USER'S GUIDE

Owner: caAdapter Development Team

COMMON MAPPING AND TRANSFORMATION SERVICE

Last Updated January 06, 2012

National Cancer Institute
Center for Biomedical Informatics and Information Technology
2115 East Jefferson Street
Rockville, MD 20852

Center for Biomedical Informatics and Information Technology

Table of Contents

caBIG® Open Source Software License

caAdapter 4.0, caAdapter 4.1 MMS, caAdapter 4.2 GME, caAdapter 4.3 HL7 MTS, caAdapter CMTS

 caAdapter was created with NCI funding and is part of the caBIG® initiative. The software subject to this notice and license includes both human readable source code form and machine readable, binary, object code form (the “caBIG® Software”).

This caBIG® Software License (the “License”) is between caBIG® Participant and You. “You (or “Your”) shall mean a person or an entity, and all other entities that control, are controlled by, or are under common control with the entity. “Control” for purposes of this definition means (i) the direct or indirect power to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

License. Provided that You agree to the conditions described below, caBIG® Participant grants You a non-exclusive, worldwide, perpetual, fully-paid-up, no-charge, irrevocable, transferable and royalty-free right and license in its rights in the caBIG™ Software, including any copyright or patent rights therein, to (i) use, install, disclose, access, operate, execute, reproduce, copy, modify, translate, market, publicly display, publicly perform, and prepare derivative works of the caBIG® Software in any manner and for any purpose, and to have or permit others to do so; (ii) make, have made, use, practice, sell, and offer for sale, import, and/or otherwise dispose of caBIG® Software (or portions thereof); (iii) distribute and have distributed to and by third parties the caBIG® Software and any modifications and derivative works thereof; and (iv) sublicense the foregoing rights set out in (i), (ii) and (iii) to third parties, including the right to license such rights to further third parties. For sake of clarity, and not by way of limitation, caBIG® Participant shall have no right of accounting or right of payment from You or Your sublicensees for the rights granted under this License. This License is granted at no charge to You. Your downloading, copying, modifying, displaying, distributing or use of caBIG® Software constitutes acceptance of all of the terms and conditions of this Agreement. If you do not agree to such terms and conditions, you have no right to download, copy, modify, display, distribute or use the caBIG™ Software.

1. Your redistributions of the source code for the caBIG® Software must retain the above copyright notice, this list of conditions and the disclaimer and limitation of liability of Article 6 below. Your redistributions in object code form must reproduce the above copyright notice, this list of conditions and the disclaimer of Article 6 in the documentation and/or other materials provided with the distribution, if any.

2. Your end-user documentation included with the redistribution, if any, must include the following acknowledgment: “This product includes software

developed by Science Applications International Corporation (SAIC).” If You do not include such end-user documentation, You shall include this acknowledgment in the caBIG® Software itself, wherever such third-party acknowledgments normally appear.

3. You may not use the names “Science Applications International Corporation”, "SAIC", “The National Cancer Institute”, “NCI”, “Cancer Bioinformatics Grid” or “caBIG™” to endorse or promote products derived from this caBIG® Software. This License does not authorize You to use any trademarks, service marks, trade names, logos or product names of either caBIG® Participant, NCI or caBIG®, except as required to comply with the terms of this License.

4. For sake of clarity, and not by way of limitation, You may incorporate this caBIG® Software into Your proprietary programs and into any third party proprietary programs. However, if You incorporate the caBIG® Software into third party proprietary programs, You agree that You are solely responsible for obtaining any permission from such third parties required to incorporate the caBIG® Software into such third party proprietary programs and for informing Your sublicensees, including without limitation Your end-users, of their obligation to secure any required permissions from such third parties before incorporating the caBIG® Software into such third party proprietary software programs. In the event that You fail to obtain such permissions, You agree to indemnify caBIG® Participant for any claims against caBIG® Participant by such third parties, except to the extent prohibited by law, resulting from Your failure to obtain such permissions.

5. For sake of clarity, and not by way of limitation, You may add Your own copyright statement to Your modifications and to the derivative works, and You may provide additional or different license terms and conditions in Your sublicenses of modifications of the caBIG® Software, or any derivative works of the caBIG® Software as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

6. THIS caBIG® SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES (INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE) ARE DISCLAIMED. IN NO EVENT SHALL SCIENCE APPLICATIONS INTERNATIONAL CORPORATION (SAIC) OR ITS AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS caBIG® SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ABOUT THIS GUIDE

This section introduces you to the *caAdapter Common Mapping and Transformation Service (CMTS) User's Guide*.

Topics in this section:

Purpose

This guide is the companion documentation to caAdapter Common Mapping and Transformation Service (caAdapter CMTS) module. It includes information and instructions for using either the caAdapter CMTS graphical user interface (GUI) Application Programming Interfaces (APIs).

Audience

Typical User

This guide is designed for the following types of users:

- Developers (such as Java programmers and system architects) who want to use the major CMTS APIs to parse, build, and validate XML data/messages driven by schema (XSD); and
- Analysts (medical analysts, database administrators, and business analysts) who need step-by-step procedures for creating XML message instances using.

Prerequisites

This guide assumes that you are familiar with the W3C XML schema recommendation.

Use of CMTS module requires additional prerequisites. For more information, see *Prerequisites for Using caAdapter CMTS* section of this document.

Organization of This Guide

This guide includes the following chapters:

- *Chapter I, Getting Started with caAdapter CMTS,* discusses *CMTS* architecture, functionalities, and implementation technologies.
- *Chapter II, Transformation Mapping,* explains the procedure to work with XML to XML transformation mapping.

- *Chapter III, Data Transformation,* explains the procedures to perform data transformation using CMTS graphic interface.
- *Chapter IV, Programming APIs,* provides detailed instructions to perform data transformation using CMTS programming APIs.
- *Chapter V, Web Service APIs,* provides detailed instruction to perform data transformation using CMTS Web Service APIs.
- *Appendix A, CMTS Web Service Definition Language (WSDL)*, describes SOAP based web service access API for CMTS data transformation portal.
- *Appendix B, CMTS RESTful Web Application Description Language (WADL)*, describes SOAP RESTful web service access API for CMTS data transformation portal.

Recommended Reading

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

Resource	URL
W3C XML Schema Definition (XSD)	http://en.wikipedia.org/wiki/XML_Schema_(W3C)
XML Query (XQuery)	http://en.wikipedia.org/wiki/XQuery
Extensible Stylesheet Language Transformations (XSLT)	http://en.wikipedia.org/wiki/XSLT
XML Path Language (XPath)	http://www.w3.org/TR/xpath/
Unified Modeling Language (UML)	http://www.cdisc.org/models/sds/v3.1/
cancer **B**iomedical **I**nformatics **G**rid (caBIG)	https://cabig.nci.nih.gov/

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 Common Mapping and Transformation Service Development and Management Teams		
Development	***Documentation***	***Program Management***
Eugene Wang2	Carolyn Kelley Klinger3	Sichen Liu1
Ki Sung Um2		
1 Center for Biomedical Informatics and Information Technology (CBIIT) 2 SAIC-Frederic, Inc. 3 Lockheed Martin Management System Designers		
Contacts and Support		
Application Support	http://ncicb.nci.nih.gov/NCICB/support Telephone: 301-451-4384 Toll free: 888-478-4423	
LISTSERV Facilities Pertinent to caAdapter Common Mapping and Transformation Service		
LISTSERV	***URL***	***Name***
caAdapter_Users	https://list.nih.gov/archives/caadapter_users-l.html	Users listserv for caAdapter

Chapter I Getting Started with caAdapter CMTS

This chapter provides an overview of caAdapter Common Mapping and Transformation Service (CMTS) module, its architecture, and its related data standards.

Topics in this chapter include:

- *About caAdapter CMTS*
    - *About XML Schema (XSD)*

About caAdapter CMTS

The National Cancer Institute Center for Biomedical Informatics and Information Technology (NCI CBIIT) developed the caAdapter tool with the goal of creating standards-based infrastructure components to facilitate medical research data exchange. caAdapter is a open source tool kits including Module Mapping Service, HL7 Map and Transformation Service, Global Model Exchange Service, and Common Mapping and Transformation Service (CMTS). The module facilitates data mapping and transformation among different kinds of data sources including CSV, XML, HL7 v2 messages, HL7 v3 messages, and Study Data Tabulation Model (SDTM) data sets.

CMTS module will provide the functionalities of mapping and transformation from source XML data into target XML data. It will be developed on the basis of XML schema (XSD) specification [1]. XSD file defines the structure, content and semantics of source data or target data. The mapping engine creates links between nodes of source XSD and target XSD; specifies data operation functions. The transformation process generates target data based target XSD using source data and processing instructions provided with mapping data. The transformation process also includes data validation against its XSD.

CMTS module supports Common Data Elements (CDEs) annotations to allow sharing of data in the standard caBIG compatible format. The CDE annotation information of a target data is defined by its XSD.

The primary objectives of the CMTS module are:

- Model and represent XSD as metadata of generic XML, ISO 21090 enabled XML, CDA, CCD, CSV, HL7 v2.
- Implement XML data parser conformed with XSD.
- Produce XML instances conformed to XSD specification.
- Support CDE semantic compatibility in the standard caBIG format.
- Implement functionalities data manipulation functions.
- Provide programming API, web service API and RESTful web API for application and service integration.

The objective of the architecture design is to leverage the state-of-art Java and XML technologies, and to organize in a highly modularized fashion that enables easy expansion that can adapt to any future needs. The architecture should also effectively streamline the main functional modules, so that it can achieve an overall high performance.

The architecture diagram is as following.

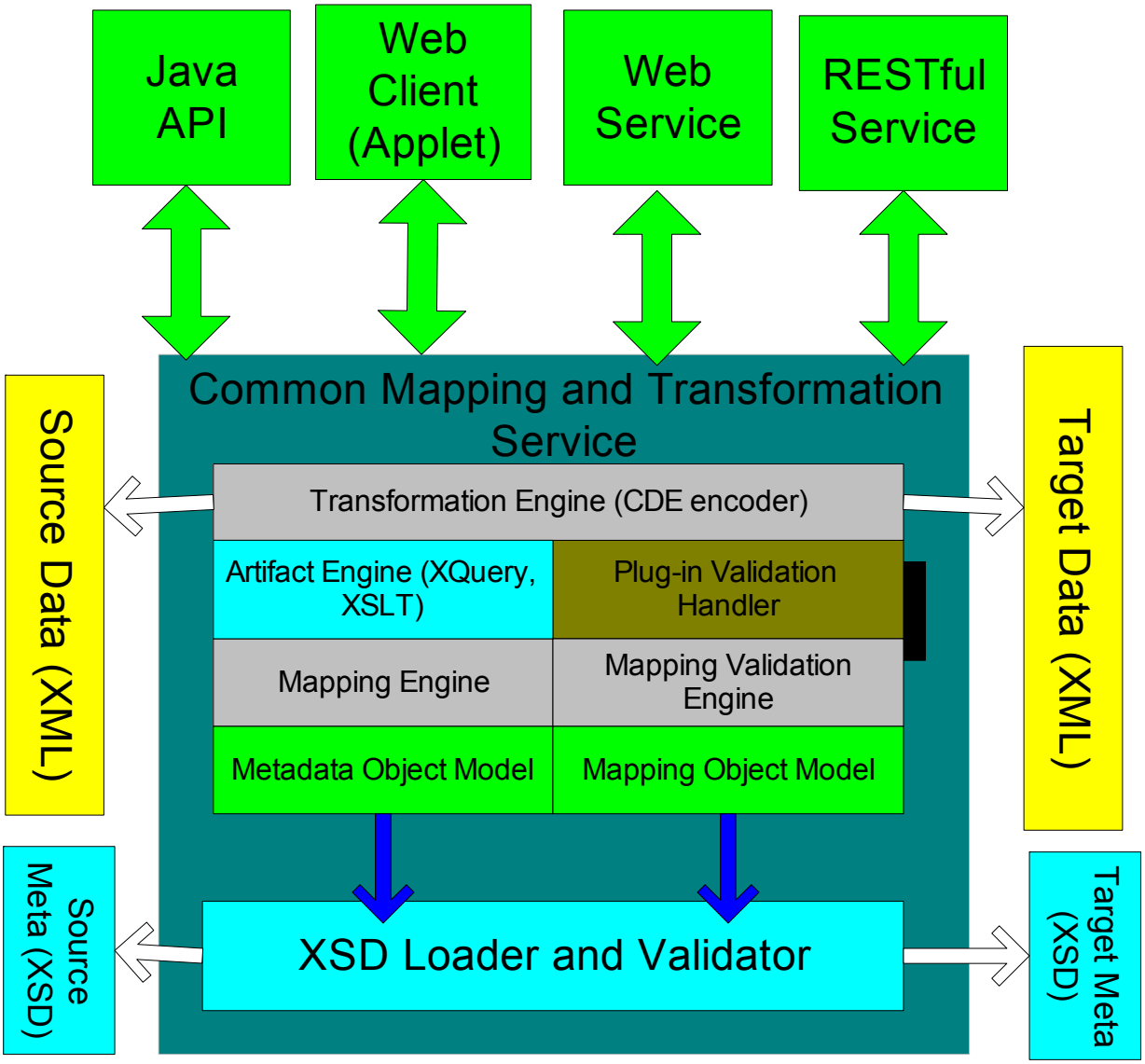


Figure 1.1 Architecture Diagram

About XML Schema (XSD)

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using

some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.

There are languages developed specifically to express XML schemas. The Document Type Definition (DTD) language, which is native to the XML specification, is a schema language that is of relatively limited capability, but that also has other uses in XML aside from the expression of schemas.

The W3C's XML Schema language, also known as XSD (XML Schema Definition), published as a W3C recommendation in May 2001, is one of several XML Schema language. It was the first separate schema language for XML to achieve Recommendation status by the W3C. Like all XML schema languages, XSD can be used to express a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. However, unlike most other schema languages, XSD was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types.

About XML Query

XQuery is a query language (with some programming language features) that is designed to query collections of XML data. It is semantically similar to SQL.

XQuery is standardized by W3C, the organization that maintains all XML related standards such as XML, XML Schema, SOAP, and WSDL. It also gains support from major software vendors such as Microsoft, IBM, Oracle, and SUN Microsystems from the very beginning of its standardization process.

XQuery provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML, such as relational databases or office documents.

XQuery is a programming language that can express arbitrary XML to XML data transformations with the following features:

- Logical/physical data independence
- Declarative
- High level
- Side-effect free
- Strongly typed language

About XSLT

XSLT is a declarative, XML-based language used for the transformation of XML documents. The original document is not changed; rather, a new document is created based on the content of an existing one. The new document may be serialized (output) by the processor in standard XML

syntax or in another format, such as HTML or plain text. XSLT is most often used to convert data between different XML schemas or to convert XML data into web pages or PDF documents.

XSLT is developed by W3C. XSLT 1.0 was published as a Recommendation by the W3C in 1999. After 2001, the XSL working group joined forces with the XQuery working group to create XPath 2.0, with a richer data model and type system based on XML Schema.

The XSLT processing model involves:

- one or more XML source documents;
- one or more XSLT stylesheet modules;
- the XSLT template processing engine (the processor); and
- one or more result documents.

XSLT capabilities overlap with XQuery. But the two languages are rooted in different traditions and serve the needs of different communities. XSLT was primarily conceived as a stylesheet language whose primary goal was to render XML for the human reader on screen, on the web (as web template language), or on paper. XQuery was primarily conceived as a database query language in the tradition of SQL. XSLT is stronger in its handling of narrative documents with more flexible structure, while XQuery is stronger in its data handling, for example when performing relational joins.

Prerequisites for Using caAdapter CMTS

The following skills will ensure successful use of caAdapter CMTS:

- Familiarity with XSD
- Understanding various data formats compatible with XML
- *Optionally, familiarity with XQuery and its transformation tools.*
- *Optionally, familiarity with XSLT and its transformation tools.*
- *Optionally, familiarity Apache Ant if you need modify and build CMTS application.*
- Training on caAdapter CMTS
- Familiarity with caAdapter CMTS mapping rules

Starting caAdapter CMTS

Starting caAdapter CMTS User's Deployment

You deploy CMTS application on your own server, follow these steps:

1. Download caAdapter-CMTS.src from caAdapter project download site.
2. Extract the download file temporary directory, for example c:\cmts.
3. In a Command Prompt window, go to the home directory in step 2.
4. Enter ant, the default task creates two web application files, caAdapter-cmts.war, caAdapterWS-cmts.war.
5. Alternatively, you can skip step 1 to 4 and directly download caAdapter-cmts.war from caAdapter project download site.

6. Deploy caAdapter-cmts.war file on your web server
7. Launch CMTS from deployment: http://webserver:port/caadapter-cmts/index.html

Starting caAdapter CMTS Online

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

Once you can directly launch it with link: https://caadapter.nci.nih.gov/caadapter-cmts/index.html.

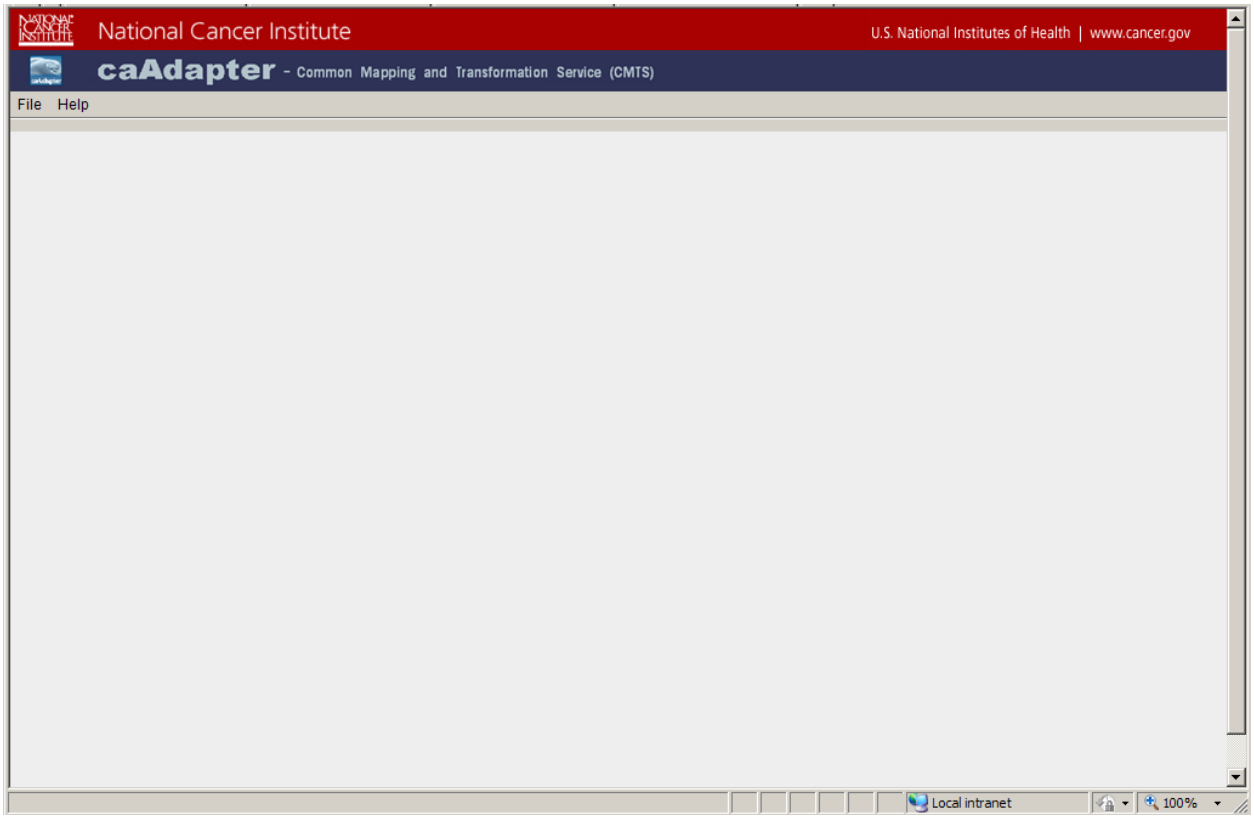


Figure 1.2 CMTS Startup Screen

caAdapter CMTS User Interface

caAdapter CMTS interface 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.

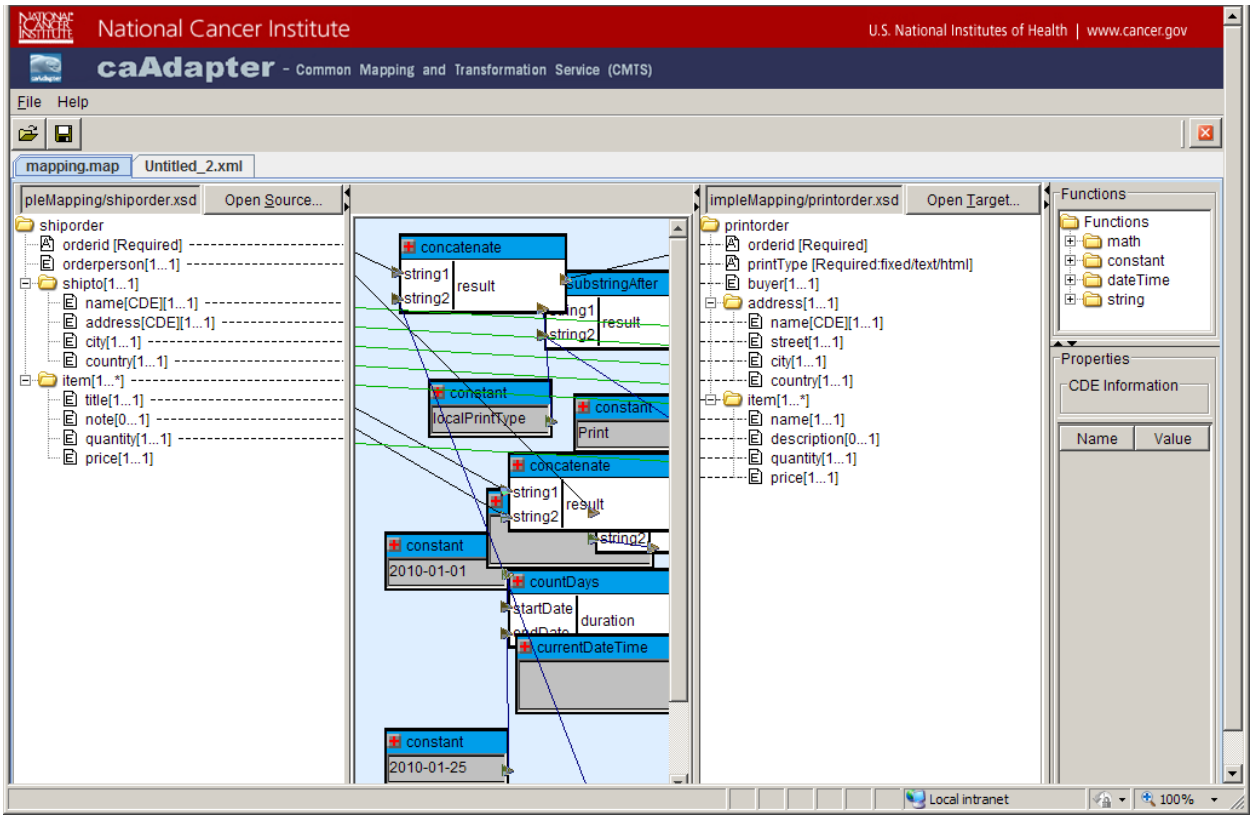


Figure 1.3 caAdapter CMTS Main Window

File Menu

File menu contains options to create map, artifact, result data, as well as to open, save, and close the selected files. The following table explains these menu options.

File Menu Item	***Description***
New	Creates a new file for the type of file you select including: • Transformation Mapping • XML to XML Transformation • XQuery Artifact • XSLT Artifact
Open	Opens an existing transformation mapping file(.map)
Save	Saves the file you are currently working on (in the selected tab), including: • Transformation Maping • Result XML Data • XQuery Artifact • XSLT Artifact
Save As	Opens a Save As dialog box to allow you to save the selected file to another file name

Close	Closes the file you are currently working on in the selected tab.
Close All	Closes all open files.
Exit	Close all open files and exit caAdapter CMTS application

Table 1.1 File menu commands

Help Menu

Help menu provides options to learn more about caAdapter CMTS module. The following table explains these menu options.

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

Table 1.2 Help menu commands

Tabs

caAdapter CMTS uses a document-oriented paradigm where up to four different file types can be open at the same time. All the open files coexist within the same single window but each is in its own tab panel. The four different types of tabs are:

XML to XML transformation mapping files (map)
XML data files (.xml)
XQuery artifact files (.xql)
XSLT artifact files (.xsl)

In some cases, such as with `.map`, only one of this file type may be open at a time. If you open a new file of the same type, 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.

Chapter II Transformation Mapping

A transformation mapping defines the relationship among source data schema, target schema and data manipulation functions. It also contains annotation information about the source data schema and target data schema.

caAdapter CMTS provides a link between one of the following:

- a source data schema entry to a target data schema entry
- a source data schema entry to a function's input port
- a function's output port to a target data schema entry

The annotation information set constraint on data schema, including:

- one and only one chosen child element of a choice group
- one or more cloned record(s) of an element

Transformation Mapping Rules

CMTS module enforces the following mapping rules:

- It must contain a valid mapping pair (source data schema and target data schema files).
- The source entry referenced must exist in the source data schema
- The destination entry referenced must exist in the target data schema
- A mandatory target entry must have either a mapping from data source or predefined default value in its schema; the mapping data source could be a source entry or function output
- Each input parameter for a function must have a mapping from source or a constant defined.
- Each output parameter for a function must have a mapping to target

Understand Transformation Mapping

This section includes the following topics:

- Overview of the Transformation Mapping Panel
- Create New transformation Mapping
- Create Mapping Link
- Delete Mapping Link
- Save Transformation Mapping
- Open Transformation Mapping
- Use Functions in Transformation Mapping
- Annotate Schema

Overview of the Transformation Mapping Panel

The transformation mapping panel graphically presents user a transformation mapping with source and target meta data, schema annotation, mapping links, function boxes, etc. It is divided into several sub-panels (*Figure 3.2*) described as following:

Source schema panel – The left-hand panel contains tree view of source schema
Target schema panel – The right-hand panel contains tree view of target schema
Mapping panel – The center panel displays mapping lines and functions.

Functions panel – The upper panel of the most right split pane display a tree view of all applicable functions.

Properties panel – The lower panel of the most right split pane displays detail information of a selected item, such as link properties, schema entry properties, and function properties.

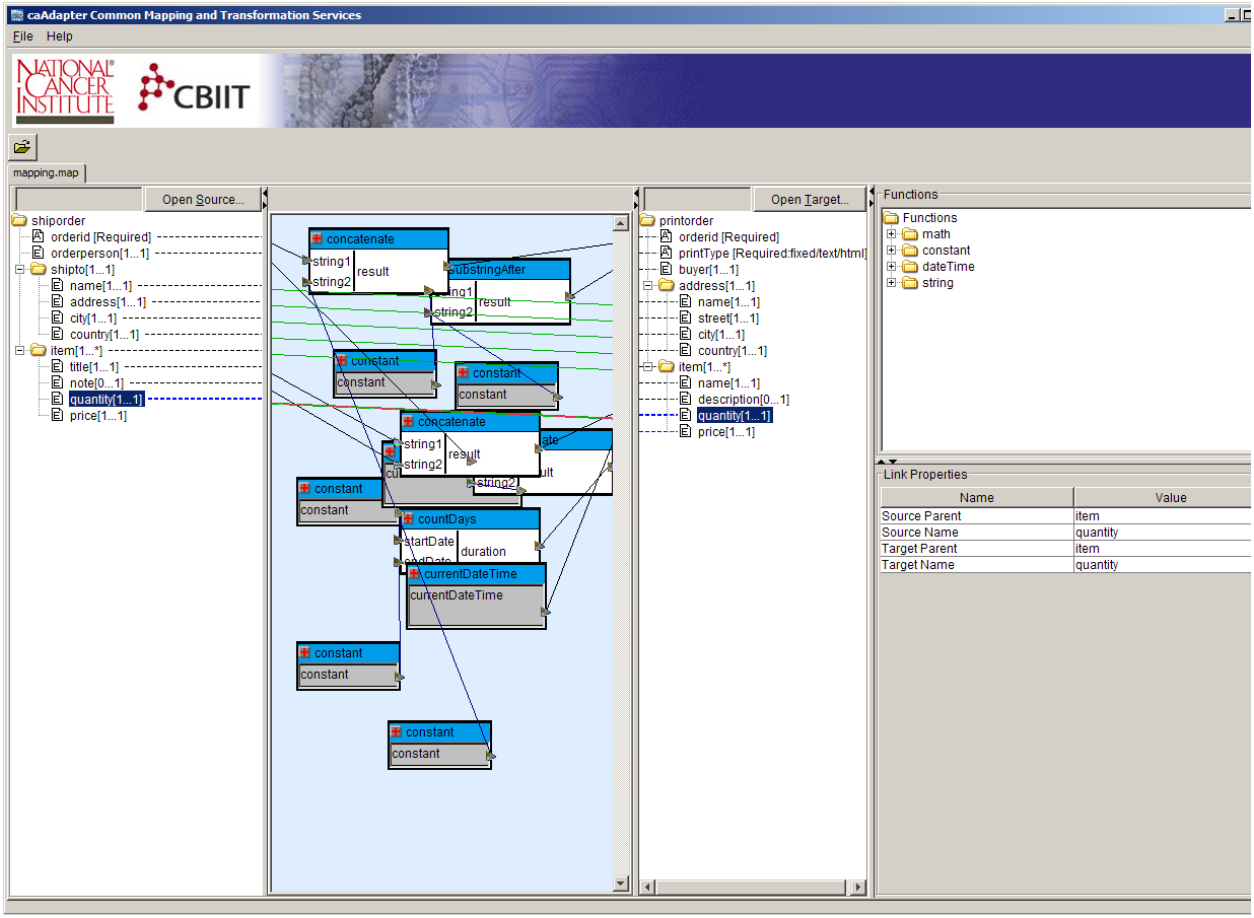


Figure 2.1 Transformation Mapping Panel

The tree view of schema data is read-only on the mapping panel, no change is allowed on any node on source schema tree or target schema tree. If user want change the referenced schema, user must work it with available XSD editing tools.

Caution: Editing or removing source or target elements may result in dropping of related mapping (link) or producing other unpredictable behavior.

The following sections describe how to create, save, open, or update a transformation mapping.

Create New Transformation Mapping

To create a new transformation mapping, perform the following steps:

1. Select **File > New > Transformation Mapping** from the menu bar

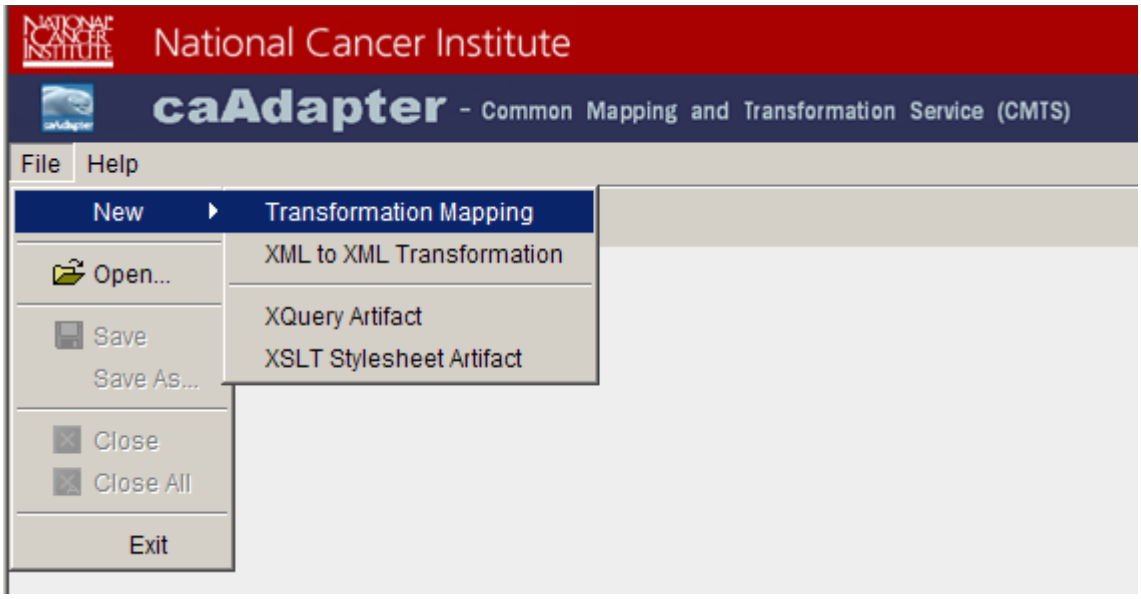


Figure 2.2 New Transformation Mapping

2. System creates a new mapping tab with empty source and target panels.

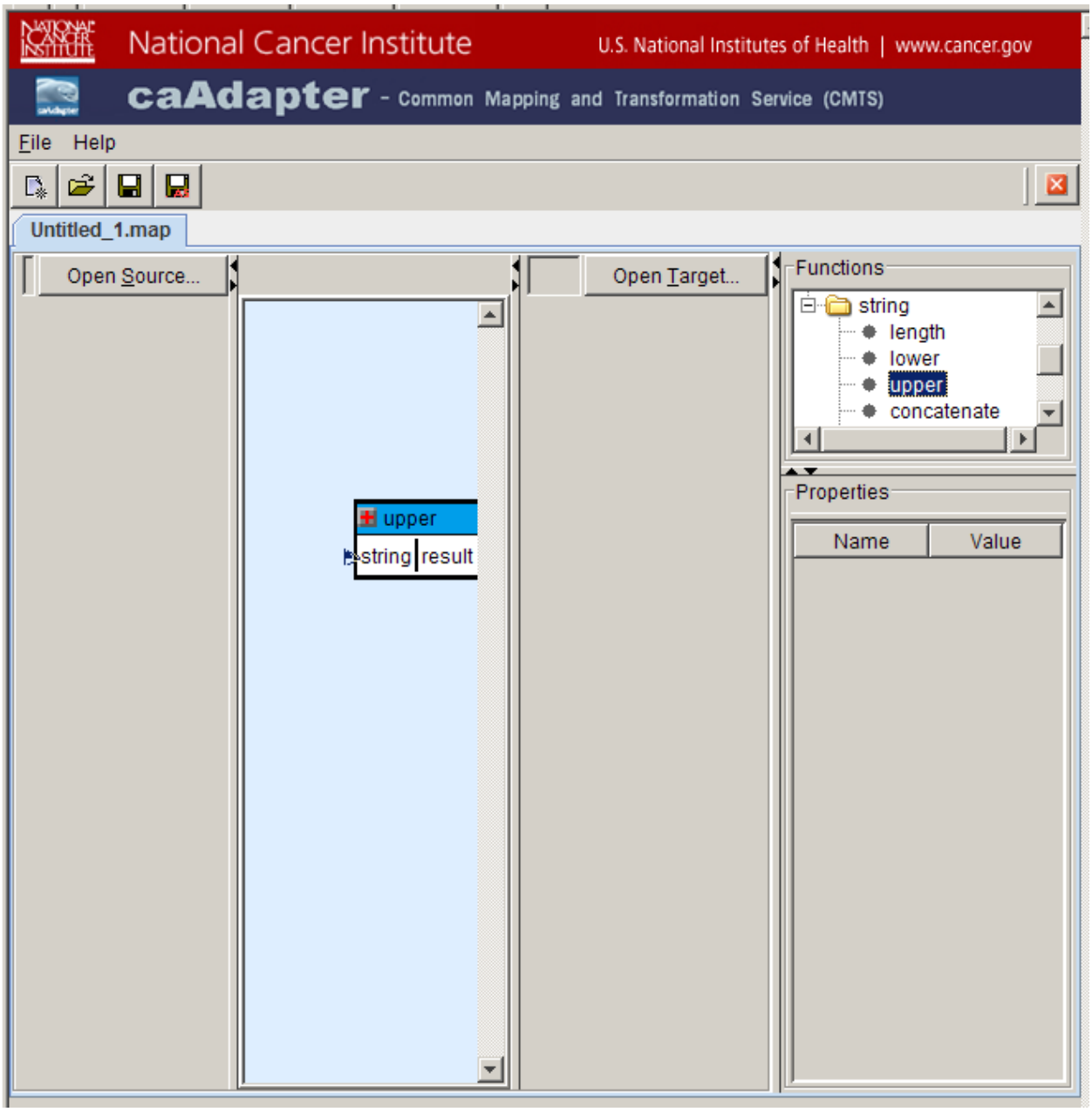


Figure 2.3 Empty Transformation Mapping Panel

3. Click **Open Source** button on source schema panel.
4. **Open Source data schema** dialog appears.

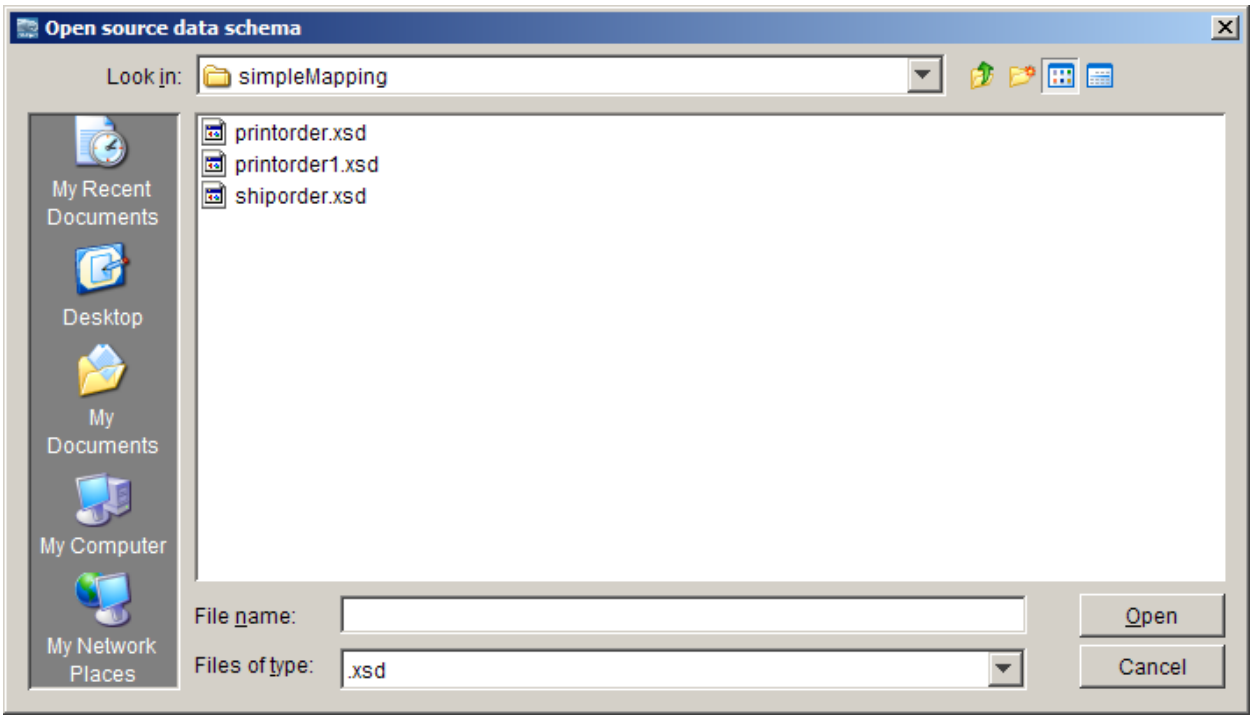


Figure 2.4 Open Source Data Schema Dialog

5. Select the source file and click **Open** to populate the source schema tree on source panel.
6. Click **Open Target** button on target schema panel.
7. **Open target data schema** dialog appears.

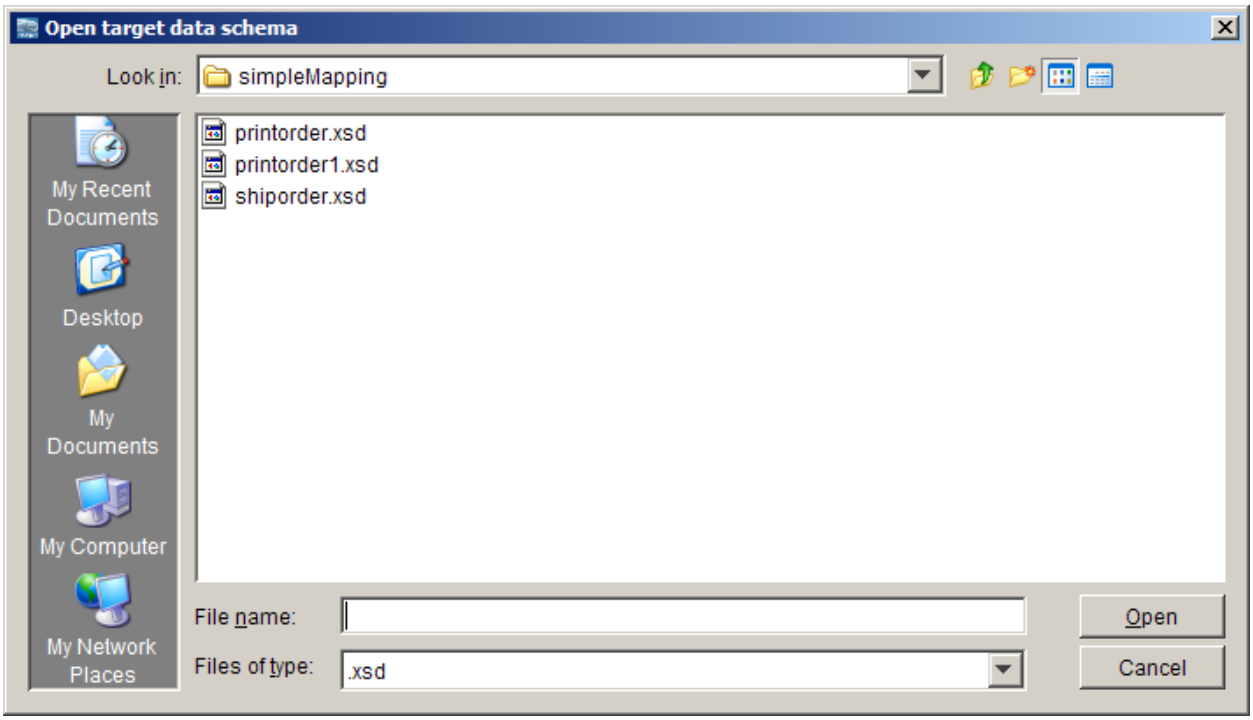


Figure 2.5 Open Target Data Schema Dialog

Create Mapping Link

To create a mapping link, perform the following steps:

1. Select a source entry and drag it to the appropriate target entry.
2. Drop the source on the target element.
3. Once a source field is mapped to a target element, a mapping line appears between them in the mapping panel. The following screen shot shows a mapping line between **address[1…1]** node on source schema tree and **shipto[1…1]** node on target schema tree.

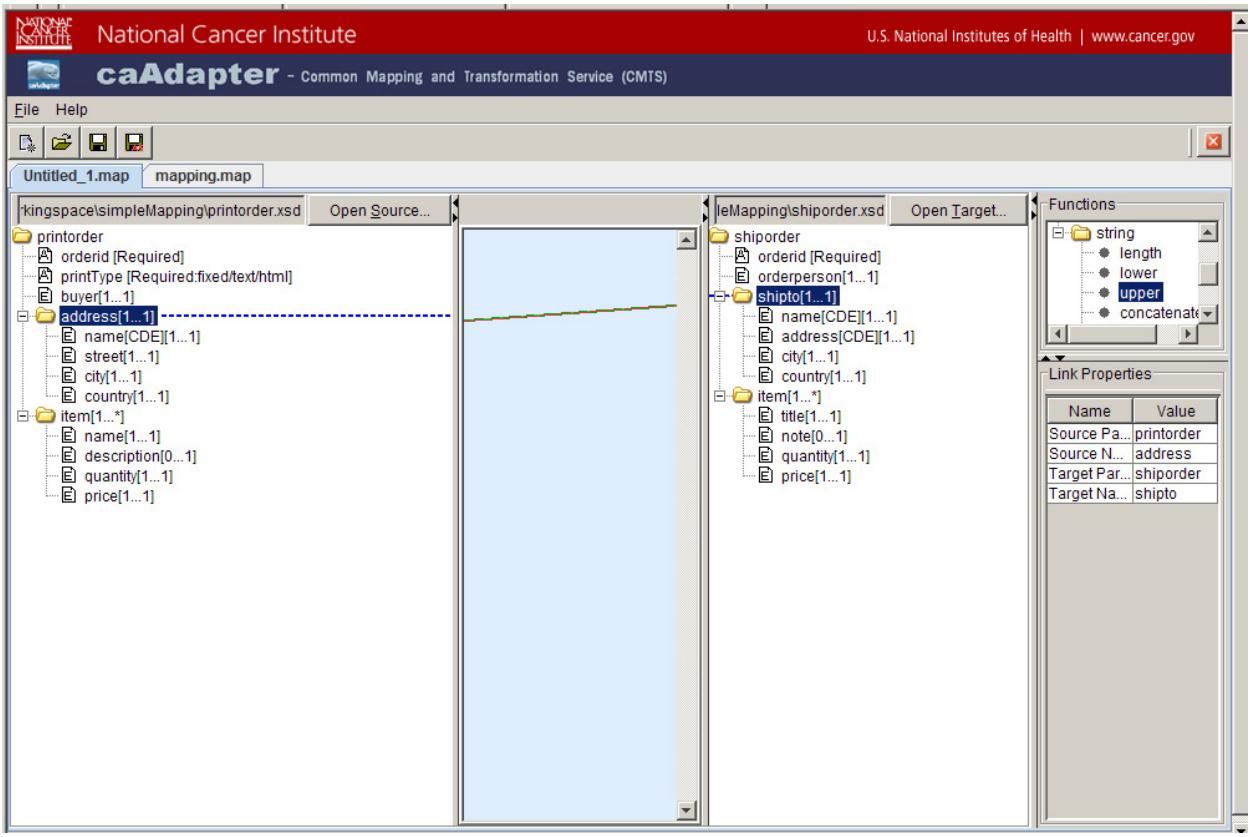


Figure 2.6 Mapping Link from Source Schema Node to Target Schema Node

Note: When a mapping link is selected, the link and associated mapping ends are highlighted; the property panel displays the link's properties.

Delete Mapping Link

To delete a mapping link, perform the following steps:

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

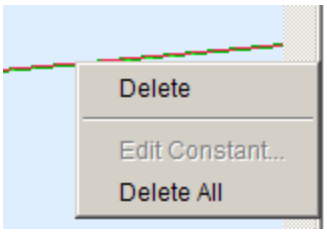


Figure 2.7 Delete Mapping Link

2. Click **Delete** to delete only the selected mapping line.
3. Alternatively, click **Delete All** to delete all mapping links.
4. A confirmation message box appears.
5. Click ***Yes*** to delete all mapping links.

Save Transformation Mapping

To save a transformation mapping, perform the following steps:

1. Select **File > Save** menu bar or ***Save*** icon on tool bar.
2. If the open mapping is an existing one, the latest mapping is saved to the referenced map file
3. If the open mapping is a new one, a file chooser dialog appears.
4. Choose or input file name, the latest mapping is saved to the chosen file.
5. Alternative, to save a mapping to different file, perform the following steps.
6. Select **File > Save As** on menu bar or **Save As** icon on tool bar.
7. A file chooser dialog appears.

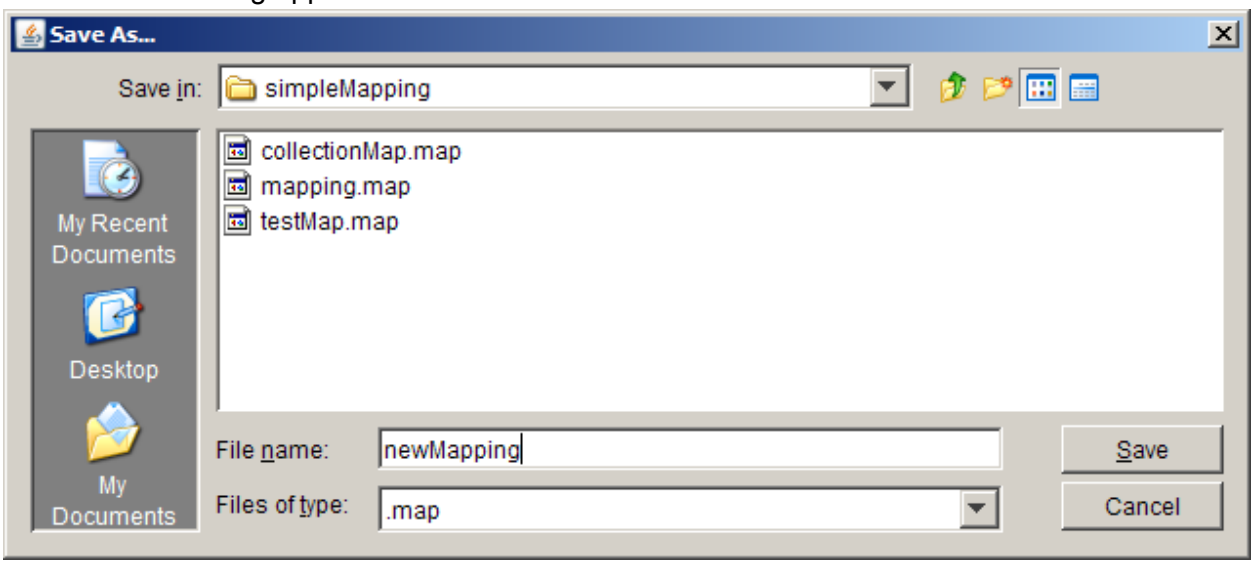


Figure 2.8 Choose or Input File Name to Save a Mapping

8. Choose or input file name, the latest mapping is saved to the chosen file

Caution: Transformation mapping has internal references to associated source schema and target schema files which are parts of the transformation mapping. If a transformation mapping file is moved to other location or other system, all the associated schema files have be moved together.

Open Transformation Mapping

To open an existing transformation mapping, perform the following steps:

1. Select **File > Open > Open** on menu bar or click ***Open*** icon on tool bar

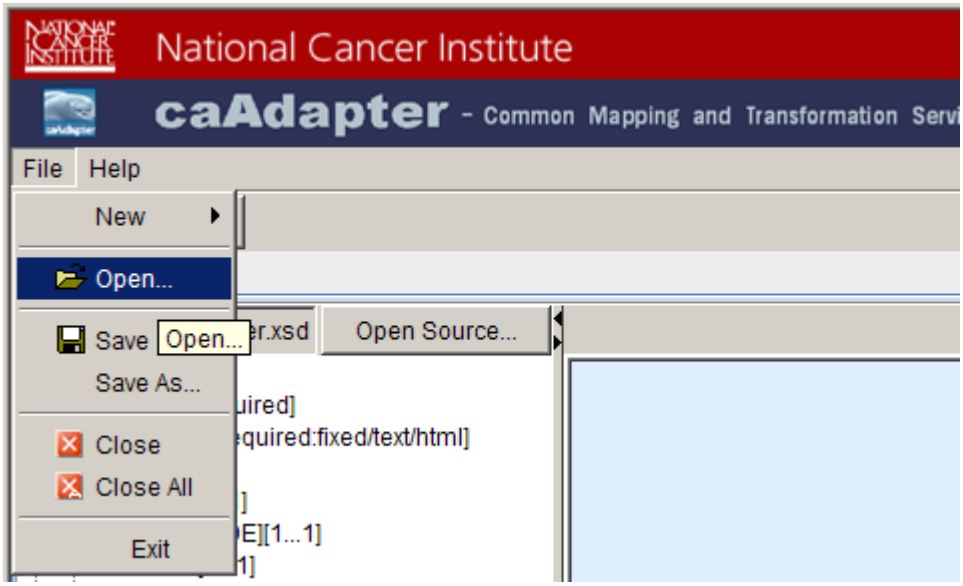


Figure 2.9 Open Existing Mapping

2. A file chooser dialog appears
3. Choose a mapping file, and click ***Open***

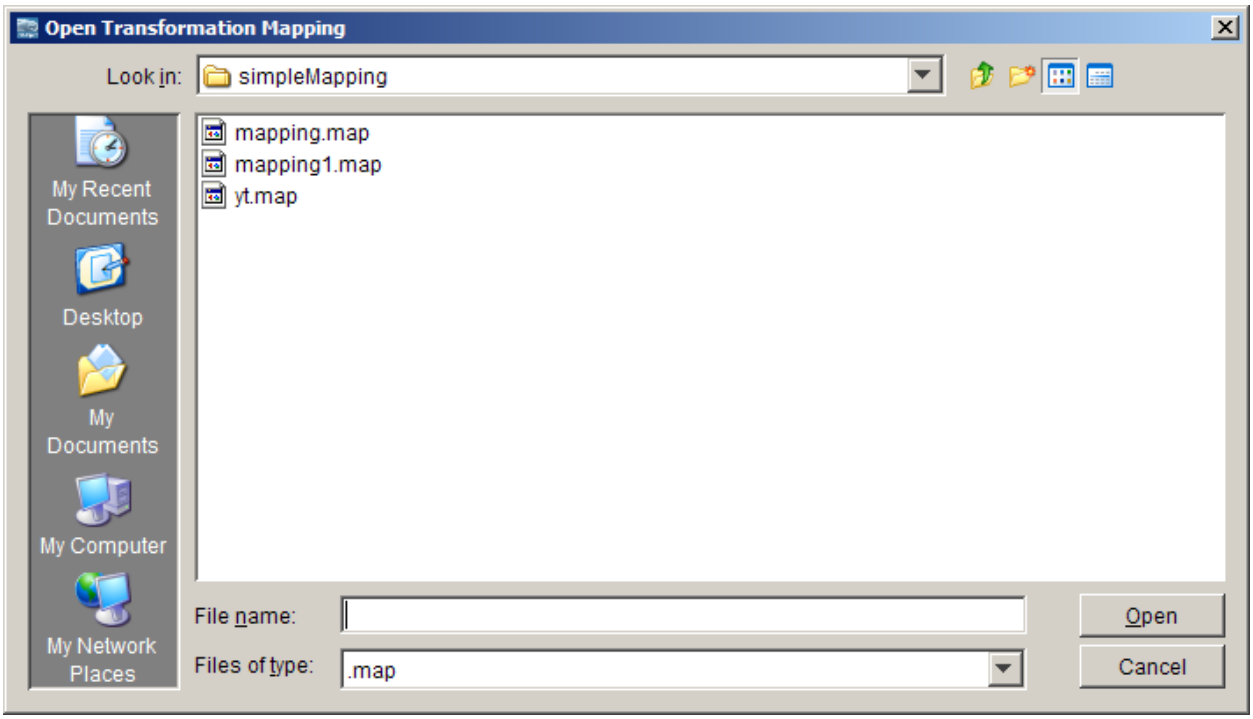


Figure 2.10 Choose a Mapping File to Open

4. Mapping panel displays mapping links, source data schema, target data schema and data processing functions.

Use Data Processing Functions in Transformation Mapping

The **Functions** panel lists all the data processing functions that facilitate the CMTS data transformation requirement. These data processing functions are grouped by functional categories:

- math
- constant
- dataTime
- string

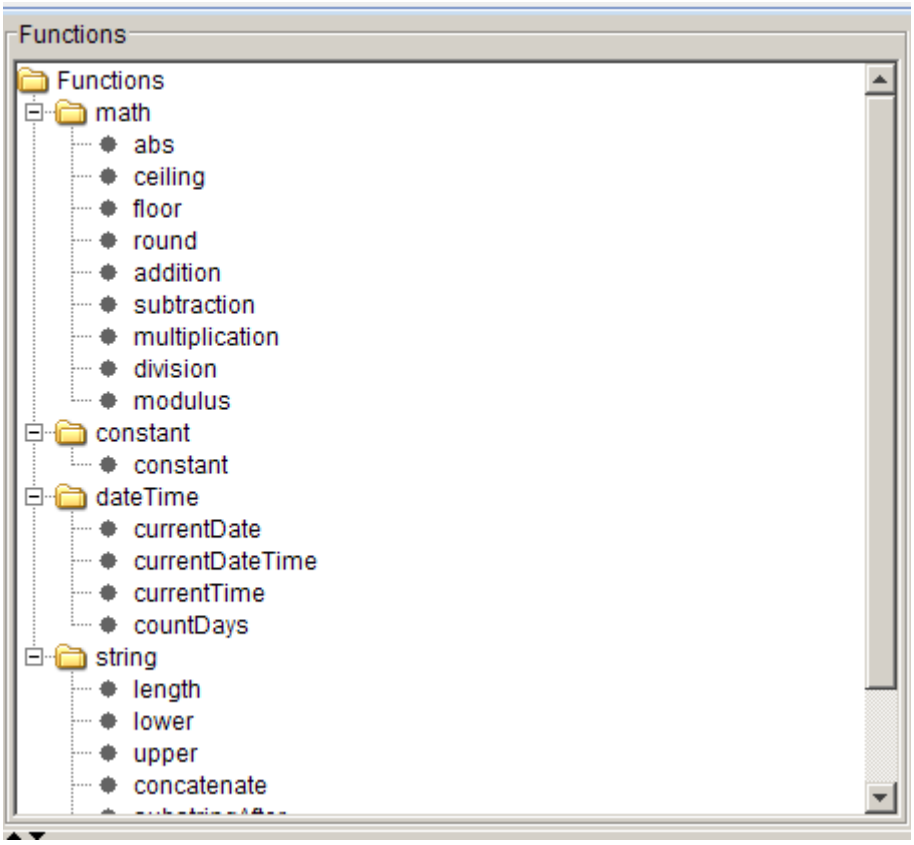


Figure 2.11 Data Processing Functions

Add Function to Transformation Mapping

To add a function item in transformation mapping, perform the following steps:

1. Select a function in the **Functions** panel.
2. Drag-and-drop the required function from the **Functions** panel to the mapping central panel
3. Move this function box around the mapping panel.

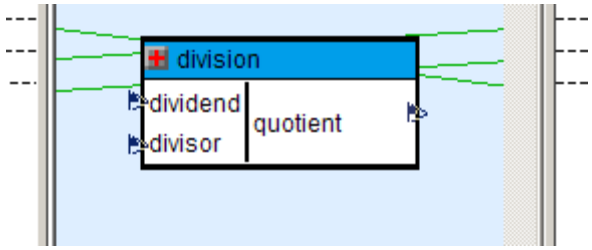


Figure 2.12 Add Function to Transformation Mapping.

Create Mapping from Source schema entry to Function Input Port

To create a mapping link from source schema entry to function input port, perform the following steps:

1. Select a schema entry from source schema tree

2. Drag-and-drop the selected schema entry to an input port of function box on the mapping panel.

Create Mapping from Function Output Port to Target Schema entry

To create a mapping link from a function output port to a target schema entry, perform the following steps:

1. Select a schema entry from target schema tree
2. Drag-and-drop the selected schema entry to an output port of function box on the mapping panel.

Delete Function from Transformation Mapping

To delete a function box from transformation mapping, perform the following steps:

1. Select a function box in the **mapping** panel.
2. Right click the selected function box, a popup menu appears.
3. Click **Delete** menu item.
4. The selected function box is removed from **mapping** panel.

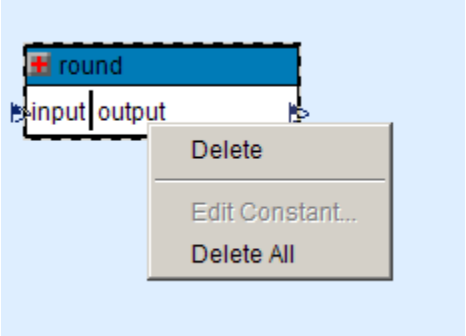


Figure 2.13 Delete Function from Transformation Mapping

Edit Constant Function

To edit a constant function, perform the following steps.

1. Select a constant function in the mapping panel.
2. Right-click and selected function box, a popup menu appears.
3. Click **Edit Constant** menu item.
4. The Edit Constant dialog box appears.

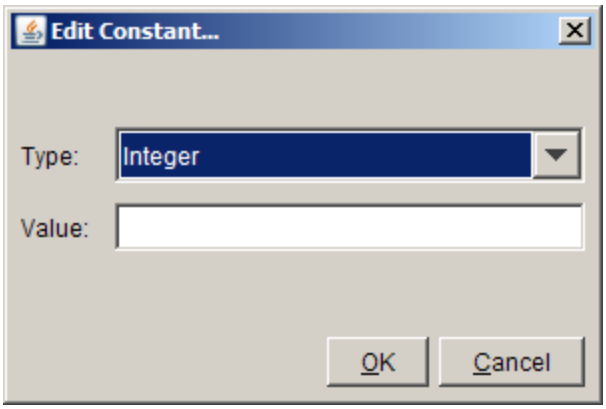


Figure 2.14 Edit Constant Function

4 Change the **Type** and/or **Value** for the constant
5 Click **OK**, the edited function is assigned with new value.

Use Math Function Group
The **math** function group includes the following functions:

- abs – returns the absolute value of the argument
- ceiling -- returns the smallest integer that is greater than the number argument
- floor -- returns the largest integer that is not greater than the number argument
- round -- rounds the number argument to the nearest integer
- addition – returns the **sum** of **term1** and **term2**
- subtraction – returns the **difference** of **term2** from **term1**
- multiplication – returns the **product** of **factor1** and **factor2**
- division – returns the **quotient** of the **dividend** and **divisor**
- modulus – returns the ***rounded*** **quotient** of the **dividend** and **divisor**

Use DateTime Function Group
The **dateTime** function group includes the following functions:

- currentDate – returns the current date (with timezone)
- currentDateTime -- returns the current dateTime (with timezone)
- currentTime -- returns the current time (with timezone)
- countDays -- returns an integer that represents the day component in the localized value of the argument

Use String Function Group
The **string** function group includes the following functions:

- length – returns the length of the specified string.
- lower -- converts the string argument to lower-case
- upper -- converts the string argument to upper-case

- concatenate -- returns the concatenation of the strings
- substringAfter – returns the remainder of string1 after string2 occurs in it
- substringBefore – returns the start of string1 before string2 occurs in it
- substring – returns the substring from the start position to the specified length. Index of the first character is 1.
- replace – Returns a string that is created by replacing the given pattern with the replace argument

Annotate XML Schema

In schema conforming step, caAdpater CMTS annotate an XML schema with the following kinds of actions:

- Choice Element – XML Schema choice element allows only one of the elements contained in the <choice> declaration to be present within the containing element. The annotation process allows user select one and only element to be present.

- Clone Element – If an element has the “maxOccurs” attribute greater than one or “unbounded”, it could be mapped to more than one source/target nodes. The annotation process allows user to clone the same element for multiple mapping source/target nodes.

- Recursive Data Type – If the data type of element is same with its ancestor, it is referred as recursive element. As default, caAdapter CMTS mapping panel only display the name of recursive data type without any content to avoid any indefinite loop. The annotation process allows user to navigate a recursive data type down level by level.

Select Choice Child Element

To select a child for choice element, perform the following steps:

1. Right click a child element of “<choice>” element. A popup menu appears.
2. The **Select Choice** menu item is enabled if the selected element has not been selected.
3. Click **Select Choice** menu item, the selected child would set as “***Selected***” child for the choice element and the other “Selected” child would be removed.

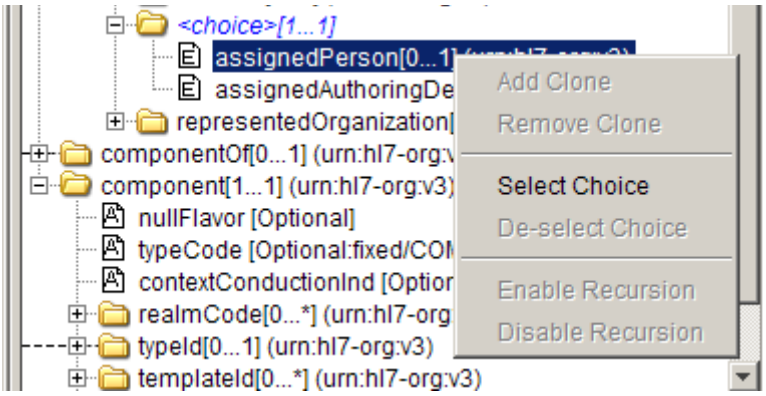


Figure 2.14 Select Choice Child Element

De-select Choice Child Element

To de-select a child for choice element, perform the following steps:

1. Right click a child element of "<choice>" element. A popup menu appears.
2. The **De-select Choice** menu item is enabled if the selected element has been designed as "S***elected***" child before.
3. Click **De-select Choice** menu item, the selected child would be released from "***Selected***" status.

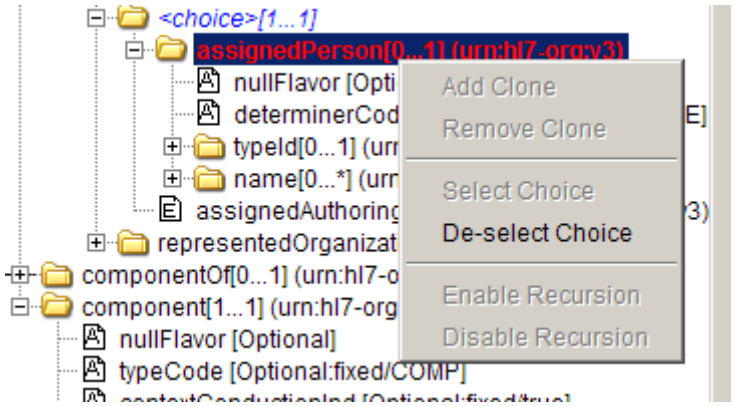


Figure 2.16 De-select Choice Child Element

Add Clone Element

To clone an element, perform the following steps:

1. Right click element. A popup menu appears.
2. The **Add Clone** menu item is enabled if the selected element has the "maxOccurs" attribute greater than one or "unbounded"
3. Click **Add Clone** menu item, the selected element would be cloned and added to the schema tree. User can only clone an element from the "original" element, i.e. the cloned element can never be cloned.

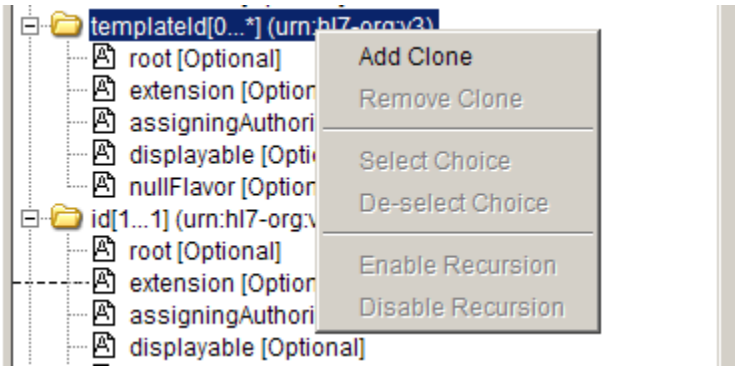


Figure 2.17 Add Clone Element

Remove Clone Element

To remove a cloned element, perform the following steps:

1. Right click element. A popup menu appears.
2. The **Remove Clone** menu item is enabled if the selected element is a cloned element.
3. Click **Remove Clone** menu item, the selected element would be removed from the schema tree.

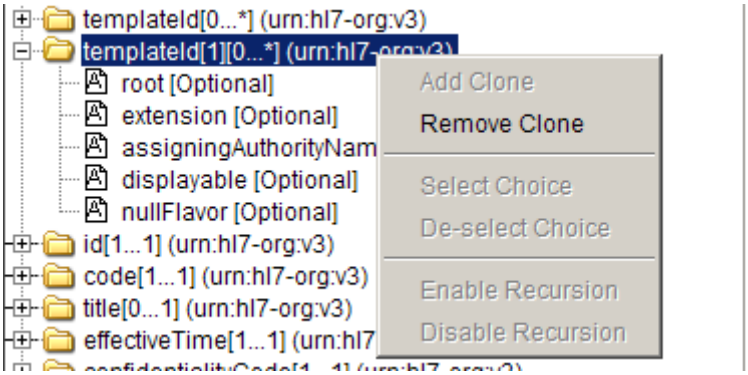


Figure 2.18 Remove Clone Element

Enable Recursion

To navigate a recursive data type down to next level, perform the following steps:

1. Right click element. A popup menu appears.
2. The **Enable Recursion** menu item is enabled if:
    a. The selected element has recursive data type.
    b. The selected element does not have its content displayed in schema tree.
3. Click **Enable Recursion** menu item, the selected element would navigate down one more level.

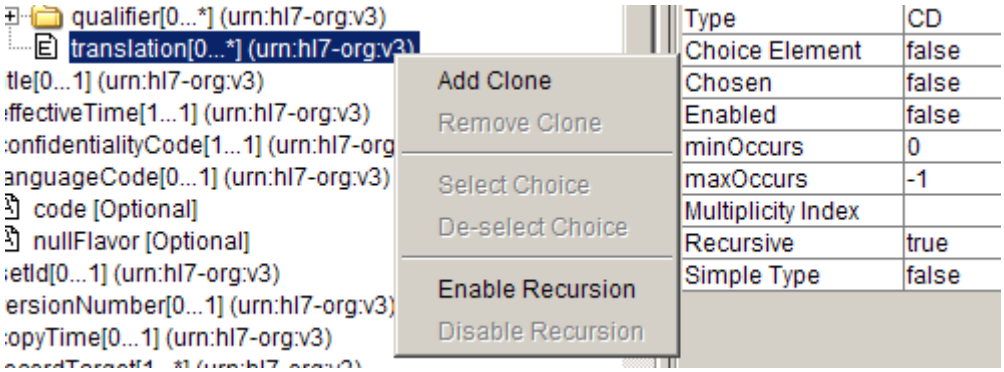


Figure 2.19 Enable Data Type Recursion

Disable Recursion

To disable a recursive element, perform the following steps:

1. Right click an element. A popup menu appears.
2. The **Disable Recursion** menu item is enabled if:
    a. The selected element has recursive data type and
    b. The selected element has its contents displayed in schema tree.
3. Click **Disable Recursion** menu item, the selected element would remove all its contents.

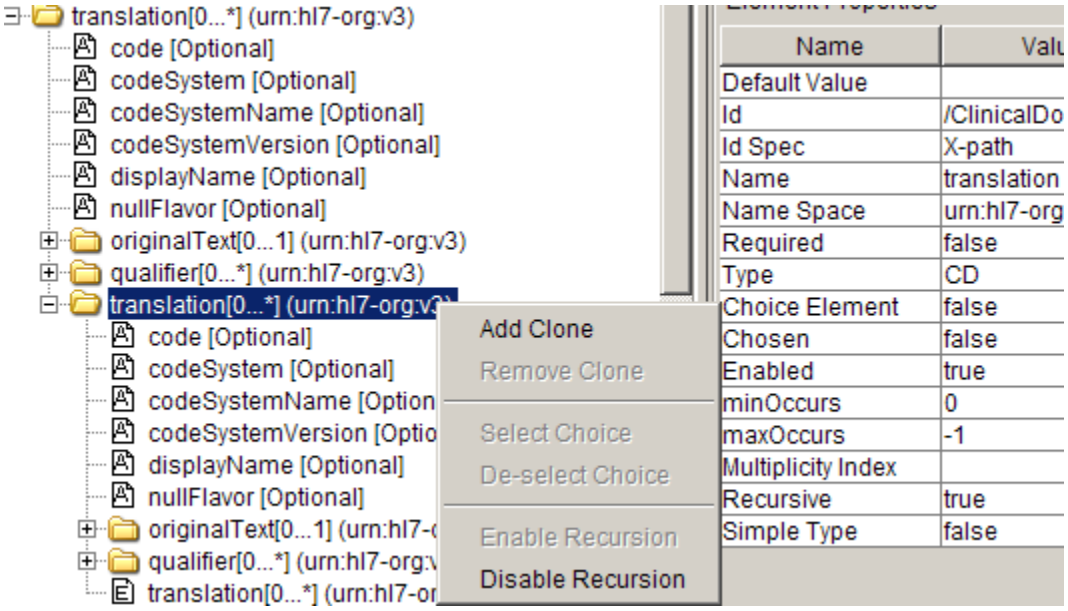


Figure 2.20 Disable Data Type Recursion

Chapter III Data Transformation

CMTS transformation engine enable user to convert source data into target data using transformation instructions. It supports three formats of transformation instruction:

- Transformation Map (.map) – Created with CMTS mapping tools.
- XQuery Script (.xql) – Created with CMTS XQuery Artifact Generator or other compatible XQuery editing tools.
- XSLT Stylesheet (.xls) – Created with CMTS XSLT Artifact Generator or other compatible XSLT stylesheet editing tools.

The transformation engine is integrated with XML schema (xsd) validator. It validates the generated data against target data schema.

XML to XML Transformation

To transfer source XML to target XML, perform the following steps:

1. Select **File > New > XML to XML Transformation** from menu bar.

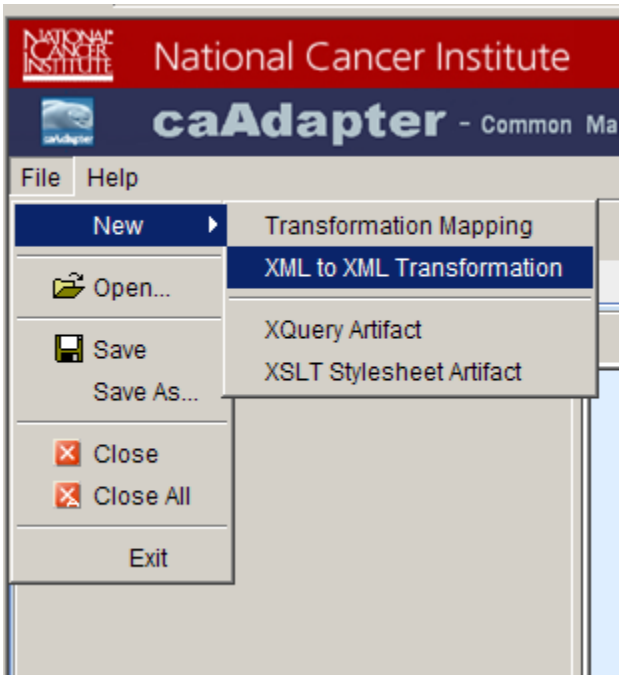


Figure 3.1 XML to XML Transformation

2. XML to XML Transformation dialog appears.

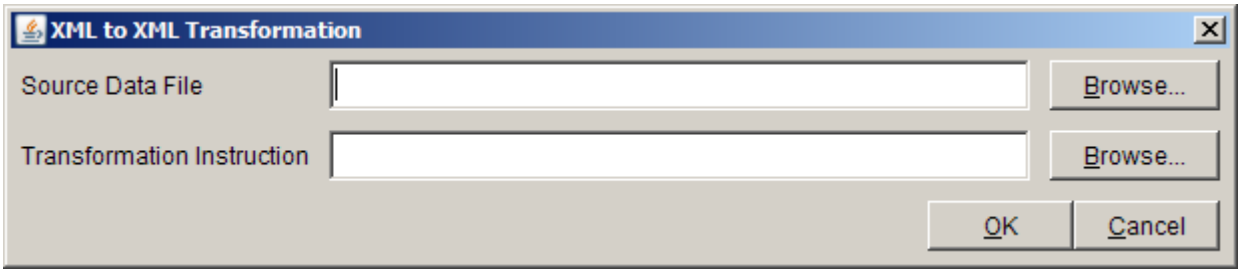


Figure 3.2 XML to XML Transformation Inputs Selection

3. Click **Browser** button next to ***Source Data File*** box
4. ***Open Source Data File*** dialog appears.

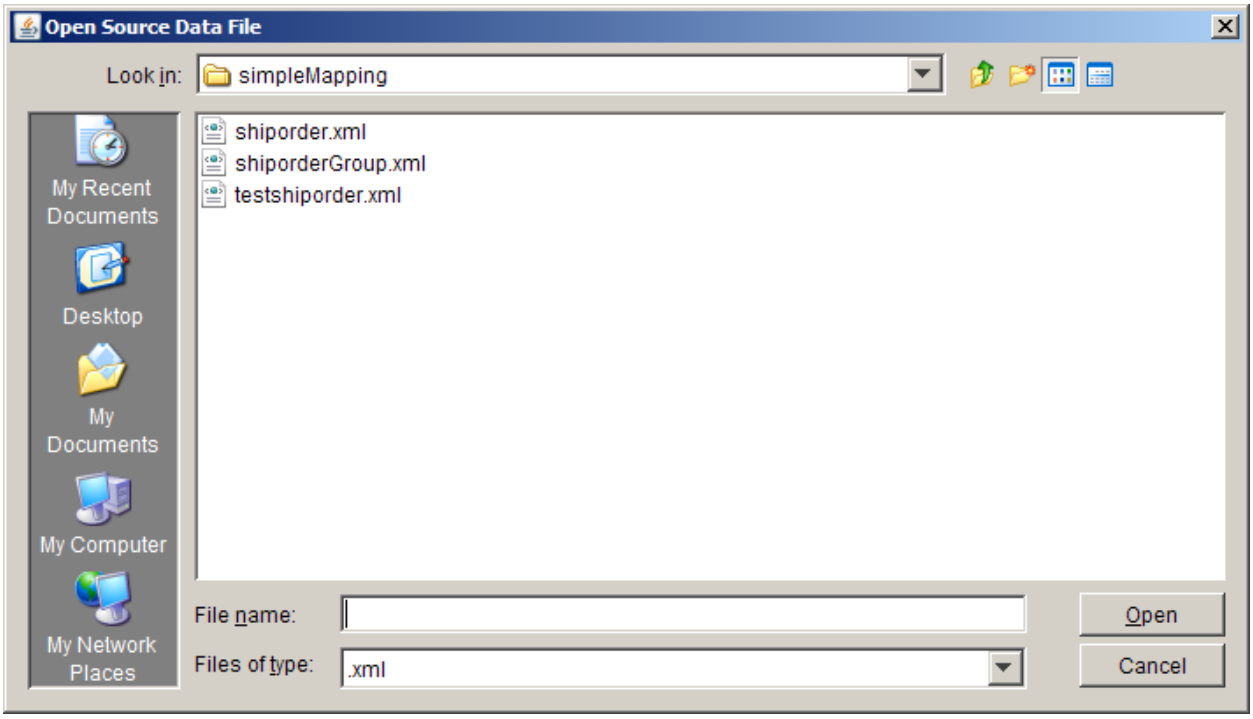


Figure 3.3 Open Source Data File

5. Choose source data XML file and click ***Open*** button
6. Click Browser next to Transformation Instruction box
7. ***Open Transformation Instruction*** dialog appears

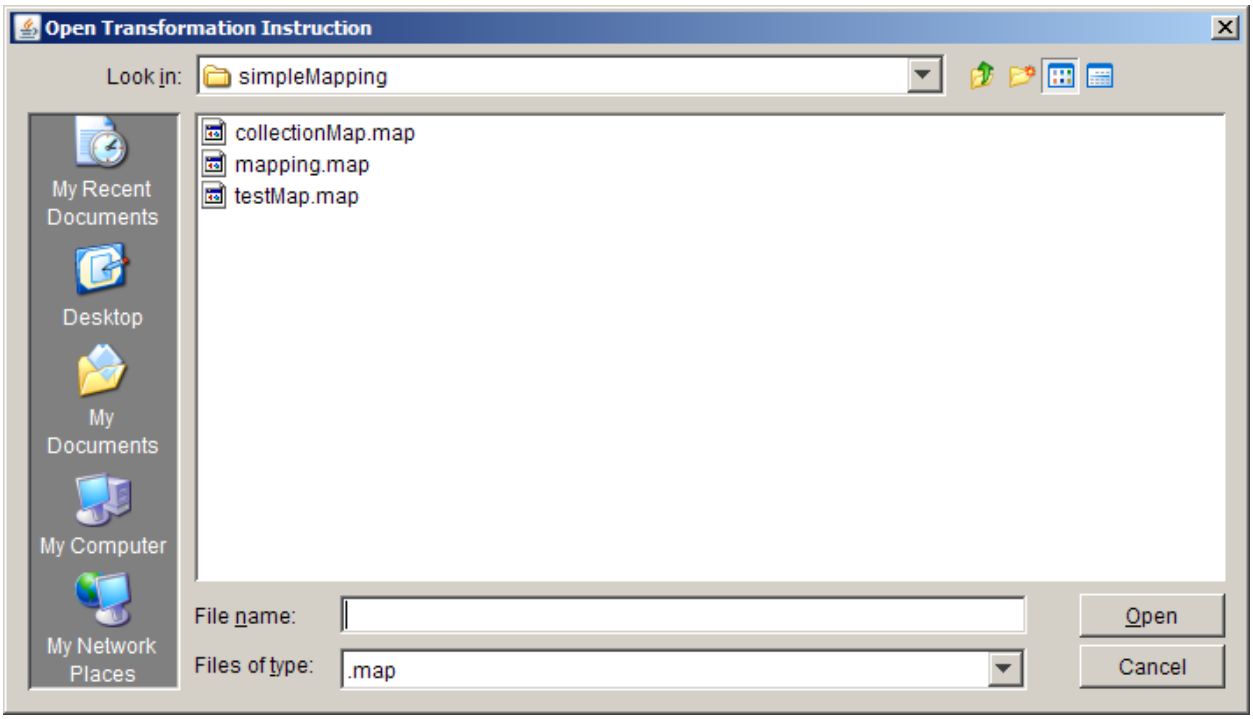


Figure 3.4 Open Transformation Instruction

8. Choose transformation file in one of following formats:
   a. XML to XML transformation mapping (.map)
   b. XQuery script (.xql)
   c. XSLT stylesheet (.xsl)
9. Click ***Open*** button to select the transformation instruction file
10. Click ***OK*** button on ***XML to XML Transformation*** dialog.
11. Transformation engine generates target data and display it on result panel.

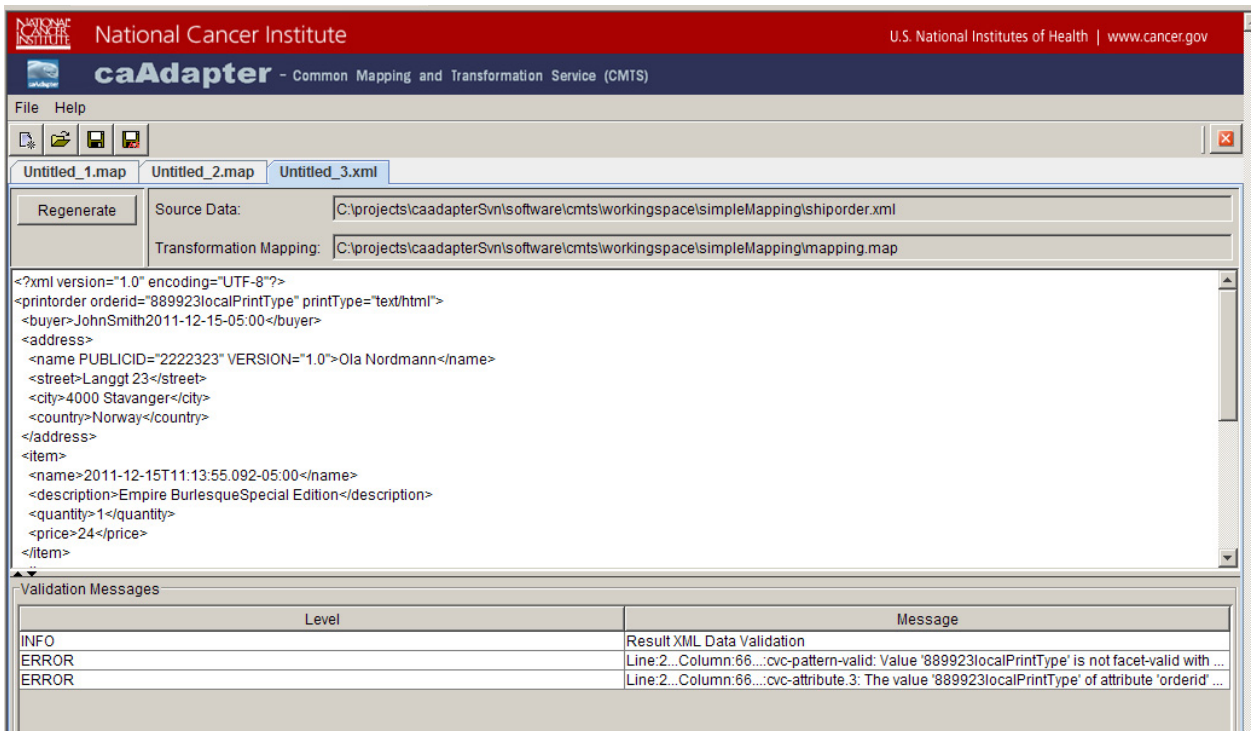


Figure 3.5 XML to XML transformation Result

The transformation result panel is a vertical split pane. The top scroll panel displays result XML data and the low scroll panel displays validation messages.

Generate XQuery Artifact

CMTS transformation supports bi-directional compatibility with any XQuery 1.0 compatible data transformation tools. Users are able to use this feature in the following scenarios:

- CMTS generate XQuery to be used by other XQuery 1.0 compatible data transformation tool
- CMTS perform data transformation using XQuery artifact generated by other XQuery 1.0 compatible data transformation tools. This scenario is included in XML to XML transformation section.

To generate XQuery artifact, perform the following steps:

1. Select **File > New > XQuery Artifact** from menu bar.

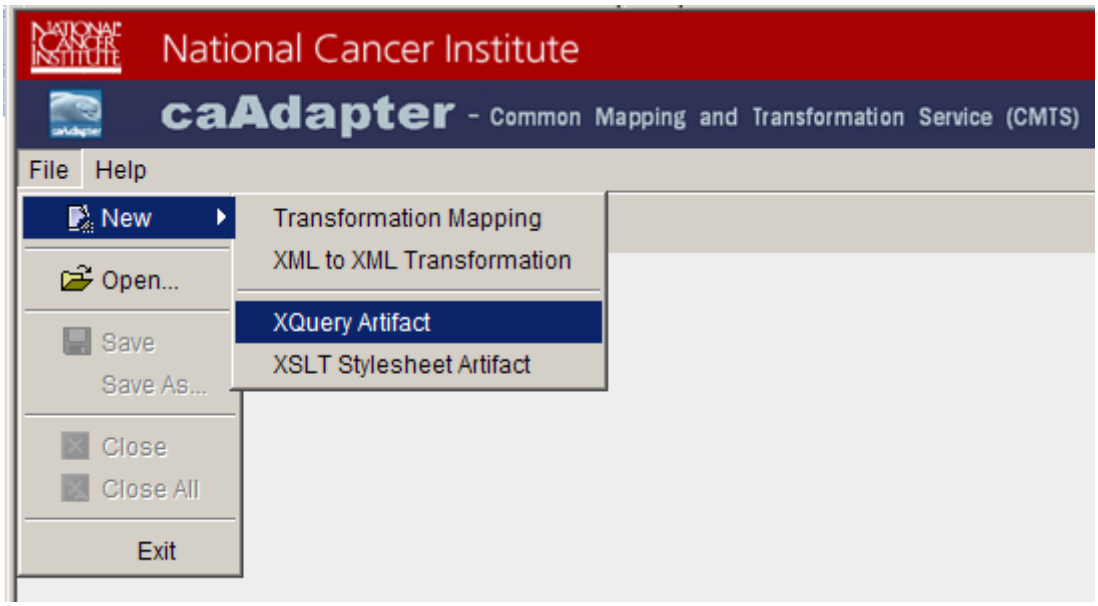


Figure 3.6 Generate XQuery Artifact

2. XQuery Artifact dialog appears.

Figure 3.7 XQuery Artifact Inputs Selection

3. Click **Browser** button next to ***Transformation Instruction*** box
4. ***Open Transformation Instruction*** dialog appears.

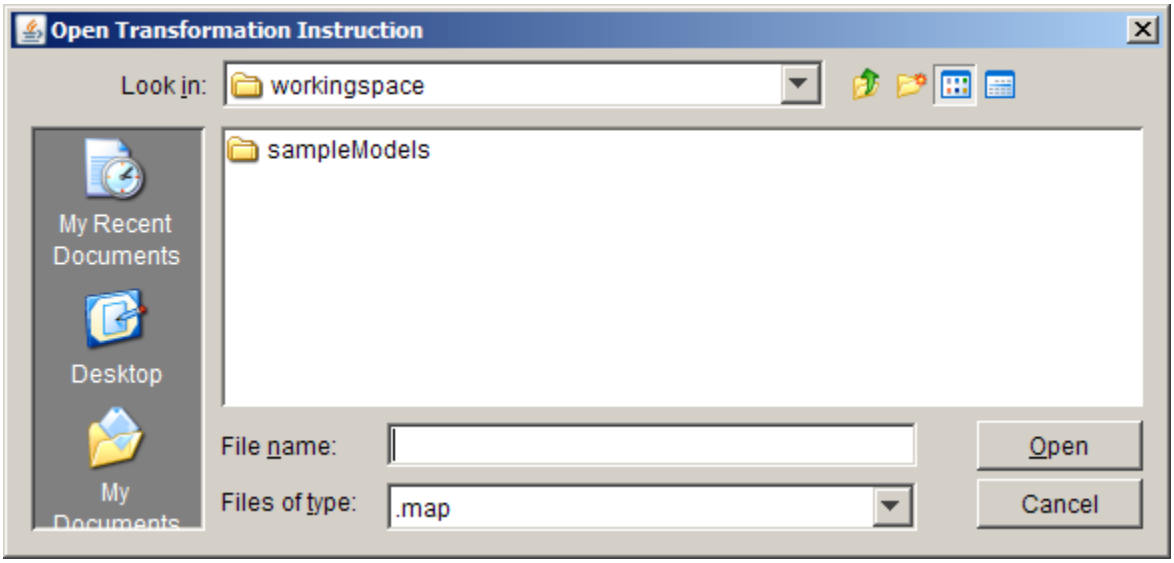


Figure 3.8 Open Transformation Instruction for XQuery Artifact

5. Choose XML to XML transformation mapping file and click ***Open*** button

6. Transformation engine generates XQuery artifact and display it on result panel.

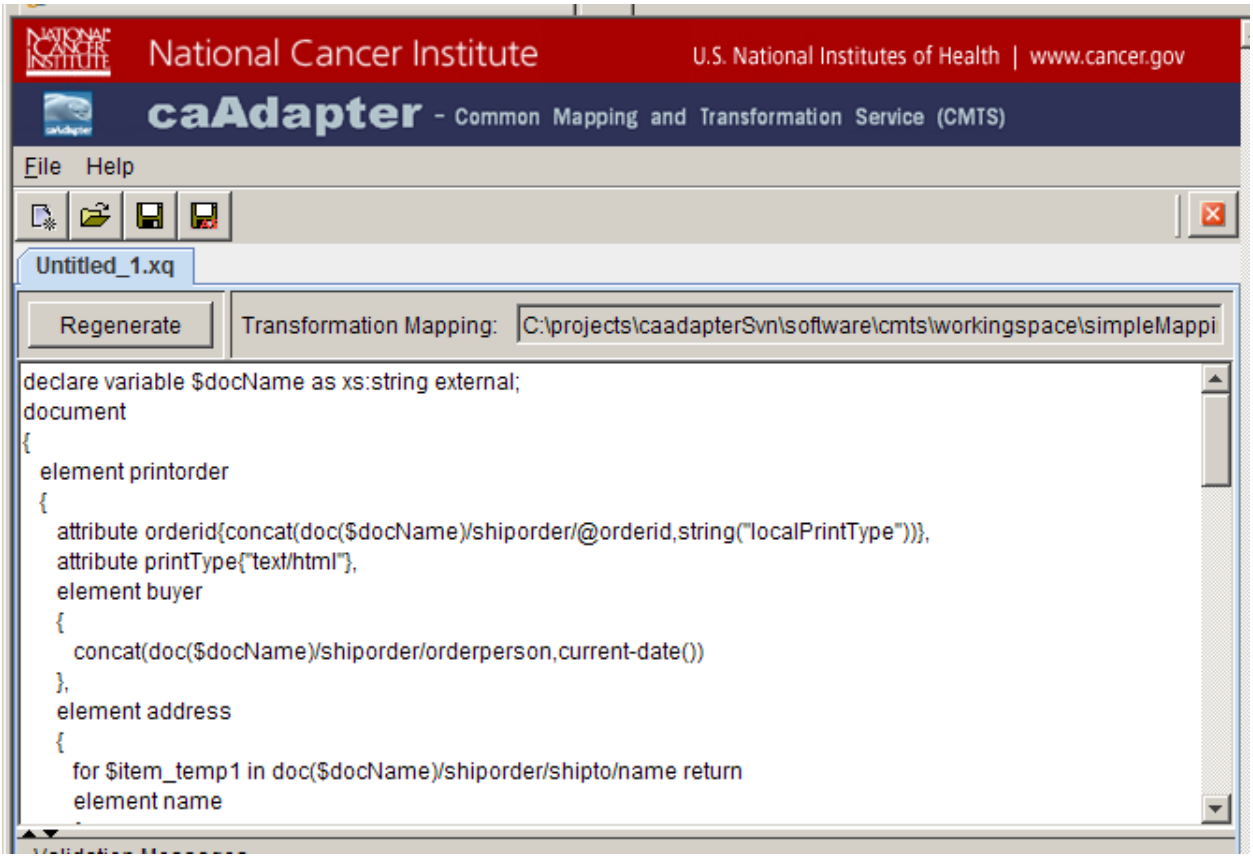


Figure 3.9 XQuery Artifact Result

Generate XQuery Artifact

CMTS transformation supports bi-directional compatibility with any XSLT 2.0 compatible data transformation tools. Users are able to use this feature in the following scenarios:

- CMTS generate XSLT stylesheet to be used by other XSLT 2.0 compatible data transformation tool
- CMTS perform data transformation using XSLT stylesheet generated by other XSLT 2.0 compatible data transformation tools. This scenario is included in XML to XML transformation section.

To generate XQuery artifact, perform the following steps:

1. Select **File > New > XSLT Stylesheet Artifact** from menu bar.

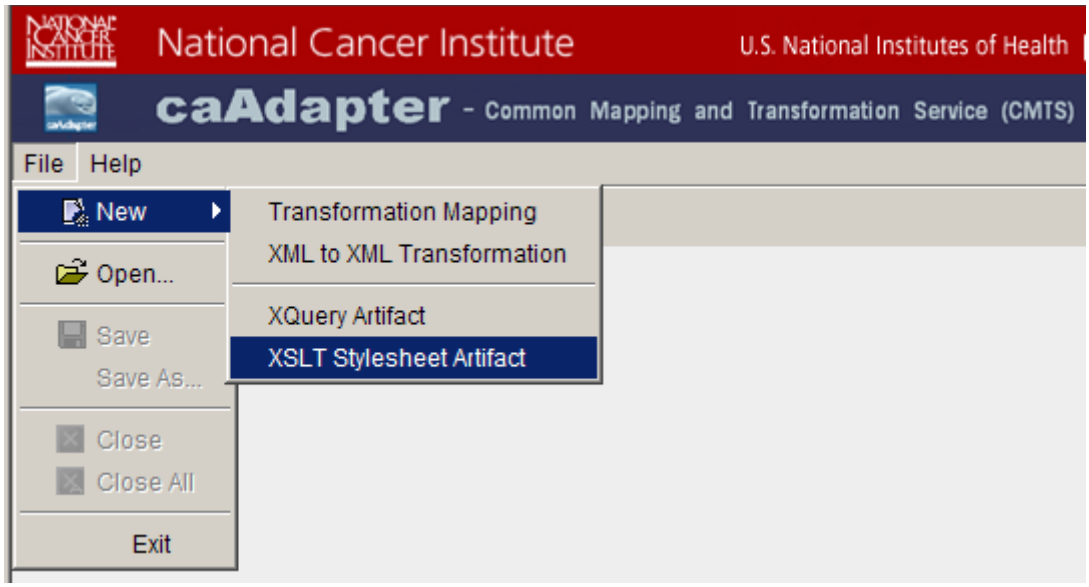


Figure 3.10 Generate XSLT Stylesheet Artifact

2. XSLT Stylesheet Artifact dialog appears.

Figure 3.11 XSLT Stylesheet Artifact Inputs Selection

3. Click **Browser** button next to ***Transformation Instruction*** box
4. ***Open Transformation Instruction*** dialog appears.

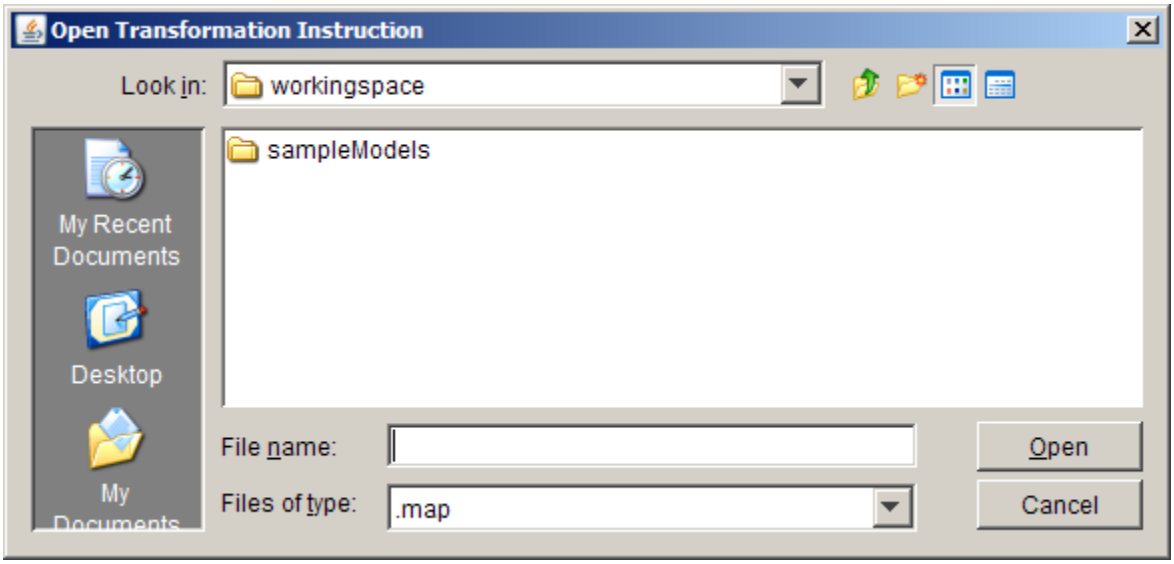


Figure 3.12 Open Transformation Instruction for XSLT stylesheet Artifact

5. Choose XML to XML transformation mapping file and click ***Open*** button
6. Transformation engine generates XQuery artifact and display it on result panel.

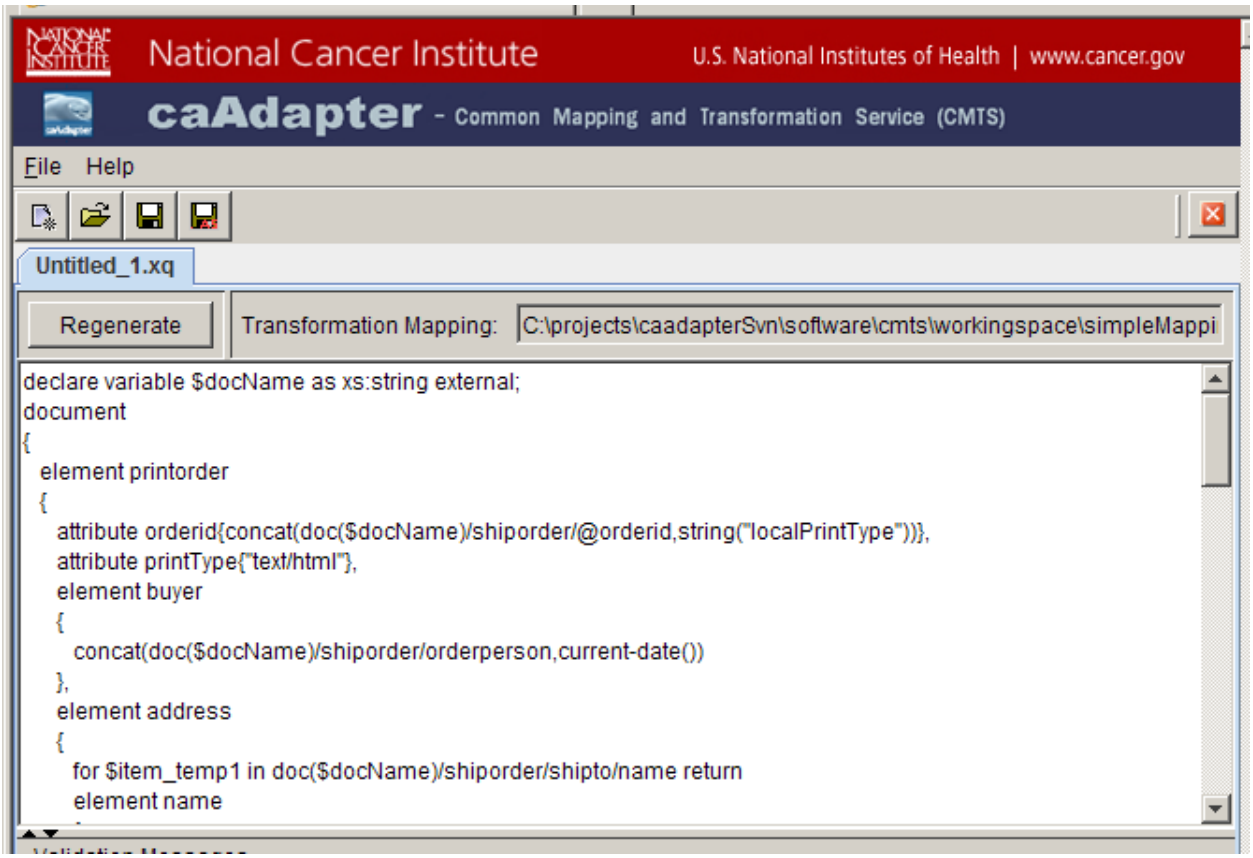
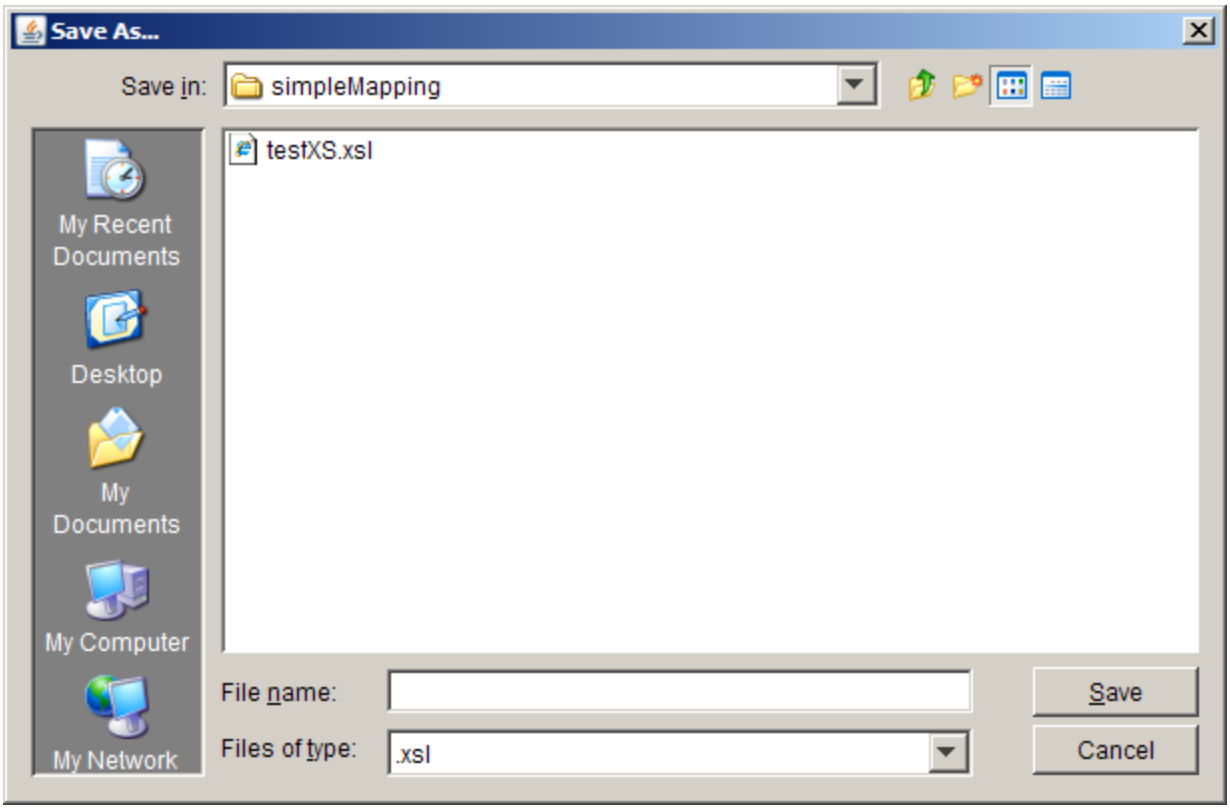


Figure 3.13 XSLT Stylesheet Artifact Result

Save Transformation Result

Once the transformation process is complete, the result panel displays transformation result (XML, XQuery artifact, XSLT artifact). To save the transformation result, perform the following steps:

1. Select **File > Save** from menu bar or **Save** icon on tool bar
2. A file chooser dialog appears.
3. Alternative, to save the saved data to different file, perform the following steps.
4. Select **File > Save As** on menu bar or **Save As** icon on tool bar.



5. Choose or input file name, the transformation result is saved to the chosen file with CMTS default file extension. The file extension is chosen as following:
    a. XML data -- .xml
    b. XQuery artifact -- .xql
    c. XSLT stylesheet artifact -- .xsl

Chapter IV Programming APIs

The programming APIs defines the program interfaces which enable CMTS transformation engine as a plug-in unit with user's application. This plug-in unit transfers source data to target data guided by transformation instructions. The transformation instruction can be in one of the following formats:

- XML to XML transformation mapping
- XQuery artifact
- XSLT stylesheet artifact.

Transformation API

The following JavaDoc section describes "transfer" operation, its parameters, and return data type.

---

transfer

```
String transfer(String sourceFile,
```

String processInstruction)

Transfer source data into target data using process instruction file

Parameters:

`sourceFile` - URI of source data file

`processInstruction` - URI of transformation file, such as, mapping, XQuery artifact, or XSLT style sheet artfact

Returns:

Result XML data

Programming Sample

Given the source data file: source and transformation process instruction file: instruction, perform the following programming steps to transfer the source data to target data:

1. Decide transformation type: map, xql, or xls
   String transformationType=”map”;
2. Get transformer from TransformationFactory
   TransformationService transformer =
   TransformerFactory.getTransformer(transformationType);
3. Set output file:result.xml
   FileWriter sWriter = new FileWriter(new File(args[2]));
4. Invoke transformation and write result to output file
   sWriter.write(transformer.transfer(args[0],args[1]));
   sWriter.flush();
   sWriter.close();

Chapter V Web Service APIs

CMT Web service APIs includes two parts: SOAP based traditional Web service access APIs and RESTful compliant Web service APIs. The definition of the SOAP based Web service APIs is detailed in Appendix A with Web Services Description Language (WSDL). It interacts with client in a manner prescribed by its description using SOAP messages. The definition of RESTful compliant Web service APIs is detailed in Appendix B with Web Application Description Language (WADL). It manipulates CMTS transformation functionalities using a uniform set of “stateless” operation. User can access the RESTful compliant service access point either programmatically or by launch Web browser request.

CMTS provides Web Services Management Portal to manage mapping scenarios, including browsing existing scenarios and creating new scenarios. The two sets of Web services use the registered mapping scenarios to process user’s data transformation request.

To use Web service APIs, perform the following steps:

- Create mapping scenario.
- Register mapping scenario.

- Invoke transformation service using scenario and source data.

Register Mapping Scenario to Web Service Management Portal

After creating a transformation mapping with source data schema (xsd) and target data schema (xsd), perform the following step to register it to CMTS Web Services Management Portal:

1. In Internet Explorer or Firefox, navigate to http://caadapter.nci.nih.gov/caadapterWS-cmts/

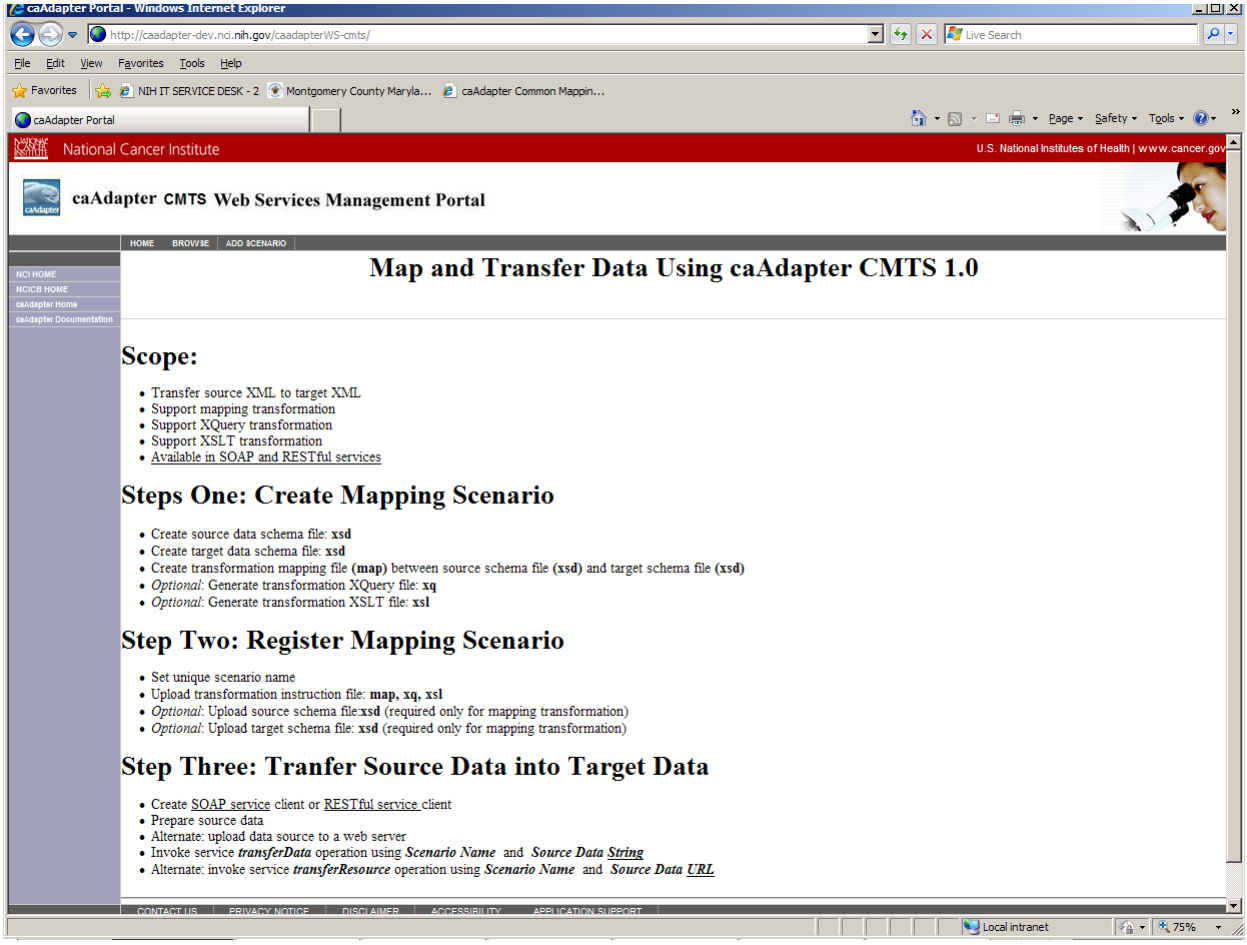


Figure 5.1 Web Services Management Portal Home Page

2. At the top of the page, click **Brows** to explore registered mapping scenarios.

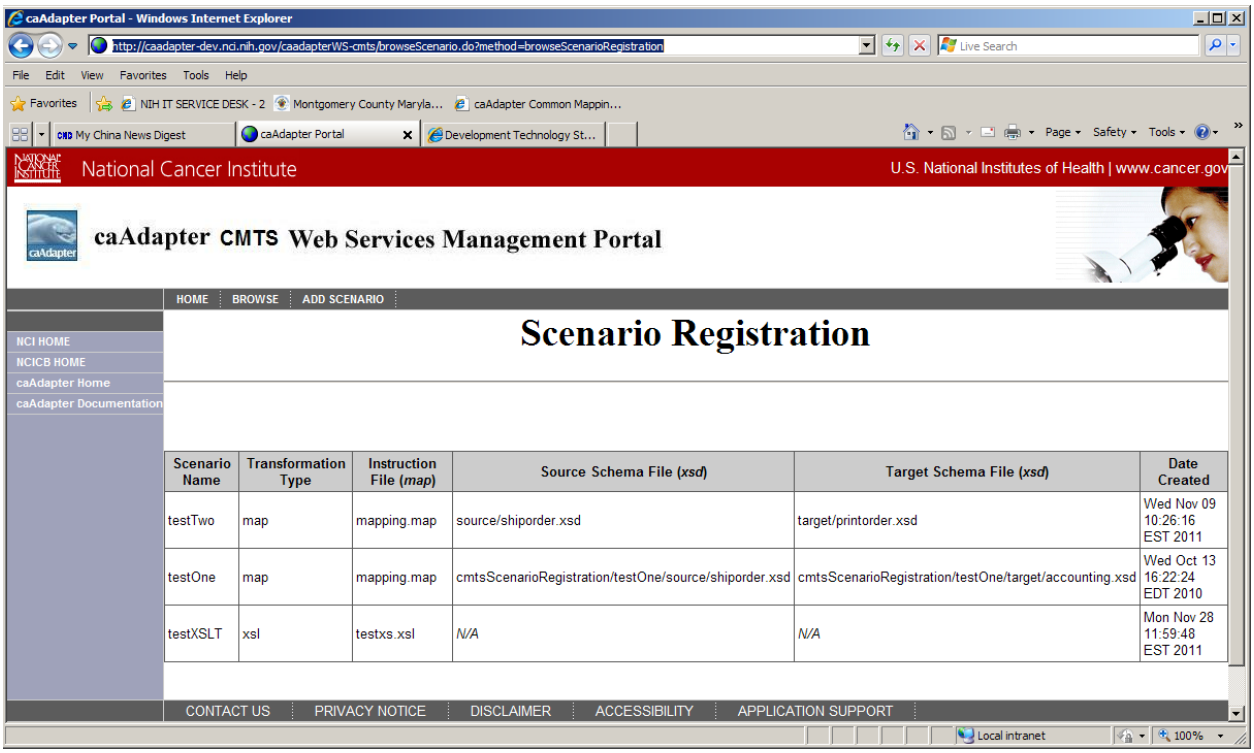


Figure 5.2 Browse Mapping Scenarios

3. At the top of the page, click **Add Scenario** to add a new mapping scenario

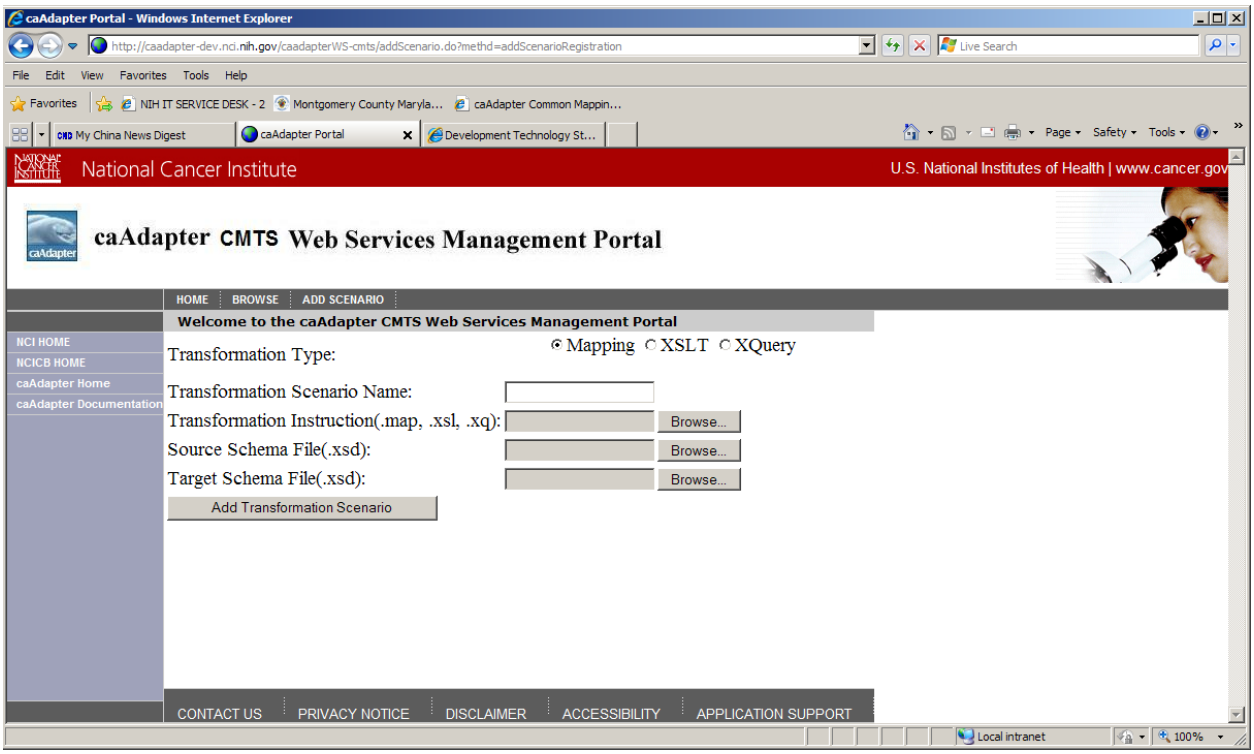


Figure 5.3 Add Mapping Scenario

4. Select mapping scenario type: mapping, XQuery, or XSLT

5. Specify the unique name for the mapping scenario to register. (Note: Use this name in the later web services clients.)
6. In the Transformation Instruction file field, click **Browse** and navigate to the corresponding instruction file, which has a suffix of `.map, .xql, or .xsl`
7. Only if the mapping scenario type is **"mapping"**, in the Source Schema file field, click **Browse** and navigate to source schema file, which has a suffix of `.xsd`.
8. Only if the mapping scenario type is **"mapping"**, in the Target Schema file field, click **Browse** and navigate to target schema file, which has a suffix of `.xsd`.
9. Click **Add Mapping Scenario**. Once caAdapter CMTS creates the mapping scenario based on your files, it sends user confirmation message.

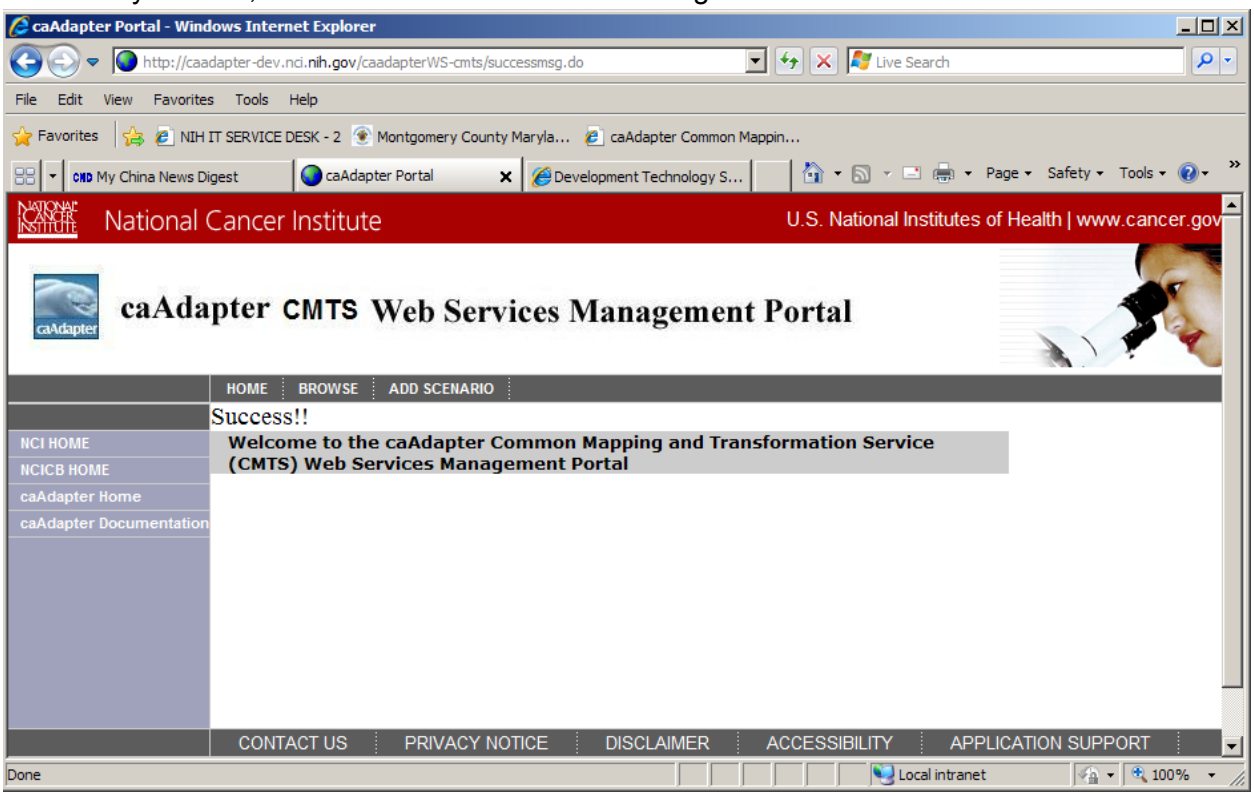


Figure 5.4 Confirm Mapping Scenario Registration

Access SOAP Web Service

SOAP web service APIs includes two transformation operations: transferData and transferResource. JavaDoc style of description of these functions follow:

transferData

```
ArrayList<String> transferData(String mappingScenario,
                               String sourceData)
```

Transfer a data string into target datasets

Parameters:

`mappingScenario` - The name of the mapping scenario, which is registered with CMTS Web Services Management Portal.

`sourceData` - source data in XML format as a string

Returns:
A collection of the transformed XML datasets

---

transferResource
`ArrayList<String> transferResource(String mappingScenario, String sourceResource)`

Transfer a URL data resource into target datasets

Parameters:

`mappingScenario` - The name of the mapping scenario, which is registered with CMTS Web Services Management Portal.

`sourceResource` - URL of source data in XML format

Returns:

A collection of the transformed XML datasets

CMTS SOAP Web Service APIs are implemented the style of Remote Procedure Calls (RPC) which presents CMST transformation operations as distributed function call interface. Users are able to access CMTS SOAP Web Service APIs with any RPC style Web service client program. Appendix A details these service APIs with Web Service Description Language (WSDL). Online version of WSDL file is available at link:
http://caadapter.nci.nih.gov/caadapterWS-cmts/services/transfer?wsdl

Axis 1.x RPC Style Access SOAP Web Service

Perform the following steps to build RPC style client program:

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-cmts/services/transfer?wsdl`
5. Use the following code to access the web services
   ```
   import java.util.*;
   import gov.nih.nci.caadapter.cmts.ws.transformationService
   ```

```
.*;
public class AxisRPCClient {
public static void main(String[] args) {
try {
      String sourceData= "… …";
      cmtsTransformationServiceService service = new
      cmtsTransformationServiceServiceLocator();
      cmtsTransformationService cmtsService
      service.getTransformationService();
      Object[] res = (Object[])cmtsService.transferData("scenario",
      sourceData);
      //or call the other operation
      // Object[] res =
      (Object[])cmtsService.transferResource("scenario", //sourceData);

      for(int i=0;i<res.length;i++)
            System.out.println((String)res[i]);
      }catch(Exception e)
      {
            e.printStackTrace();
      }
}
Note:
scenario – name of the mapping scenario registered with CMTS Web Services
management Portal
sourceData- If "transferData" is called, "sourceData" refers to the path
source file; If "transferResource" is called, "sourceData" refers to
source data URL.
```

Axis 1.x DII Style Access SOAP Web Service

Perform the following steps to build Dynamic Invocation Interface (DII) style client program:

1. Download Axis 1.x (`axis-bin-1_4.zip`) from the following URL: http://
2. www.apache.org/dyn/closer.cgi/ws/axis/1_4
3. Unzip `axis-bin-1_4.zip` file.
4. 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`
5. Use the following code to access the 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-
      cmts/services/transfer";
      String operationName = "... ...";
      String sourcData = "... ...";
      Service service = new Service();
      Call call = (Call)service.createCall();
      call.setTargetEndpointAddress(new
      java.net.URL(endpointURL));
      call.setOperationName(operationName);
      call.addParameter("arg0",XMLType.XSD_STRING,
      ParameterMode.IN );
      call.addParameter("arg1",XMLType.XSD_STRING,
      ParameterMode.IN );
      call.setReturnClass(java.util.ArrayList.class);
      ArrayList res = (ArrayList)call.invoke(
      new Object[]{"My_WS_Scenario", sourcData});
      System.out.println(res);

      }catch(Exception e)
      {
            e.printStackTrace();
      }
}
```

Note:

operationName – name of remote operation: transferData or transferResource

scenario – name of the mapping scenario registered with CMTS Web Services management Portal

sourceData - If "transferData" is called, "sourceData" refers to the path source file; If "transferResource" is called, "sourceData" refers to source data URL.

Access RESTful Web Service

Similar to SOAP web service APIs, RESTful service APIs includes two transformation operations: transferData and transferResource. JavaDoc style of description of these functions follow:

restfulTransferData

```
ResultList restfulTransferData(String mappingScenario,
                               String sourceData)
```

RESTfull API: Transfer a data string into target datasets

Parameters:

`mappingScenario` - The name of the mapping scenario, which is registered with CMTS Web Services Management Portal.

`sourceData` - source data in XML format as a string

Returns:

A collection of the transformed XML datasets wrapped as JAXB compliant format

---

restfulTransferResource

```
ResultList restfulTransferResource(String mappingScenario,
                                   String sourceURL)
```

RESTfull API: Transfer a URL data resource into target datasets

Parameters:

`mappingScenario` - The name of the mapping scenario, which is registered with CMTS Web Services Management Portal.

`sourceURL` - URL of source data in XML format

Returns:

A collection of the transformed XML datasets wrapped as JAXB compliant format

RESTful Web Service APIs are implemented one the base of HTTP. RESTful service server maps each operation to an HTTP URL and parameter to query parameter.

- restfulTransferData -- /transferData
  - mappingScenario-- scenario
  - sourceData -- source
- restfulTransferResource – transferResource
  - mappingScenario-- scenario
  - sourceData -- source

Users are able to access these APIs with any RESTful style client program or any Web browser launching a client request. Appendix B details these service APIs with Web Application Description Language (WADL). Online version of WADL file is available at link:

http://caadapter.nci.nih.gov/caadapterWS-cmts/services/restful?_wadl&_type=xml

Apache CXF Client Access RESTful Service

Perform the following steps to build Apache CXF client program:

Download Apache CXF 2.5 (`apache-cxf-2.5.0.zip`) from the following URL: http://cxf.apache.org/download.html

Unzip `apache-cxf-2.5.0.zip` file.

Add the `cxf-2.5.0.jar` to your classpath.

Use the following code to access RESTful web services

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLDecoder;
import java.net.URLEncoder;

import org.apache.cxf.helpers.IOUtils;
import org.apache.cxf.io.CachedOutputStream;
public class ApacheCxfClient {
public static void main(String args[]) throws Exception {
   // Sent HTTP GET request to query data
   if (args.length<4)
   {
        System.out.println("Client Usage: [scenarioName]|
        [operationName]|[sourceData]|[serviceURL]");
   return;
   }

      //read WS paramters
    String scenarioName= args[0];
      String srcURI =args[1];
      String operationName=args[2];
      String serviceURL =args[3];

      if (operationName.equals("transferData"))
      {
           String srcData =buildSourceDataString(srcURI);
        srcURI =URLEncoder.encode(srcData, "UTF-8");
      }
String querySt="scenario="+scenarioName+"&source="+srcURL;
     String targetUrl= serviceURL +"/"+operationName;
     String urlSt=targetUrl+"?"+querySt;//

     URL url = new URL(urlSt);
     InputStream in = url.openStream();
```

```
        String rtnString=getStringFromInputStream(in);
        System.out.println("decoded return:\n" +
        URLDecoder.decode(rtnString, "UTF-8"));
           System.exit(0);
       }
}
```

Note:
scenario – name of the mapping scenario registered with CMTS Web Services management Portal
operationName – name of remote operation: transferData or transferResource
sourceData - If "transferData" is called, "sourceData" refers to the path source file; If "transferResource" is called, "sourceData" refers to source data URL.
serviceURL -- http://caadapter.nci.nih.gov/caadapterWS-cmts/services/restful

Web Browser Access RESTful Service

From any Web browser, users are able to invoke RESTful service APIs as following:
http://caadapter.nci.nih.gov/caadapterWS-cmts/services/restful/transferData?scenario={scenario}&source={encoded source XML data}
or
http://caadapter.nci.nih.gov/caadapterWS-cmts/services/restful/transferResource?scenario={scenario}&source={source data URL}

Note:
scenario – name of the mapping scenario registered with CMTS Web Services management Portal
source - If "transferData" is called, "source" refers to the path source file; If "transferResource" is called, "source" refers to source data URL.

Appendix A CMTS Web Service Definition Language (WSDL)

The following WSDL define the web service access API for CMTS data transformation portal. It describes the web services as a set of endpoints operation on messages. It uses the following elements in the definition:

- **Types**– a container for data type definitions using XSD.
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.

- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions name="TransformationServiceImplService"
        targetNamespace="http://ws.cmts.cbiit.nci.nih.gov/"
        xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://ws.cmts.cbiit.nci.nih.gov/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <wsdl:types>
                <xsd:schema attributeFormDefault="unqualified" elementFormDefault="unqualified"
                        targetNamespace="http://ws.cmts.cbiit.nci.nih.gov/"
                        xmlns:tns="http://ws.cmts.cbiit.nci.nih.gov/"
                        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                  <xsd:element name="transferResource" type="tns:transferResource"/>
                  <xsd:complexType name="transferResource">
                          <xsd:sequence>
                                  <xsd:element minOccurs="0" name="arg0" type="xsd:string"/>
                                  <xsd:element minOccurs="0" name="arg1" type="xsd:string"/>
                          </xsd:sequence>
                  </xsd:complexType>
                  <xsd:element name="transferResourceResponse"
                                  type="tns:transferResourceResponse"/>
                  <xsd:complexType name="transferResourceResponse">
                          <xsd:sequence>
                                  <xsd:element maxOccurs="unbounded"
                                          minOccurs="0" name="return" type="xsd:string"/>
                          </xsd:sequence>
                  </xsd:complexType>
                  <xsd:element name="transferData" type="tns:transferData"/>
                  <xsd:complexType name="transferData">
                          <xsd:sequence>
                                  <xsd:element minOccurs="0" name="arg0" type="xsd:string"/>
                                  <xsd:element minOccurs="0" name="arg1" type="xsd:string"/>
                          </xsd:sequence>
                  </xsd:complexType>
                  <xsd:element name="transferDataResponse"
                                  type="tns:transferDataResponse"/>
                  <xsd:complexType name="transferDataResponse">
                          <xsd:sequence>
```

```
                        <xsd:element maxOccurs="unbounded" minOccurs="0"
                                name="return" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:schema>
</wsdl:types>
<wsdl:message name="transferResourceResponse">
<wsdl:part element="tns:transferResourceResponse" name="parameters">
</wsdl:part>
</wsdl:message>
<wsdl:message name="transferDataResponse">
  <wsdl:part element="tns:transferDataResponse" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="transferResource">
  <wsdl:part element="tns:transferResource" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="transferData">
  <wsdl:part element="tns:transferData" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:portType name="TransformationWebService">
<wsdl:operation name="transferResource">
            <wsdl:input message="tns:transferResource" name="transferResource">
            </wsdl:input>
            <wsdl:output message="tns:transferResourceResponse"
                    name="transferResourceResponse">
            </wsdl:output>
</wsdl:operation>
<wsdl:operation name="transferData">
            <wsdl:input message="tns:transferData" name="transferData">
            </wsdl:input>
            <wsdl:output message="tns:transferDataResponse"
                    name="transferDataResponse">
            </wsdl:output>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="TransformationServiceImplServiceSoapBinding"
            type="tns:TransformationWebService">
<soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="transferResource">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name="transferResource">
            <soap:body use="literal"/>
```

```
            </wsdl:input>
            <wsdl:output name="transferResourceResponse">
            <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="transferData">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name="transferData">
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="transferDataResponse">
            <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="TransformationServiceImplService">
        <wsdl:port binding="tns:TransformationServiceImplServiceSoapBinding"
            name="TransformationServiceImplPort">
            <soap:address
            location="http://caadapter.nci.nih.gov/caadapterWS-cmts/services/transfer"/>
        </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Appendix B CMTS RESTful Web Application Description Language (WADL)

The following WADL describe the RESTful service access API for CMTS data transformation portal. It describes each allowable service with a set of ***resource*** element. Each of these contains ***param*** elements to describe the inputs, and ***method*** elements which describe the ***request*** and ***response*** of a resource. The ***request*** element specific how to represent the input, what types are required and any HTTP headers that are required. The ***response*** describes the representation of the service's response, as well as any fault information, to deal with errors.

```
<application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <grammars>
            <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
      elementFormDefault="unqualified">
            <xs:element name="ReulstData" type="resultList"/>
            <xs:complexType name="resultList">
              <xs:sequence>
                <xs:element maxOccurs="unbounded" name="result" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
            </xs:schema>
        </grammars>
        <resources base="http://caadapter.nci.nih.gov/caadapterWS-cmts/services/restful">
```

```
                <resource path="/">
                        <resource path="transferData">
                                <method name="GET">
                                        <request>
                                                <param name="scenario" style="query" type="xs:string"/>
                                                <param name="source" style="query" type="xs:string"/>
                                        </request>
                                        <response>
                                                <representation mediaType="text/xml"/>
                                                <representation mediaType="application/xml"/>
                                        </response>
                                </method>
                        </resource>
                        <resource path="transferResource">
                                <method name="GET">
                                        <request>
                                                <param name="scenario" style="query" type="xs:string"/>
                                                <param name="source" style="query" type="xs:string"/>
                                        </request>
                                        <response>
                                                <representation mediaType="text/xml"/>
                                                <representation mediaType="application/xml"/>
                                        </response>
                                </method>
                        </resource>
                </resource>
        </resources>
</application>
```