scs" xmlPath="ORGS.ORG_CODE"/>
         </source>
         <target>
            <linkpointer kind="h3s" xmlPath="Organiza-
tion.contactParty00.contactPerson.name00.inlineText"/>
         </target>
      </link>
      <link>
```

```
            <source>
               <linkpointer kind="scs" xmlPath="ORGS.ORGID.Root"/>
            </source>
            <target>
               <linkpointer kind="h3s" xmlPath="Organiza-
tion.contactParty00.id00.extension"/>
            </target>
         </link>
         <link>
            <source>
               <linkpointer kind="scs" xmlPath="ORGS.ORGID"/>
            </source>
            <target>
               <linkpointer kind="h3s" xmlPath="Organiza-
tion.contactParty00"/>
            </target>
         </link>
      </links>
      <views>
         <view component-id="source.scs.0" height="0" width="0"
x="0" y="0"/>
         <view component-id="target.h3s.0" height="0" width="0"
x="0" y="0"/>
      </views>
   </mapping>
```

Object to Database Map Specification

An object to database map specification describes the relationship between objects/attributes and database tables/columns via links. It has the following main elements:

```
<components>
<links>
```

A component is a reference to an XMI file that exists in the system prior to the mapping. The location attribute of the component specifies the exact name and location of that XMI file.

A link describes a mapping for an object, an attribute or an association. A link element has a type and datatype attribute.

If the type value is “*dependency* “, the <source> sub-element describes an object to be mapped, and the <target> sub-element describes the target table that will be mapped to.

```
<link type="dependency" parent="null">
  <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene</source>
  <target>Logical View.Data Model.GENE</target>
</link>
```

If the type value is “*attribute* “, the <source> sub-element describes an attribute to be mapped, and the <target> sub-element describes the target table column that will be mapped to.

```
<link type="attribute" datatype="String">
  <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.locusLinkSummary</source>
  <target>Logical View.Data Model.GENE.LOCUS_LINK_SUMMARY</
target>
</link>
```

If type value is “*association* “, this section describes the one-to-one or one-to-many association, the <source> sub-element describes an association attribute to be mapped, and the <target> sub-element describes the target foreign key column that will be mapped to.

```
<link type="association">
  <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.chromosome</source>
  <target>Logical View.Data Model.GENE.CHROMOSOME_ID</target>
</link>
```

If the type value is “*manytomany* “, the section describes the many-to-many association. The <source> sub-element describes an association attribute to be mapped, and the <target> sub-element describes the target foreign key column that will be mapped to.

```
<link type="manytomany">
  <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Sequence.geneCollection</source>
  <target>Logical View.Data Model.GENE_SEQUENCE.GENE_ID</tar-
get>
</link>
```

Following is a part of a map specification file (`example.map`) example. See the `{home`

directory}\workingspace\examples\Object-2-DB-Example for the entire file.

```
<?xml version="1.0" encoding="UTF-8"?>
<mappings type="sdkintegration">
  <components>
    <component location="D:\projects\hl7sdk-new\working-
space\sample.xmi" />
    <component location="D:\projects\hl7sdk-new\working-
space\sample.xmi" />
  </components>
  <link type="dependency" parent="null">
    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene</source>
    <target>Logical View.Data Model.GENE</target>
  </link>
  <link type="dependency" parent="null">
    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Taxon</source>
    <target>Logical View.Data Model.TAXON</target>
  </link>
      … …
  <link type="attribute" datatype="String">
    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.locusLinkSummary</source>
    <target>Logical View.Data Model.GENE.LOCUS_LINK_SUMMARY</
target>
  </link>
  <link type="attribute" datatype="String">
    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.OMIMID</source>
    <target>Logical View.Data Model.GENE.OMIM_ID</target>
  </link>
      ……
  <link type="association">
```

```
    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.taxon</source>

    <target>Logical View.Data Model.GENE.TAXON_ID</target>

  </link>

  <link type="association">

    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.chromosome</source>

    <target>Logical View.Data Model.GENE.CHROMOSOME_ID</target>

  </link>

  <link type="manytomany">

    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Sequence.geneCollection</source>

    <target>Logical View.Data Model.GENE_SEQUENCE.GENE_ID</tar-
get>

  </link>

  <link type="manytomany">

    <source>Logical View.Logical
Model.gov.nih.nci.cabio.domain.Gene.libraryCollection</source>

    <target>Logical View.Data Model.LIBRARY_GENE.LIBRARY_ID</
target>

  </link>

… …

</mappings>
```

This version of caAdapter, although still supports the map file, it no longer requires it. All mapping specifications are now stored in the XMI file as shown in *Figure 8.7*.

```
<UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="VARCHAR" />
  <UML:TaggedValue tag="length" value="0" />
  <UML:TaggedValue tag="ordered" value="0" />
  <UML:TaggedValue tag="precision" value="0" />
  <UML:TaggedValue tag="scale" value="0" />
  <UML:TaggedValue tag="collection" value="false" />
  <UML:TaggedValue tag="position" value="3" />
  <UML:TaggedValue tag="lowerBound" value="1" />
  <UML:TaggedValue tag="upperBound" value="1" />
  <UML:TaggedValue tag="ea_guid" value="{7F4C5830-8609-4971-8327-F0CFE011B170}" />
  <UML:TaggedValue tag="ea_localid" value="151" />
  <UML:TaggedValue tag="precision" value="0" xmi.id="EAID_15C86B99_08E9_4476_9DF8_6A1218C182ED" />
  <UML:TaggedValue tag="length" value="100" xmi.id="EAID_1D9EA187_76F2_46b5_8A0F_4B5838AB8359" />
  <UML:TaggedValue tag="ordered" value="0" xmi.id="EAID_22D57E72_9D42_41bf_9A39_B37B981C76FE" />
  <UML:TaggedValue tag="duplicates" value="0" xmi.id="EAID_34C16746_5399_4e81_A3FA_F582556FBEDE" />
  <UML:TaggedValue tag="ea_guid" value="{6400A5A5-9C5A-4c14-8603-75BB8AA16A02}" xmi.id="EAID_401F6583_
  <UML:TaggedValue tag="position" value="3" xmi.id="EAID_65FB4B3F_D1CB_446e_A46C_961C14932898" />
  <UML:TaggedValue tag="collection" value="false" xmi.id="EAID_895AC9A4_AAAE_4880_A56B_F9BF44678FA2" /
  <UML:TaggedValue tag="lowerBound" value="1" xmi.id="EAID_9447309B_5CC8_4b75_AA46_51F3679CDE9F" />
  <UML:TaggedValue tag="upperBound" value="1" xmi.id="EAID_BAC8D21C_349B_496a_B8CD_A394E2E265AC" />
  <UML:TaggedValue tag="scale" value="0" xmi.id="EAID_D78F23B1_8E8B_4064_B465_D28512F0F370" />
  <UML:TaggedValue tag="type" value="VARCHAR" xmi.id="EAID_F49E1284_DA5B_43ef_95E3_D1499440A425" />
  <UML:TaggedValue tag="stereotype" value="column" xmi.id="EAID_F6A61E25_C78A_4244_8BF0_7D09438AD381"
  <UML:TaggedValue tag="mapped-attributes" value="gov.nih.nci.cabio.domain.Clone.version" xmi.id="EAID
</UML:ModelElement.taggedValue>
```

Figure 8.7 Mapping Specifications in the XMI File

APPENDIX A

caADAPTER EXAMPLE DATA

Example data are 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. Example data are located at the `{home directory}\workingspace\examples directoryexamples` directory (for example, `C:\caadapter\workingspace\examples`). The example data directory structure is shown in *Figure A.1*.

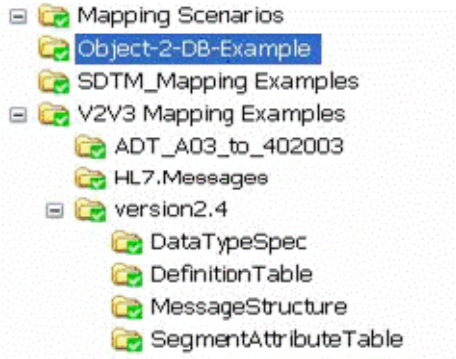


Figure A.1 Example data directory structure

The `examples` directory contains small (`090102`), medium (`040002`) and large (`040001040011`) sample HL7 V3 message files. The large HL7 V3 message example is an ICSR message. The other directories contain a subset of this data. For more information on mapping scenarios see the caAdapter Mapping Rules documentation.

The `V2V3 Mapping Examples` directory contains `ADT_A03_to_402003`, `HL7.Messages`, `version2.4`. The `version2.4` contains `DataTypeSpec`, `DefinitionTable`, `MessageStructure` and `SegmentAttributeTable`.

APPENDIX B
REFERENCES

Articles

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

caBIG Material

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

caCORE Material

- NCICB: http://ncicb.nci.nih.gov
- caCORE: http://ncicb.nci.nih.gov/NCICB/infrastructure
- caBIO: https://wiki.nci.nih.gov/display/ICR/caBIO;jsessionid=05C3565FB5DDB15ED00F27EF057D1A15
- caDSR: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/cadsr

HL7 Concepts and Material

- HL7: http://www.hl7.org/
- HL7 Tutorial: http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter/HL7_Tutorial
- caAdapter: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caadapter/

- HL7 Reference Information Model: https://www.hl7.org/library/data-model/RIM/C30202/rim.htm
- HL7 Vocabulary Domains: http://www.hl7.org/library/data-model/RIM/C30123/vocabulary.htm
- HL7 Version 3 Standard: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm
- UCUM: http://aurora.regenstrief.org/UCUM/ucum.html

Software Products

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

caAdapter Glossary

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

Term	***Definition***
CDMS	Clinical Data Management System.
CSV	Comma Separated Value
DMIM	Domain Message Information Model. 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).
DTD	Document Type Definition
EA	Enterprise Architect. UML Modeling Tool
HL7	Health Level 7 (http://www.hl7.org/) is one of several American National Standards Institute (ANSI)-accredited Standards Developing Organizations (SDOs) operating in the healthcare arena.
MIF	Model Interchange Format. An XML representation of the information contained in an HL7 specification, and is the format that all HL7 V3 specification authoring and manipulation tools will be expected to use.
MT	Message Type. The specification of an individual message in a specific implementation technology.
OID	HL7 V3 artifacts used to identify coding schemes and identifier namespaces.
NCI CBIIT	National Cancer Institute Center for Biomedical Informatics and Information Technology
RIM	Reference Information Model. The foundational Unified Modeling Language (UML) class diagram representing the universe of all healthcare data that may be exchanged between systems.
RMIM	Refined Message Information Model. 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.

Term	***Definition***
SDK	caCORE Software Development Kit or caCORE SDK, a data management framework designed for researchers who need to be able to navigate through a large number of data sources. caCORE SDK is NCICB's platform for data management and semantic integration, built using formal techniques from the software engineering and computer science communities.
SDTM	Study Data Tabulation Model. A set of standards developed by the Clinical Data Interchange Standards Consortium (CDISC).
TDMS	Translational Data Mapping Service
UCUM	Unified Code for Units of Measure
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

INDEX

A

Abstract data types
  updating 35
Add Clone option 33
Add Function option 45
Adding
  fields in CSV 25
  functions 75, 102
  functions to function library 75, 102
  function to map specification 45
  multiple attributes on HL7 v3 specification 34
  multiple clones on HL7 v3 specification 34
  segments in CSV 25
Add Multiple Clone option 34
Adverse event
  reporting 16
ANY label 35
API, caAdapter 78
Attribute, HL7 v3 specification 29

B

Building
  object graph 16
Business rules
  HL7 v3 message 48
  HL7 v3 specification 28
  map specification 39

C

caAdapter 113
  APIs 16
  overview 5
caadapter.log file 81
caBIG solution 5
Cardinality 34
CDMS, operational scenario 17
changeFormat date function 46
Changing logging properties 81
Choice
  boxes 35
Clone
  adding 33
  Attribute Object Properties panel 43, 59
  HL7 v3 specification 29
Clone List dialog box 33, 35
codeSystem data type 33
Component, defined 103, 104
Components of caAdapter 5
Constant function 46
Converting data file into HL7 v3 message 49, 53
core.fls file 75
Creating
  HL7 v3 message 49, 53
  HL7 v3 Specification 30
  mapping link 41
  map specification 40
CSV data file format 90
CSV Field Properties 42, 58
CSV specification
  business rules 22
  file example 91
  format 91
  updating 25

D

Data type
  element 29
  field 35
Date function, changeFormat 46
Default values
  defining 32
Defining
  default data 32
  mappings 20
  object identifiers 32
  units of measure 31
Delete button 26
Deleting
  fields in CSV 26
  map lines 42, 58
  segments in CSV 26

Dragging-and-dropping elements in CSV 26
DTD 113

E

Edit Constant option 46
Editing
  constant function 46
  field name 26
  fields in CSV 26
  segment name 25
  segments in CSV 26
Element
  types of HL7 v3 specification 29
EVS, validation 16
Example
  CSV specification file 91
  data 77, 78
  function specification file 66, 73
  HL7 v3 message file 102
  HL7 v3 specification 92
  map specification file 103, 105
  OIDs 33
examples directory 77, 78, 92, 103, 106
Excel spreadsheet 48
Extensions, file descriptions 89

F

FDA, operational scenario 17
Field properties 26
File
  New CSV Specification 23
  New HL7 v3 Message 50, 53
  New HL7 v3 Specification 30
  New Map Specification 40
  Open CSV Specification 24
  Open HL7 v3 Specification 31
  Open Map Specification 41
  Save 27, 38, 47, 52, 54
  Save As 27, 38, 47, 52, 54
  Validate 27, 38, 47
File extensions 89
File types 89
Format
  CSV data file 90
  CSV specification 91
  function specification 65
  HL7 v3 message 102
  map specification 103, 104
  of files 89
Function
  component, defined 103
  group properties panel 44, 60
  library 75
  panel, defined 45
  properties panel 44, 45, 60
  requirements 75
  specification, example file 66, 73
  specification format 65, 73

G

Generating
  CSV Report 28
  CSV specification 20
  HL7 specification 20
  HL7 v3 messages 49, 53
  Map Report 48
Glossary 113
gov.nih.nci.hl7.map package 79

H

HL7
  assigned OIDs 32
  choice boxes 35, 37
HL7 v3 message
  business rules 48
  creating 49, 53
  defined 48
  dialog box 50, 53
  Example file 102
  format 102
  overview 49, 52
  tab features 51
HL7 v3 specification
  attribute properties panel 43, 59
  data type field properties panel 43, 59
  dialog box 30
  element options 30
  example file 92
  format 97
  tab overview 29
  validating 37

I

ICSR
  operational scenario 17
inlineText data type field 31

J

JdomMessageTypeLoader 78

L

Link
  properties panel 43, 59
Log files 81
logging.properties file 81

M

Mandatory
  values 32
MapGenerateResult class 79
Mapping
  line 42, 58
Mapping tool
  basic steps 20
  interface 11
Map specification
  business rules 39
  example file 103, 105
  format 103, 104
  internal reference 90
  opening 41
  status 48
  tab overview 39
  updating 41
  validating 47
Menu bar 11
Message types, supported 31
Meta Data Loader 78
MIF
  file 78
  format 78
  HL7 file 17
Move Down button 25
Move Up button 25
Moving a segment in CSV 26
Multiples in HL7 v3 specification 34

N

NCICB 5
NCI CBIIT 113
Next button 52, 53

O

OID
  defining 32
  registry page 32
Open CSV specification dialog 24
Open Data File dialog box 50, 53
Open HL7 v3 Specification File dialog box 31
Opening
  HL7 v3 specification 31
  map specification 41
  new file 14
Open Map Specification dialog box 50, 53
Open Source File dialog box 40
Open Target File dialog box 40
Optional associations 33

P

Parsing
  message 16
Previous button 52, 53
Properties
  panel 42, 58

Q

QTY label 35

R

Regenerate button 52, 54
Registering custom function libraries 75
Remove Clone option 34
Remove Multiple Attribute option 34
Remove Multiple Clone option 34
Removing, multiple attributes from HL7 v3 specification 34
Removing, multiple clones from HL7 v3 specification 34
Report
  CSV example 28
  generate map report 48
  generate report 28
  Map specification 48
Reset button 26
Resizing panels 11

S

Saving
  HL7 v3 Message 52, 54
  HL7 v3 Specification 38
  map specification 47
Scroll bars 11
Segment
  options 25
  properties 25
Select Choice option 35
Selected Choice for label 35
SimpleDateFormat class 46

T

Tab
  CSV specification 22
  HL7 v3 message 49, 52
  HL7 v3 specification 29
  map specification 39
  open 11
  types of 14
TDMS 114
Tool bar 11

Transformation Service 79
TransformationService class 79
Transforming, into RIM object graph 20

U

UCUM, units of measure 31
Units of measure properties 31
Updating
  map specification 41
User-defined default value 32, 33
Using
  date function 46

V

Validating
  CSV 27
  CSV data against specialization 27
  CSV specification 27
  HL7 structural attributes 80
  HL7 v3 specification 37
  vocabulary using EVS 16
Validation Messages dialog box 47
Validation Messages panel 38
Vocabulary validation 80

W

Windows layout, mapping tool 11