# caAdapter Common Mapping and Transformation Service v1.0 User's Guide

## caAdapter Common Mapping and Transformation Service v1.0 User's Guide

> ✅ **You can convert the wiki version of this guide to PDF for viewing and printing**
> For instructions refer to How do I print or export multiple pages? If you want to print a single page, refer to How do I print a page?.
> Some longer links may appear truncated when viewed in PDFs, but they work regardless.

This guide contains the following topics.

- About this Guide v1.0
- 1 - Getting Started with caAdapter CMTS v1.0
- 2 - Transformation Mapping v1.0
- 3 - Data Transformation v1.0
- 4 - Programming APIs v1.0
- 5 - Web Service APIs v1.0
- A - CMTS Web Service Definition Language
- B - CMTS RESTful Web Application Description Language

# About this Guide v1.0

## About this Guide v1.0

This section introduces you to the *caAdapter Common Mapping and Transformation Service 1.0 User's Guide*.

Topics in this guide include:

- Purpose
- Release Schedule
- Audience
- Prerequisites
- Topics Covered
- Recommended Reading
- Credits and Resources
- NCI CBIIT Application Support
- LISTSERV facilities pertinent to caAdapter

### 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).

### Release Schedule

This guide may be updated between releases if errors or omissions are found. The current document refers to the 1.0 version of caAdapter, released in April 2012 by the NCI Center for Biomedical Informatics and Information Technology (CBIIT).

### Audience

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 the schema (XSD); and
- Analysts (medical analysts, database administrators, and business analysts) who need step-by-step procedures for creating XML message instances.

## Prerequisites

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

Use of the CMTS module requires additional prerequisites. For more information, see Prerequisites for Using caAdapter CMTS.

## Topics Covered

This guide includes the following sections.

- 1 - Getting Started with caAdapter CMTS v1.0, discusses CMTS architecture, functionalities, and implementation technologies.
- 2 - Transformation Mapping v1.0, explains the procedure to work with XML to XML transformation mapping.
- 3 - Data Transformation v1.0, explains the procedures to perform data transformation using CMTS graphic interface.
- 4 - Programming APIs v1.0, provides detailed instructions to perform data transformation using CMTS programming APIs.
- 5 - Web Service APIs v1.0, provides detailed instruction to perform data transformation using CMTS Web Service APIs.
- A - CMTS Web Service Definition Language, describes SOAP based web service access API for CMTS data transformation portal.
- B - CMTS RESTful Web Application Description Language, describes the 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 Biomedical Informatics Grid (caBIG) | https://cabig.nci.nih.gov/ |

## Credits and Resources

The following people contributed to the development of this document.

| Development | Documentation | Program Management |
| --- | --- | --- |
| Eugene Wang [2]<br>Ki Sung Um [2] | Carolyn Kelley Klinger [3] | Sichen Liu [1] |

[1] Center for Biomedical Informatics and Information Technology (CBIIT)
[2] Science Application International Corporation (SAIC)-Frederick, Inc.
[3] Independent Consultant

## NCI CBIIT Application Support

| Web Page | http://ncicb.nci.nih.gov/NCICB/support |
| --- | --- |
| Telephone | 301-451-4384<br>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 |

# 1 - Getting Started with caAdapter CMTS v1.0

## 1 - Getting Started with caAdapter CMTS v1.0

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

Topics in this guide include:

- About caAdapter CMTS
    - About XML Schema (XSD)
    - About XML Query
    - About XSLT
- Prerequisites for Using caAdapter CMTS
- Starting caAdapter CMTS
    - Starting caAdapter CMTS User's Deployment
    - Starting caAdapter CMTS Online
- caAdapter CMTS User Interface
    - File Menu
    - Help Menu
    - Tabs

### 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.

The CMTS module provides functionality for mapping and transforming from source XML data to target XML data. The mapping functionality creates mapping data, which is composed of links between nodes of source XML schema (XSD) and target XSD and data operation functions. In addition, it converts mapping data to transformation instructions for transformation tools compatible with XSLT or XQuery. The transformation functionality generates target XML data based on the target XSD from the source XML data using the mapping data, XSLT or XQuery

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 follows.
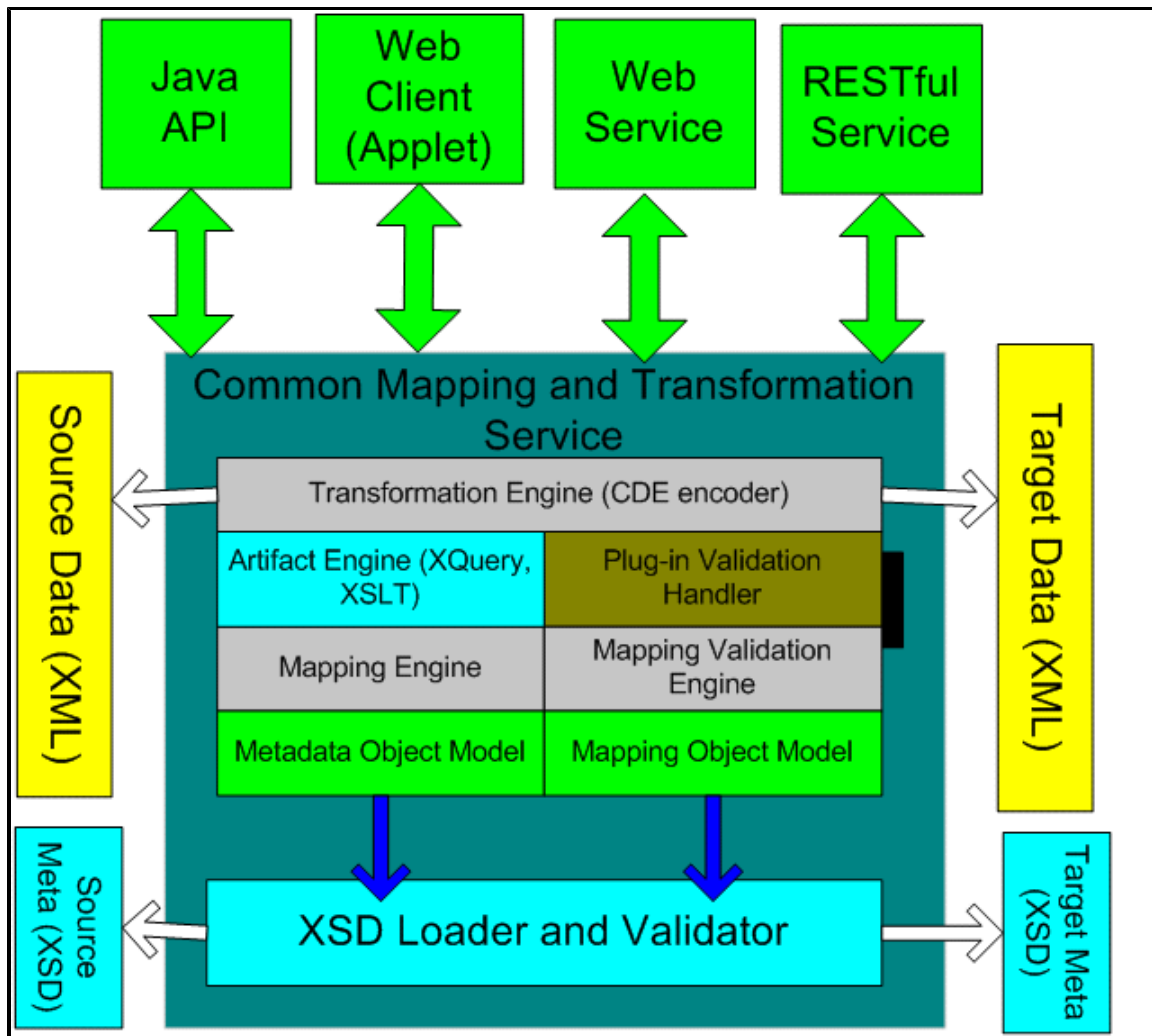
## About XML Schema (XSD)

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 with 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

Deploy CMTS application on your own server by following these steps:

1. Download `caAdapterCMTSv1.0_src_wo.zip` 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 the deployment folder: 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 caAdapter CMTS v1.0 Web GUI section on the caAdapter wiki home page.

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

## 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.
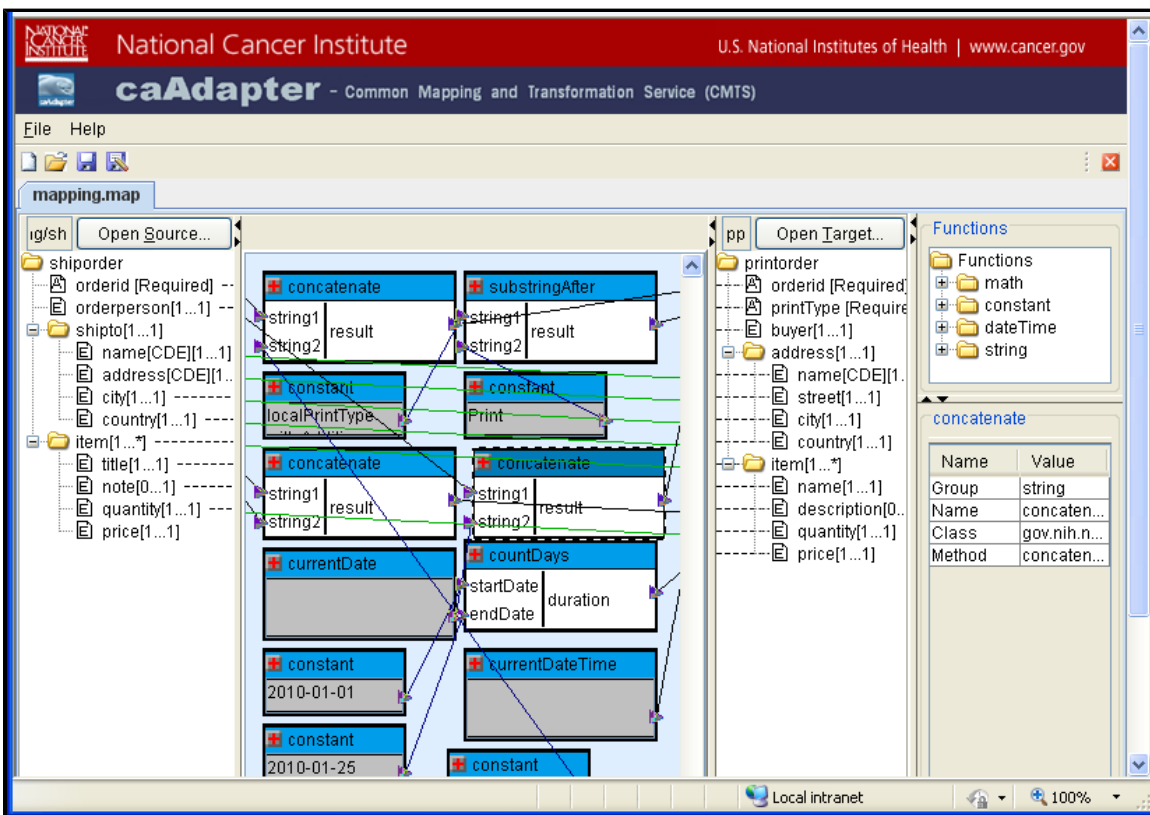


### File Menu

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: <br><br> • Transformation Mapping <br> • XML to XML Transformation <br> • XQuery Artifact <br> • XSLT Artifact |
|---|---|
| Open | Opens an existing transformation mapping file (.map) |
| Save | Saves the file you are currently working on (in the selected tab), including: <br><br> • Transformation Mapping <br> • Result XML Data <br> • XQuery Artifact <br> • 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 |

> ⛔ **Warning**
> On some web browsers, the 'Exit' menu may be deactivated according to the preset security policies.

### 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. The online help is part of the NCI Wiki. In the navigation panel on the left, you can search topics of interest within the caAdapter wiki space. Above the content panel on the right, you can search topics of interest across the entire NCI Wiki. |

### 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_<number>.<ext>`. The latter means unsaved content, where `<ext>` is the appropriate file extension for that type of tab.

# 2 - Transformation Mapping v1.0

## 2 - Transformation Mapping v1.0

This chapter explains how to create relationships between the source data schema, target schema, and data manipulation functions using

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 constrains the 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
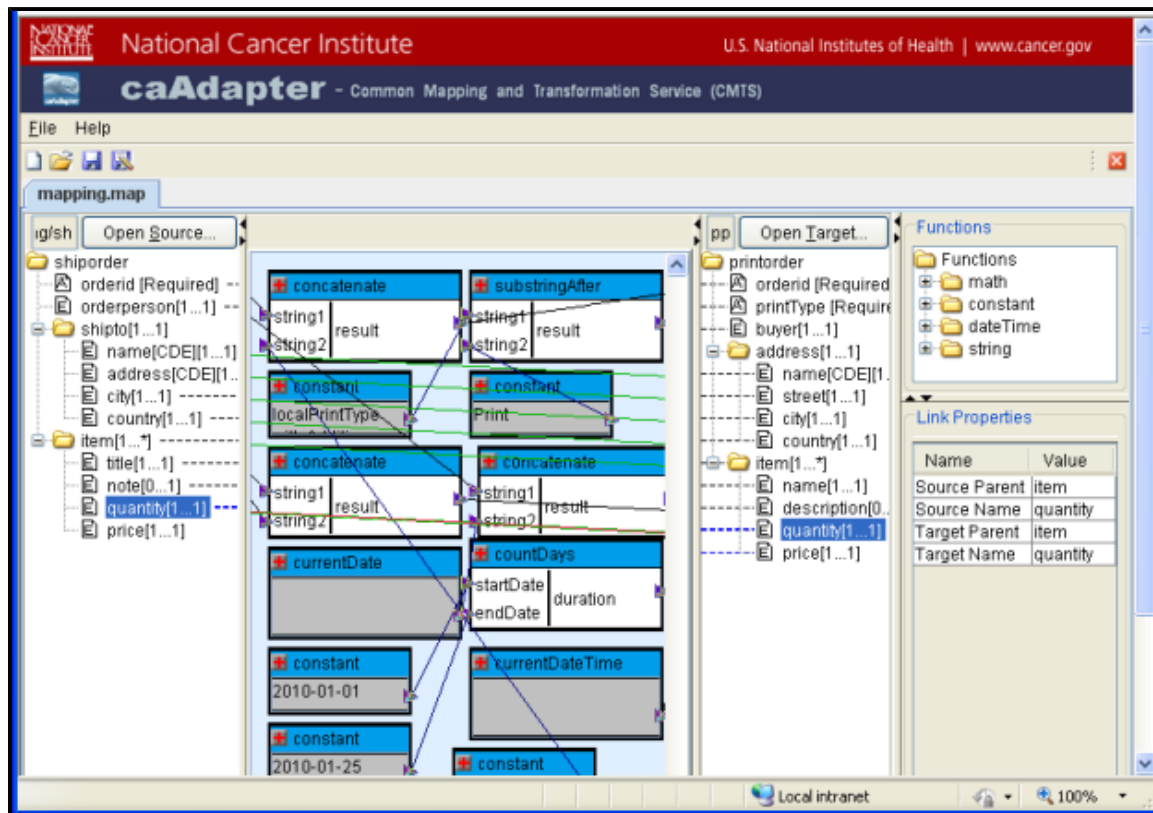
# Understand Transformation Mapping

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 a transformation mapping with source and target metadata, schema annotation, mapping links, function boxes, and other elements. It is divided into several sub-panels. An example follows.



The transformation mapping panel has the following components.

- 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.

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.

⚠ **Be Careful**
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.
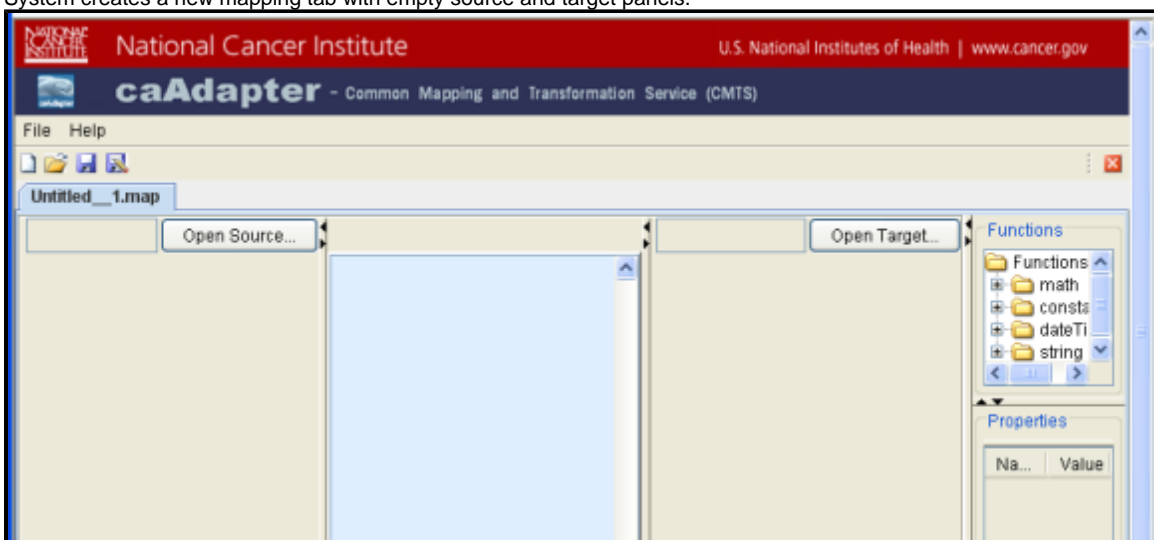
## Create New Transformation Mapping

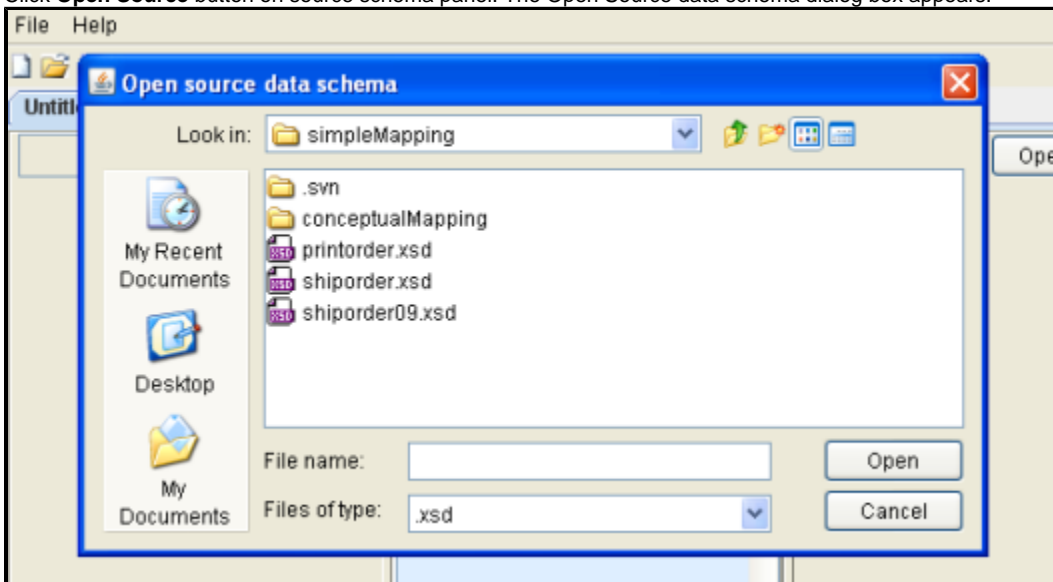To create a new transformation mapping, perform the following steps:

1. Select **File** > **New** > **Transformation Mapping**.
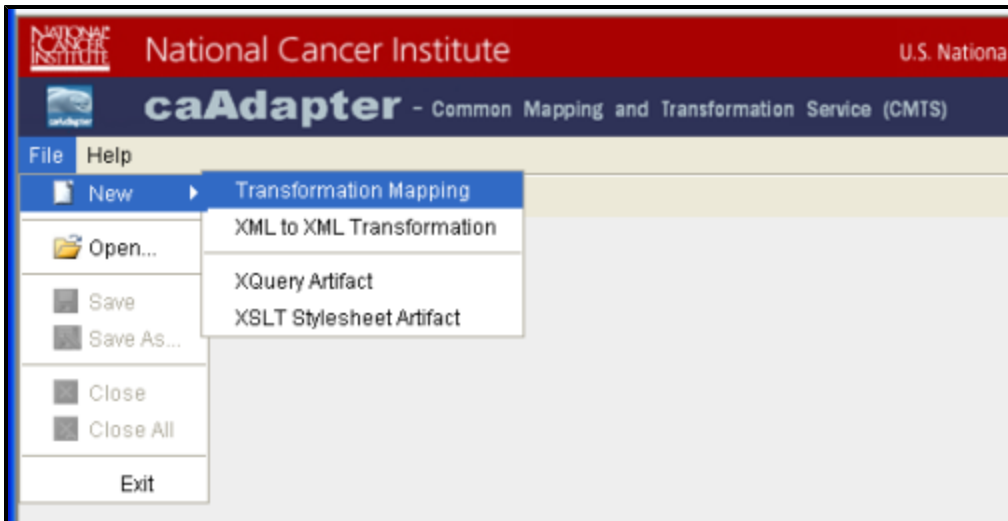
2. System creates a new mapping tab with empty source and target panels.



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



4. Select the source xsd file and click **Open** to populate the source Choose Root Element dialog box.
5. Select the root element of the XSD file. In general cases, the root element name can be seen just like the following figure, but if not, it must be searched among the element names that will be listed down after clicking on the circled button. Click **OK**.
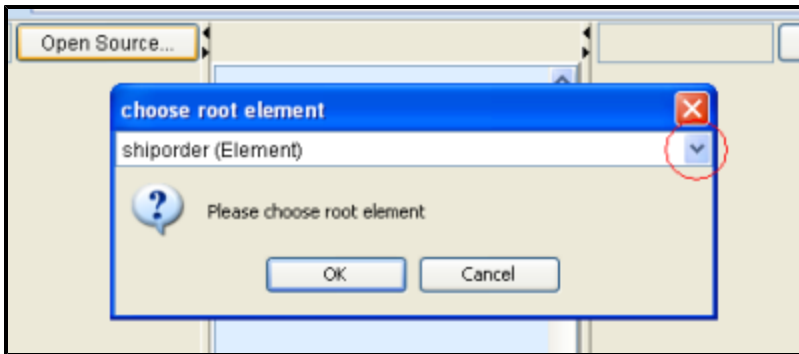
The parsed XSD Tree appears.
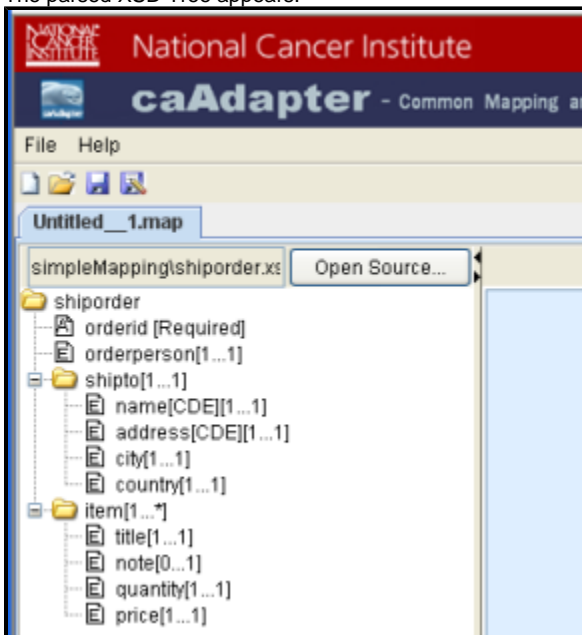


> ⚠ **Be Careful**
> In many cases, one schema links to others with '<include>' tags, though appearance is single file. If the system cannot found all of those linked schema file, the reading and parsing schema processing will be fail.

6. Click the **Open Target** button on target schema panel and the next procedures are the same as above 3. to 5. items.
   A mapping panel appears.
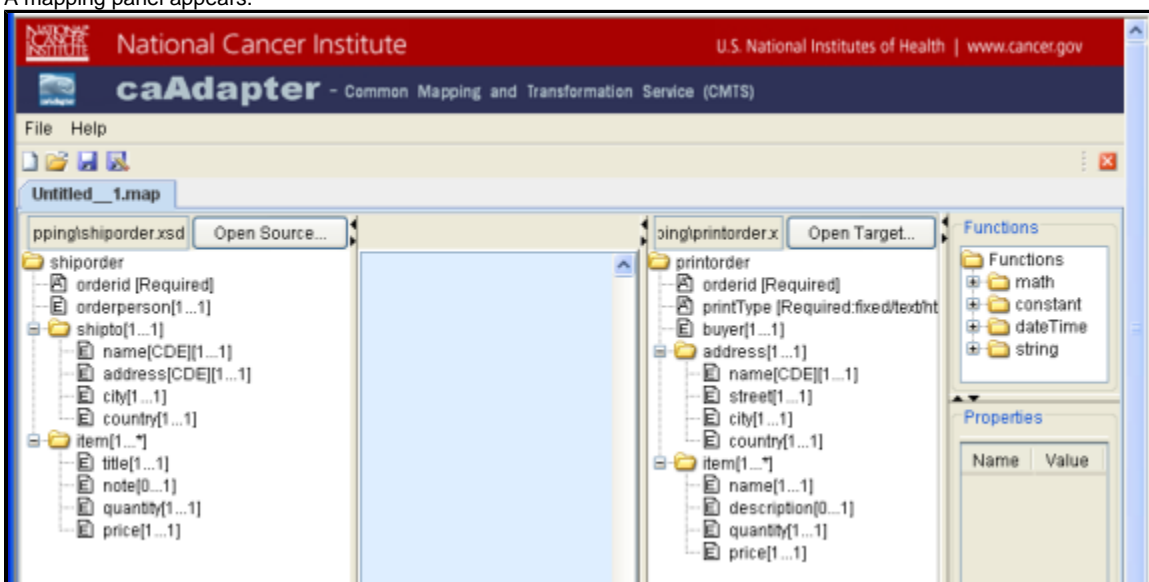
## Create Mapping Link

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 **shipto[1…1]** node on source schema tree and **address[1…1]** node on target schema tree.
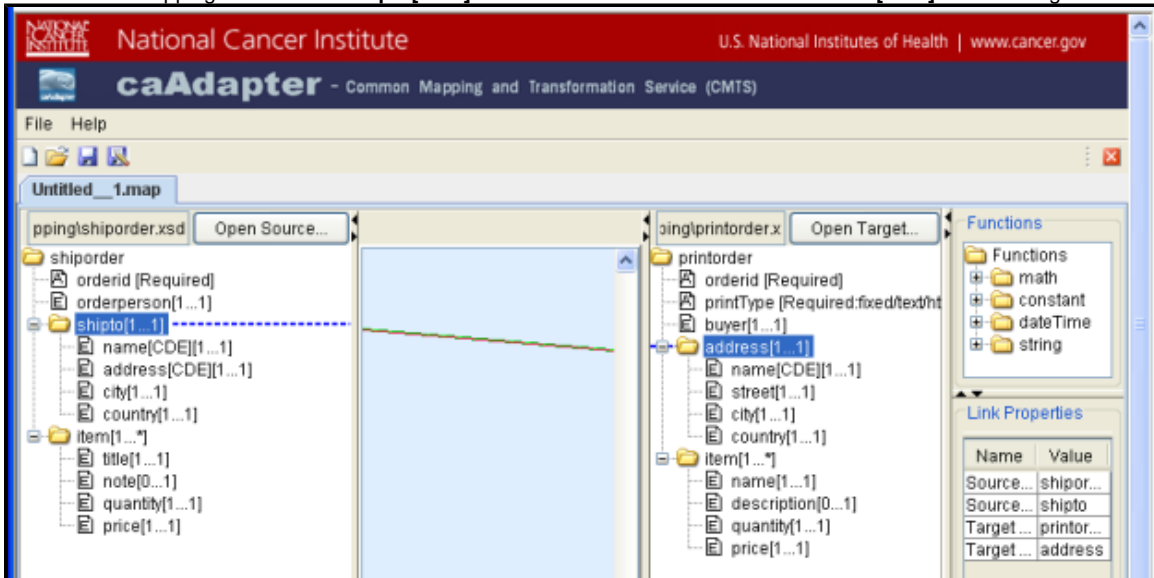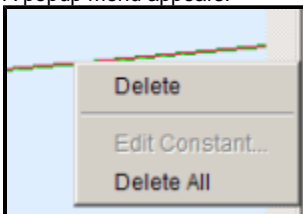


> ### Note
> When a mapping link is selected, the link and associated mapping ends are highlighted; the property panel displays the link's properties.

## 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.



2. Click **Delete** to delete only the selected mapping line.
3. Alternatively, click **Delete All** to delete all mapping links. A confirmation message box appears.
4. 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 the **Save** icon.
   If the open mapping is an existing one, the latest mapping is saved to the referenced map file.
   If the open mapping is a new one, you are prompted to choose a file.
2. Choose or input file name. The latest mapping is saved to the chosen file.
3. To save a mapping to different file, perform the following steps.
   a. Select **File > Save As** on menu bar or **Save As** icon on tool bar.
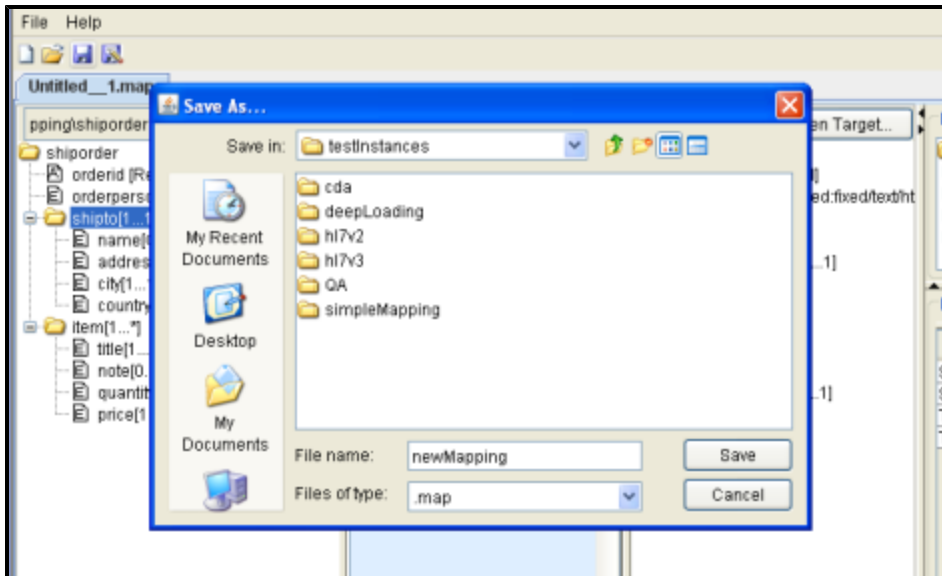
b.  Choose or input the file name. The latest mapping is saved to the chosen file.

> ⊖ **Warning**
> 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.
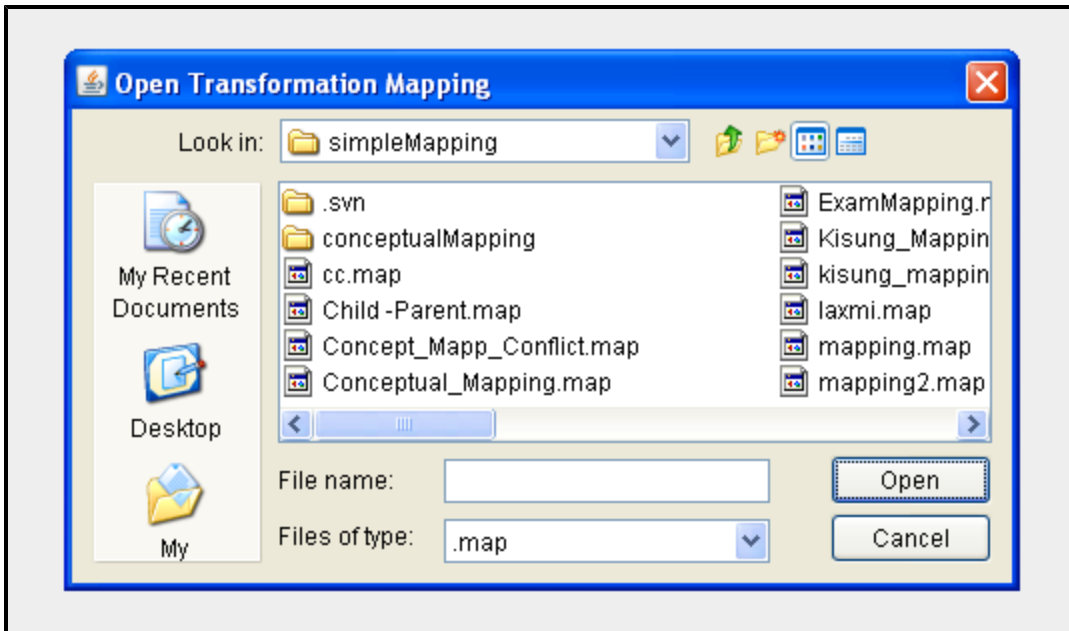
## Open Transformation Mapping

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

1.  Select **File** > **Open** > **Open** or click **Open**.
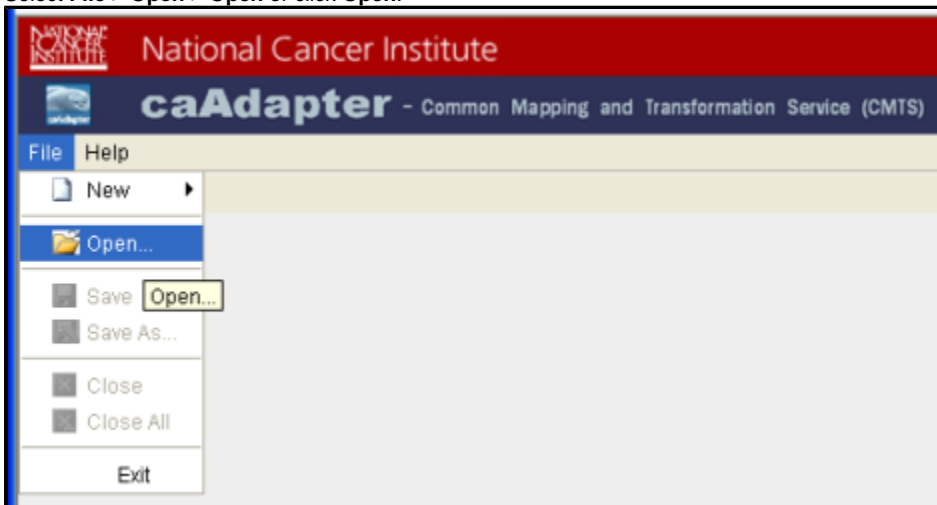


    You are prompted to choose a mapping file.
2.  Choose a mapping file and click **Open**.

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:

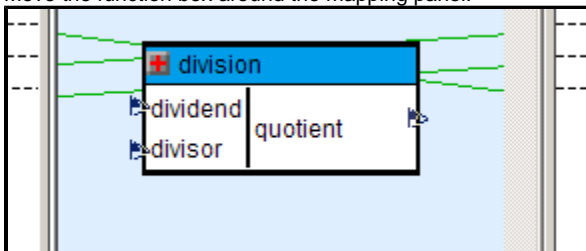- math
- constant
- dateTime
- string
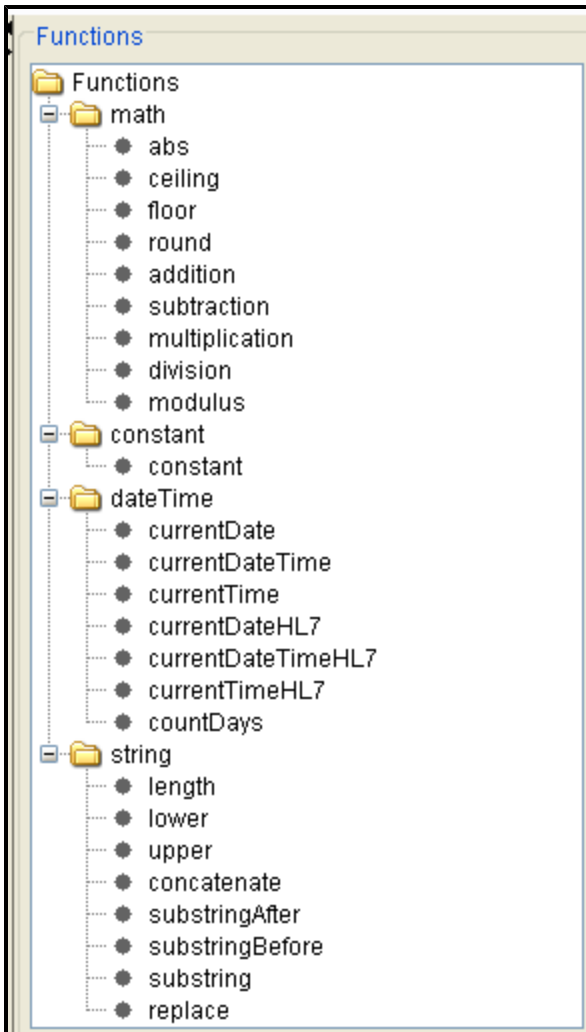
### Add Function to Transformation Mapping

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 the function box around the mapping panel.



### 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**.
4. The selected function box is removed from the mapping panel.
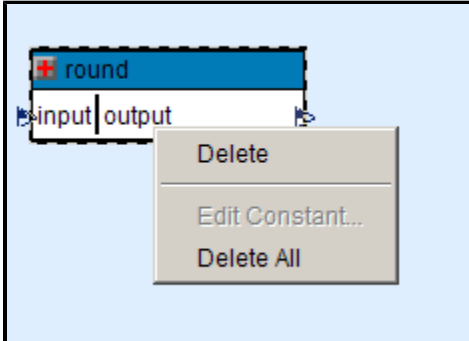


⚠️ **Be Careful**

Before deleting a function box, all the linked mapping line must be deleted.



### Edit Constant Function

To edit a constant function, perform the following steps.

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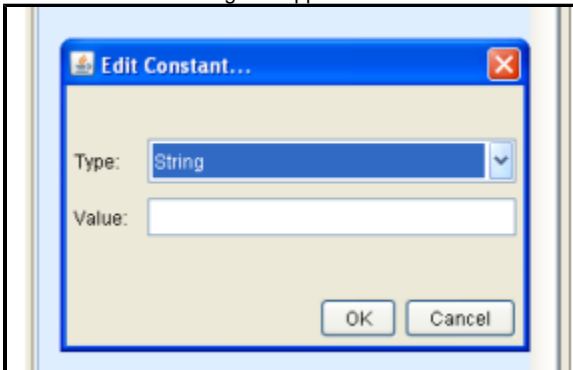


4. Change the **Type** and/or **Value** for the constant.
5. Click **OK**. The edited function is assigned with the new value.

### 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
  Format: yyyy-MM-dd
- currentDateTime – returns the current dateTime (with timezone)
  Format: yyyy-MM-ddThh:mm:ss.sss{+/-}9999
- currentTime – returns the current time (with timezone)
  Format: hh:mm:ss.sss{+/-}9999
- countDays – returns an integer that represents the day component in the localized value of the argument
- currentDateHL7 – returns the current date with HL7/ISO21090 format
  Format: yyyyMMdd
- currentDateTimeHL7 – returns the current date and time with HL7/ISO21090 format (with timezone)
  Format: yyyyMMddhhmmss.sss{+/-}9999
- currentTimeHL7 – returns the current time with HL7/ISO21090 format (with timezone)
  Format: hhmmss.sss{+/-}9999

### 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**. The selected child is set as "*Selected*" child for the choice element and the other "Selected" child is removed.
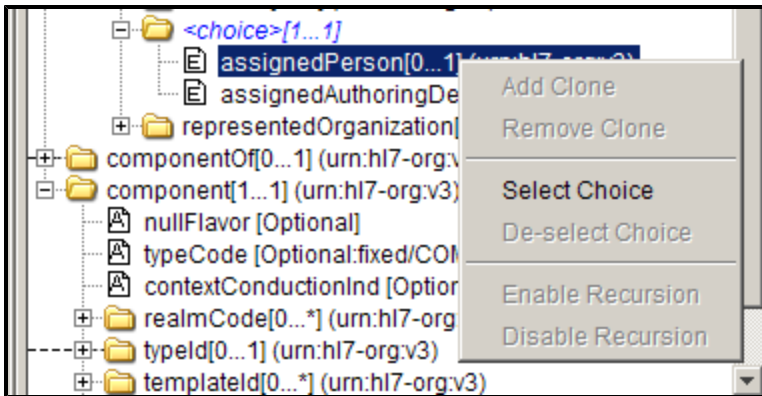
### De-select Choice Child Element

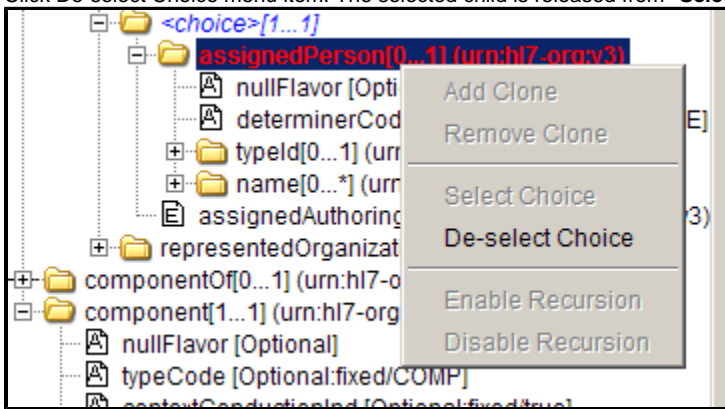To de-select a child for choice element, perform the following steps:

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 is released from "*Selected*" status.



### 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.
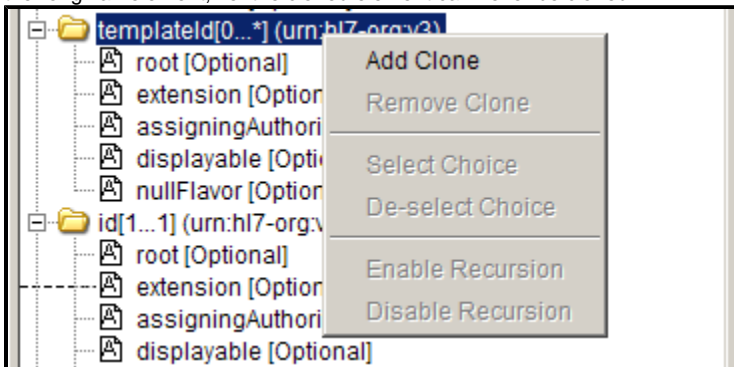


### Remove Clone Element

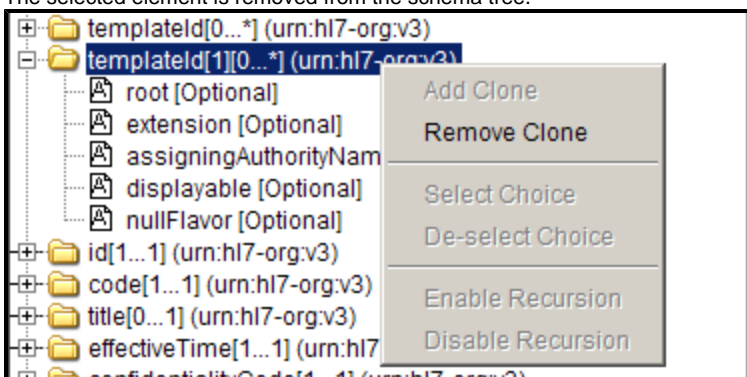To remove a cloned element, perform the following steps:

1. Right-click the element.

A popup menu appears.
The **Remove Clone** menu item is enabled if the selected element is a cloned element.
2. Click **Remove Clone**.
The selected element is removed from the schema tree.



## Enable Recursion

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

1. Right-click the 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.
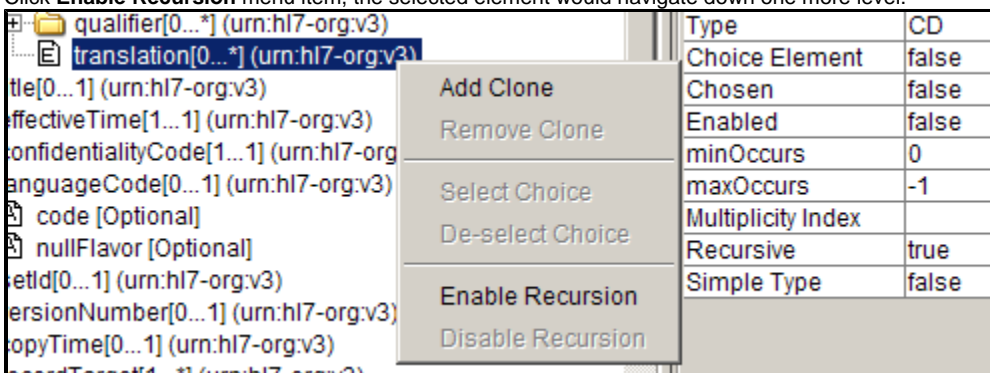


## Disable Recursion

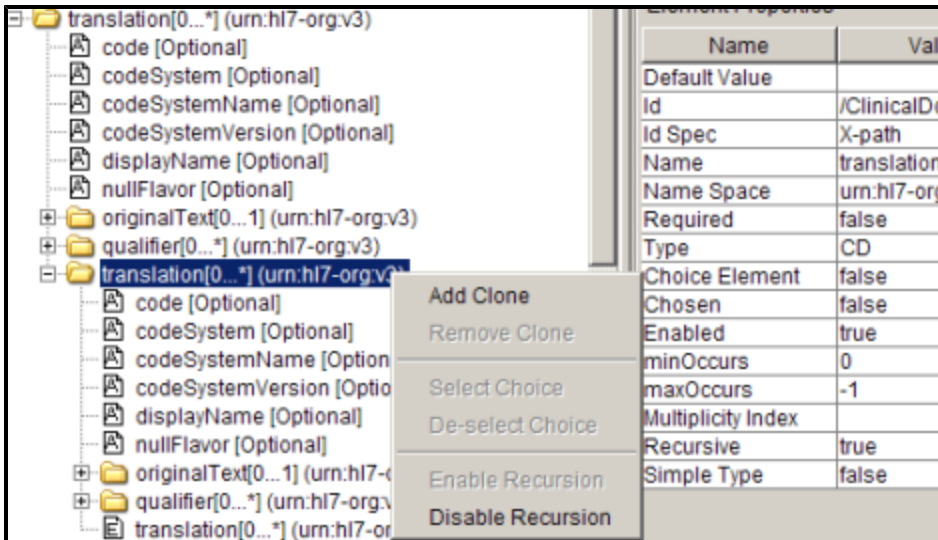To disable a recursive element, perform the following steps:

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.

# 3 - Data Transformation v1.0

## 3 - Data Transformation v1.0

This chapter explains how to create relationships between the source data schema, target schema, and data manipulation functions using transformation mapping.

Topics in this chapter include:

- XML to XML Transformation
- Generate XQuery Artifact
- Generate XSLT Artifact
- Save Transformation Result and Artifacts

CMTS transformation engine enable user to convert source data into target data using transformation instructions. It supports the following 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 the menu bar.
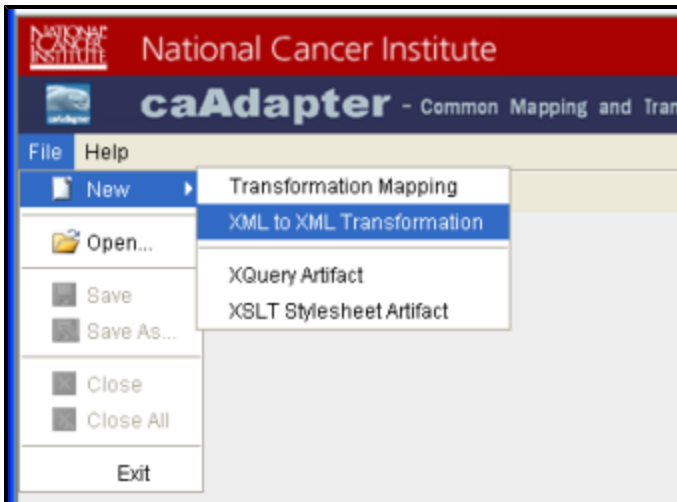
The XML to XML Transformation dialog box appears.
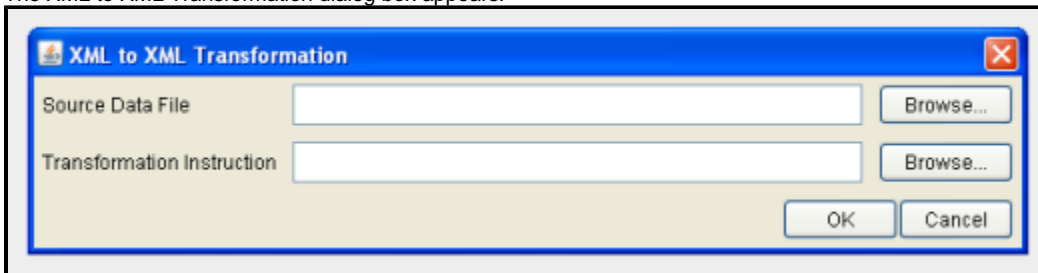


2. Next to the Source Data File box, click **Browser**.
   The Open Source Data File dialog box appears.



3. Choose the source data XML file and click **Open**.
4. Next to the Transformation Instruction box, click **Browser**.
   The Open Transformation Instruction dialog box appears.

5. Choose the transformation file in one of following formats:
   - XML to XML transformation mapping (`.map`)
   - XQuery script (`.xql`)
   - XSLT stylesheet (`.xsl`)

> **ℹ Note**
> The default transformation file format is `.map`. To change this, press the **v** button on the right side of the File of type field. Other extensions options appear in the list.

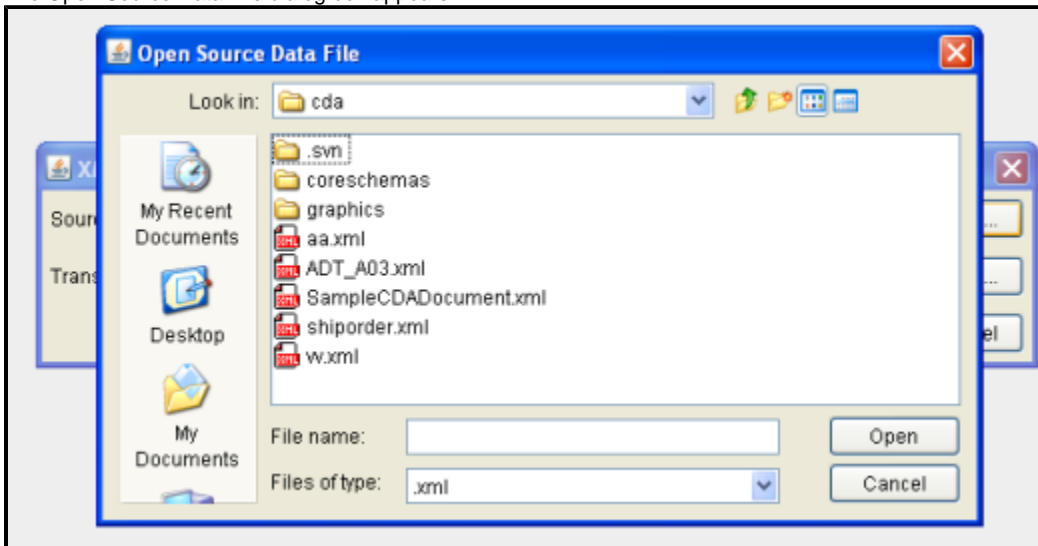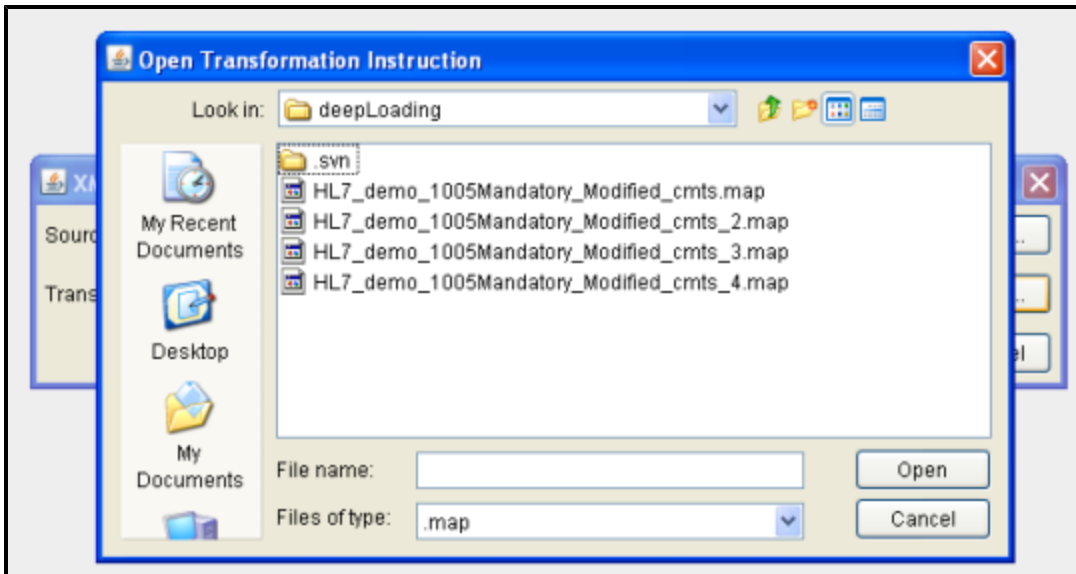6. Click **Open** to select the transformation instruction file.
7. On the XML to XML Transformation dialog box, click **OK** .
8. The transformation engine generates target data and displays it on the result panel.
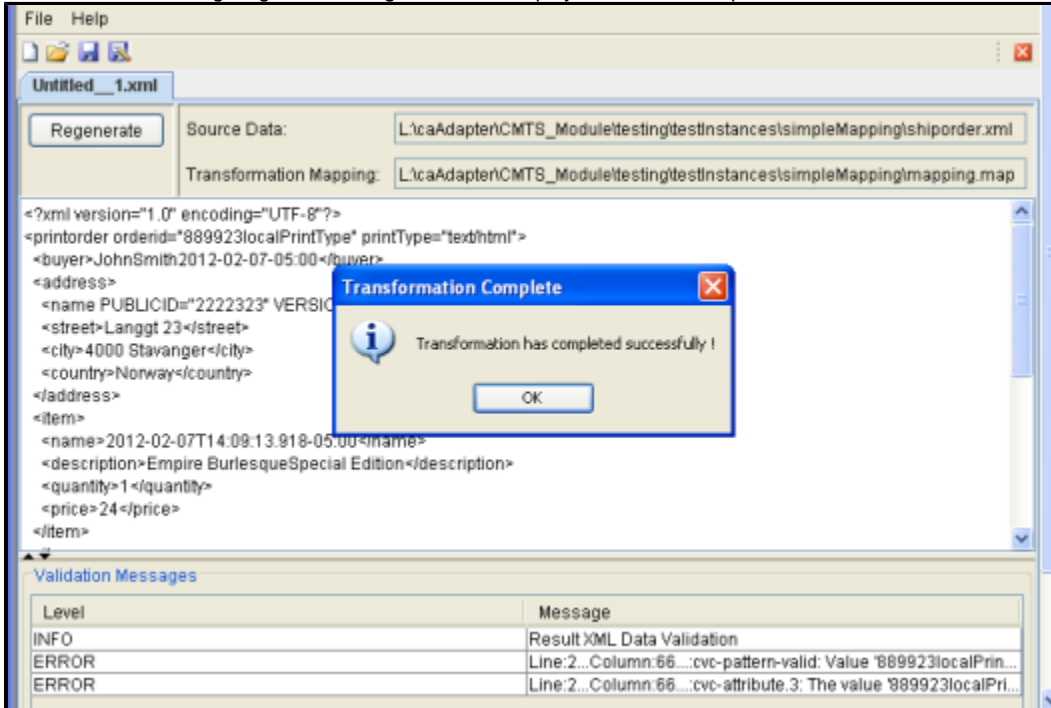


The transformation result panel is a vertical split pane. The top scroll panel displays result XML data and the low scroll panel displays validation messages.

> **ℹ Note**
> In the case of transforming with XSL or XQL, the validation message panel must be empty. If you are only transforming with MAP instructions, validation messages appear.

# Generate XQuery Artifact

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

- CMTS generates XQuery to be used by other XQuery 1.0 compatible data transformation tool
- CMTS performs 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 an XQuery artifact, perform the following steps:

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



   The XQuery Artifact dialog box appears.



2. Next to the Transformation Instruction box, click the **Browser** button.
   The Open Transformation Instruction dialog box appears.
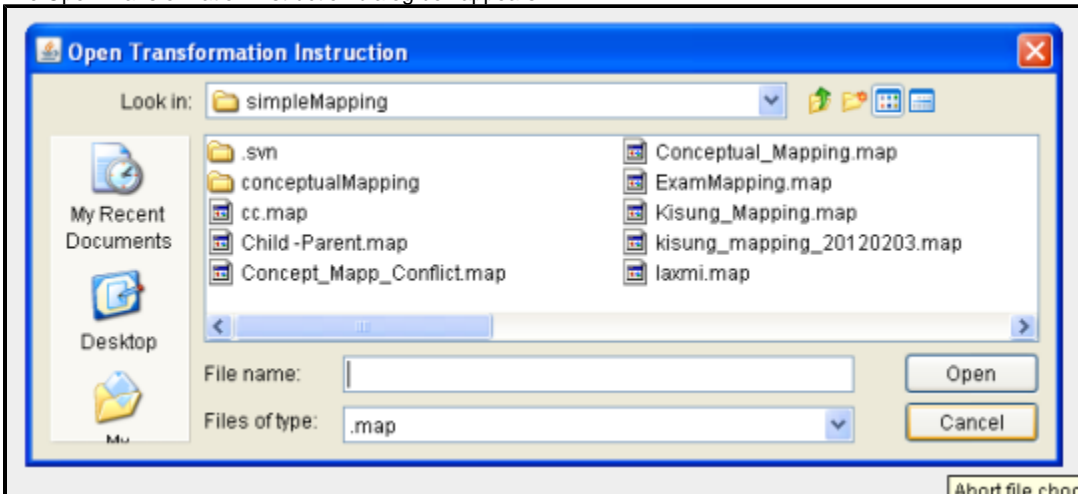


3. Choose the XML to XML transformation mapping file and click **Open**.
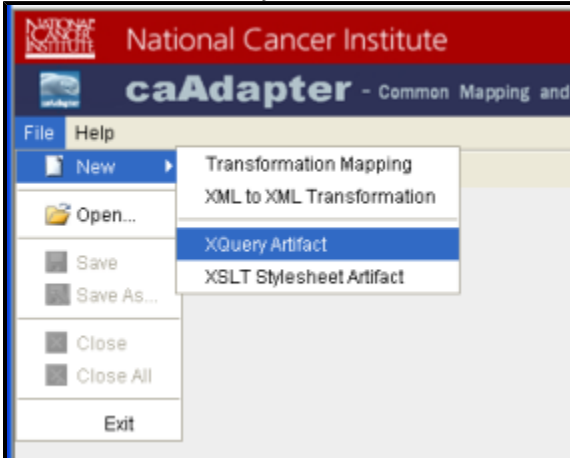4. Transformation engine generates the XQuery artifact and displays it on the result panel.

## Generate XSLT Artifact

CMTS transformation supports bi-directional compatibility with any XSLT 2.0 compatible data transformation tools. You can use this feature in the following scenarios:

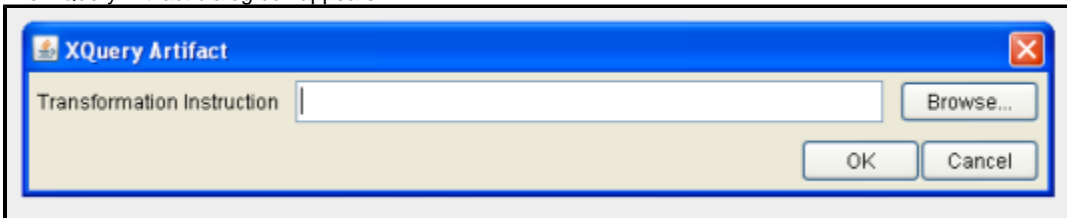- CMTS generate XSLT stylesheet to be used by other XSLT 2.0 compatible data transformation tool.
- CMTS perform data transformation using XSLT stylesheet generated by other XSLT 2.0 compatible data transformation tools. This scenario is included in the XML to XML transformation section.

To generate an XQuery artifact, perform the following steps:

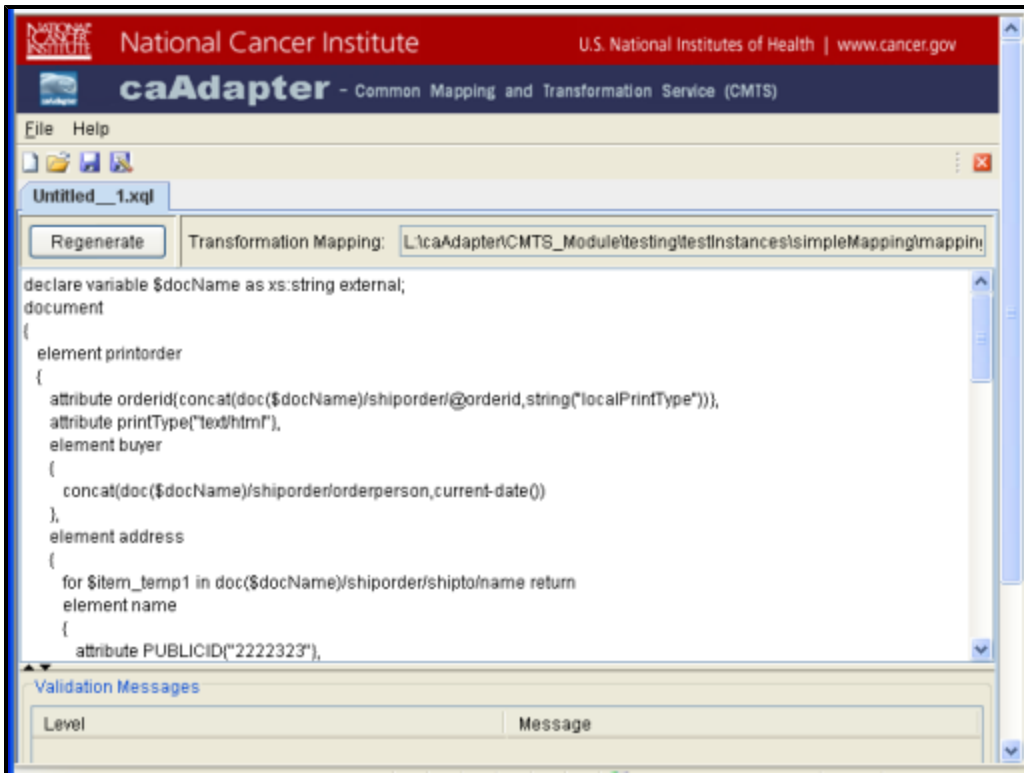1. Select **File > New > XSLT Stylesheet Artifact** from menu bar.
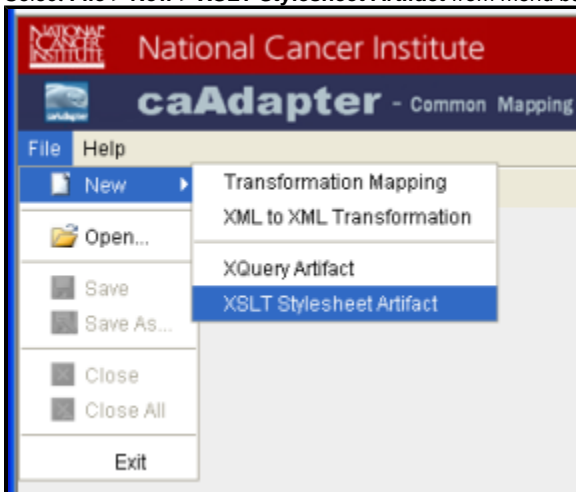


The XSLT Stylesheet Artifact dialog box appears.
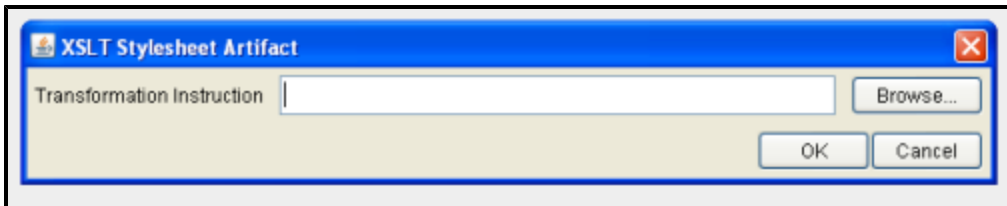
2. Next to the Transformation Instruction box, click the **Browser** button.
   The Open Transformation Instruction dialog box appears.
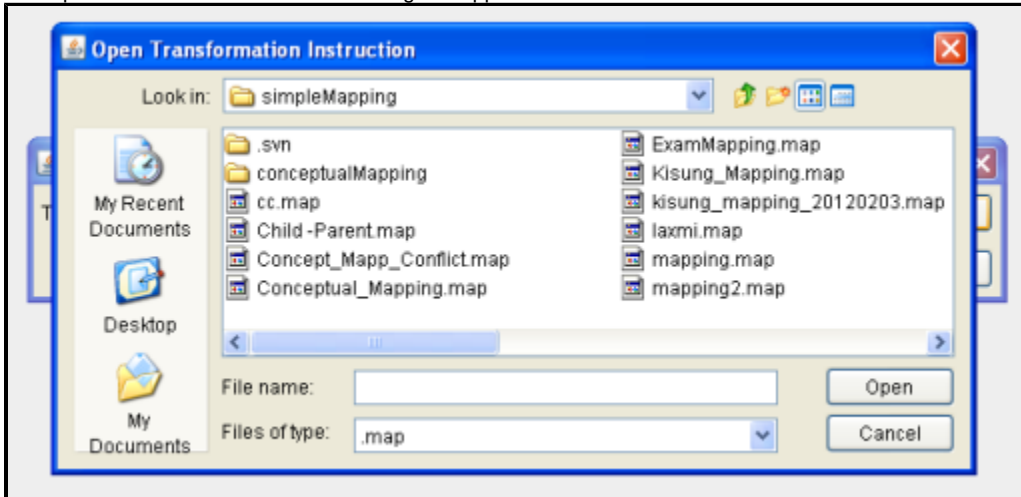


3. Choose the XML to XML transformation mapping file and click the **Open** button.
   The Transformation engine generates XQuery artifact and displays it on the result panel.
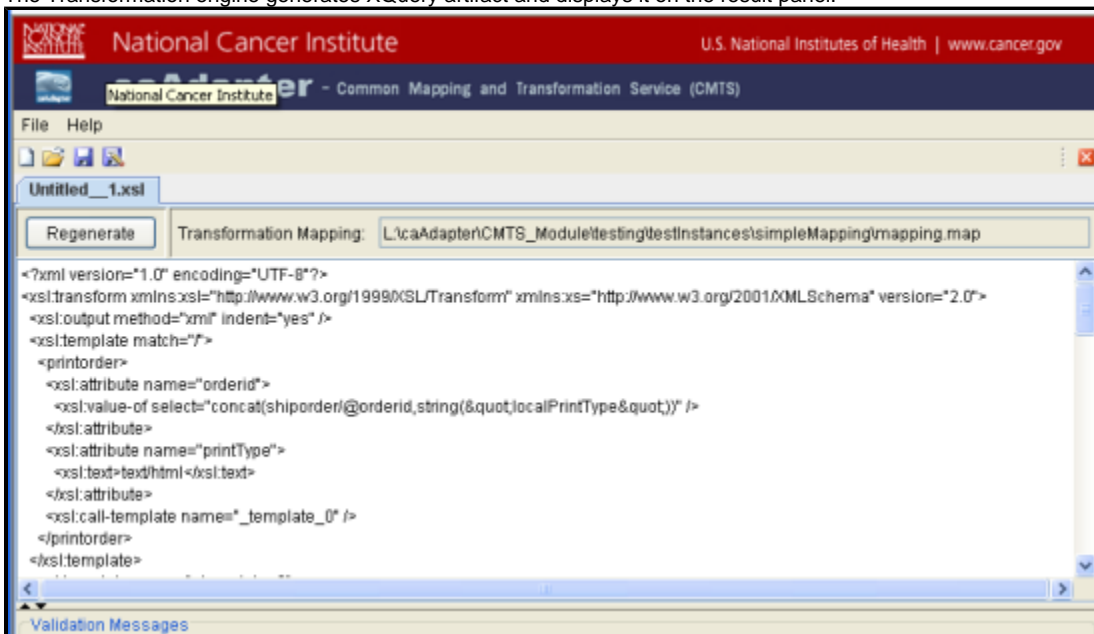


## Save Transformation Result and Artifacts

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

- Select **File** > **Save** the or **Save** button on the toolbar to save the result to the current file.

To save the saved data to different file, perform the following steps.

1. Select **File > Save As** or the **Save As** button on the toolbar.
   The Save As dialog box appears.
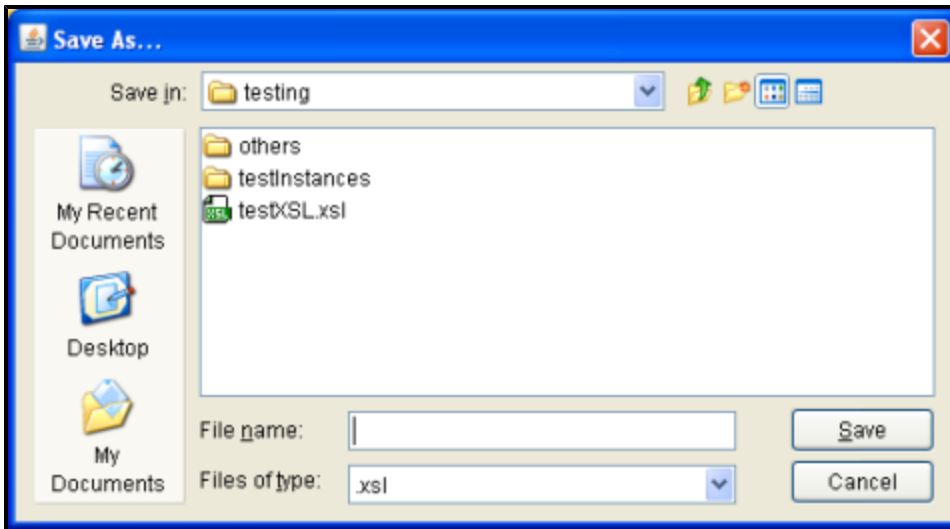
2. Choose or input the file name.
   The transformation result is saved to the chosen file with the CMTS default file extension. The file extension options are as follows:
   - XML data – .xml
   - XQuery artifact – .xql
   - XSLT stylesheet artifact – .xsl

# 4 - Programming APIs v1.0

## 4 - Programming APIs v1.0

This chapter explains how to use the Application Programming Interfaces (APIs) that allow the caAdapter Common Mapping and Transformation Service (CMTS) transformation engine to interact with a user's application.

> Topics in this chapter include:
> - Transformation API
> - Programming Sample

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 of "TransformationService" interface, its parameters, and return data.

```
transfer

java.lang.String transfer(java.lang.String sourceFile,
                          java.lang.String processInstruction)

    Transfer source data into target data using process instruction file



    Parameters:
        sourceFile - Location of source data file
        processInstruction - Location 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("c:\\result.xml"));
    ```

4.  Invoke the transformation and write the result to the output file.

    ```
    sWriter.write(transformer.transfer(sourceFile, processInstruction));
    sWriter.flush();
    sWriter.close();
    ```

> ⊖ **Warning**
> These codes may need "try-catch" blocks for specific "Exception".


# 5 - Web Service APIs v1.0


## 5 - Web Service APIs v1.0

This chapter explains how to use the caAdapter Common Mapping and Transformation Service (CMTS) web service Application Programming Interfaces (APIs) to create new mapping scenarios, browse existing mapping scenarios, and manage mapping scenarios.

Topics in this chapter include:

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 A - CMTS Web Service Definition Language. 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 B - CMTS RESTful Web Application Description Language. It manipulates CMTS transformation functionalities using a uniform set of "stateless" operation. Users can access the RESTful compliant service access point either programmatically or by launch Web browser request.

CMTS provides a 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 a mapping scenario.
- Register the mapping scenario.
- Invoke the transformation service using scenario and source data.

## Register the Mapping Scenario to the 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/.
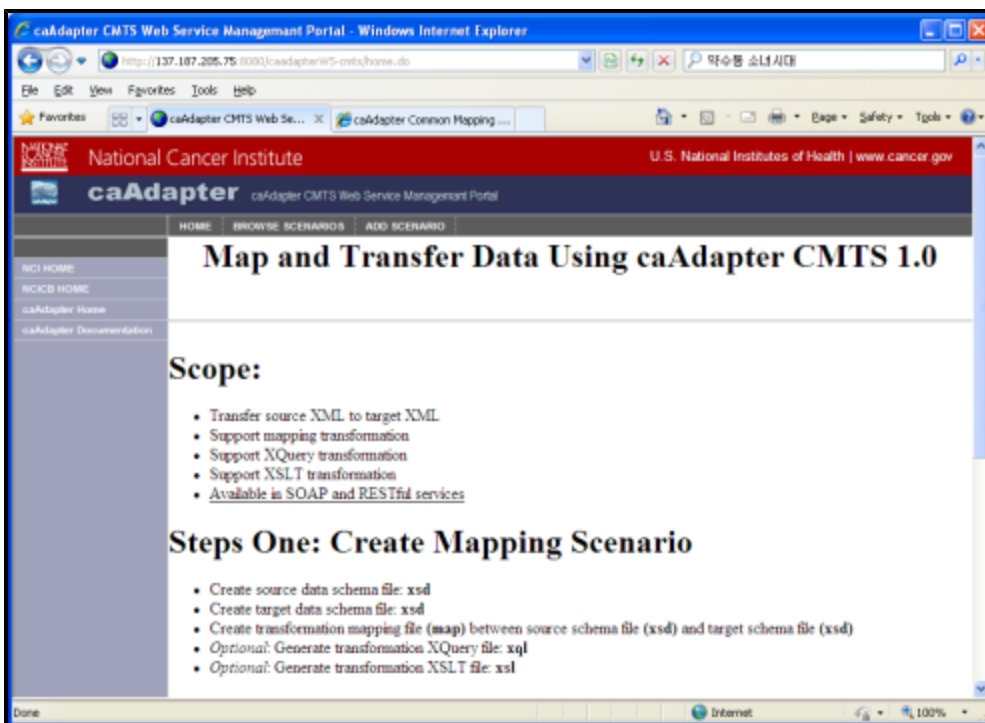
2. At the top of the page, click **Browse Scenarios** to explore registered mapping scenarios.



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



4. Before inputting registration data, count the number of `.xsd` files in each source or target side. If the number is greater than one, add more input fields by clicking **Add a Source XSD field** or **Add a Target XSD field**.

5. Select the mapping scenario type: **mapping**, **XQuery**, or **XSLT**.
6. Specify the unique name for the mapping scenario to register.

> ℹ️ **Note**
> Use this name for the web services clients later.

7. In the Transformation Instruction field, click **Browse** and navigate to the corresponding instruction file, which has a suffix of `.map`, `.xql`, or `.xsl`.
8. If the mapping scenario type is *mapping*, click the **Browse** button in the Source Schema field and navigate to the source schema file, which has a suffix of `.xsd`.
   a. If the `.xsd` file links to other schema files, click **Add a Source XSD field**. One file input field is added. You can input multiple `.xsd` files.
   b. You can add zip files of multiple source `.xsd` files. In this case, file names of all `.xsd` files you include must be unique in the zip file.
9. If the mapping scenario type is *mapping*, click the **Browse** button in the Target Schema file field and navigate to the target schema file, which has a suffix of `.xsd`. Usages are the same as step 8.
10. Click **Add Mapping Scenario**.caAdapter CMTS creates the mapping scenario based on input files and sends you a confirmation message.
    a. 
11. If any errors exist in the input fields, an error message is displayed and the request is canceled.
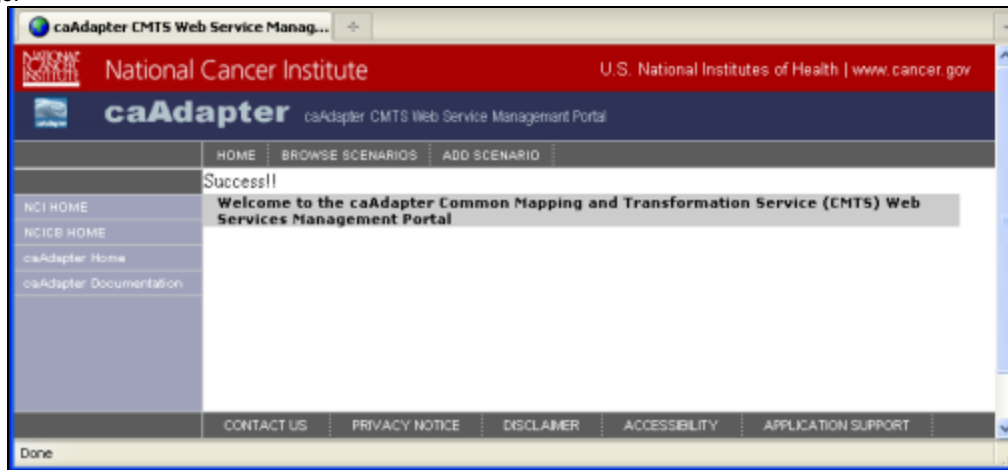


## Access SOAP Web Service

SOAP web service APIs includes two transformation operations: `transferData` and `transferResource`. Descriptions of these functions follow.

### transferData

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

Transfer a data string into target datasets.

#### Parameters

| Parameter | Description |
| --- | --- |
| mappingScenario | The name of the mapping scenario, which is registered with the 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**

| Parameter | Description |
|---|---|
| mappingScenario | The name of the mapping scenario that is registered with the 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.

A - CMTS Web Service Definition Language details these service APIs with Web Service Description Language (WSDL). The online version of the WSDL file is available at 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 Apache site .
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= "<sample_xml> ... ... </sample_xml>";
      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();
      }
}
```

ℹ️ **Legend**

| Variable | Description |
| --- | --- |
| scenario | Name of the mapping scenario registered with CMTS Web Services Management Portal |
| sourceData | If `transferData` is called, `sourceData` refers to the source file path. If `transferResource` is called, `sourceData` refers to the 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 Apache site 🗗.
2. Unzip `axis-bin-1_4.zip`.
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. 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 = "transferData";              // or "transferResource"
      String sourcData = "<sample_xml> ... </sample_xml>"; // or URL of the source data if
"transferResource"
      String myScenario = "My_WS_Scenario";                // The registered name of CMTS WS
portal

      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[]{myScenario, sourcData});
      System.out.println(res);

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

**ℹ Legend**

| Variable | Description |
|---|---|
| operationName | Name of remote operation: `transferData` or `transferResource` |
| myScenario | 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`. A JavaDoc-style description of these functions follow:

### restfulTransferData

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

RESTful API: Transfer a data string into target datasets.

#### Parameters

| Parameter | Description |
|---|---|
| mappingScenario | The name of the mapping scenario that 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)

RESTful API: Transfer a URL data resource into target datasets

**Parameters**

| Parameter | Description |
|---|---|
| mappingScenario | The name of the mapping scenario that 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 on 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. B - CMTS RESTful Web Application Description Language details these service APIs with Web Application Description Language (WADL). The online version of WADL file is available at http://caadapter.nci.nih.gov/caadapterWS-cmts/services/restful?_wadl&_type=xml.

## Apache CXF Client Access RESTful Service

Perform the following steps to build the Apache CXF client program.

1. Download Apache CXF 2.5 (apache-cxf-2.5.0.zip) from the Apache site.
2. Unzip apache-cxf-2.5.0.zip.
3. Add the cxf-2.5.0.jar to your classpath.
4. 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 parameters
            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);
             }
    }
```

> **Legend**
>
> | Variable | Description |
> | --- | --- |
> | 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}

| ⓘ | **Legend** | |
|---|---|---|
| | **Variable** | **Description** |
| | 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 the source data URL. |

# A - CMTS Web Service Definition Language

## A - CMTS Web Service Definition Language

This chapter explains how to use Web Service Definition Language (WSDL) to access the Application Programming Interface (API) for the caAdapter Common Mapping and Transformation Service (CMTS)data transformation portal.

Topics in this chapter include:

* WSDL Elements

### WSDL Elements

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"
```

```xml
   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>
```

# B - CMTS RESTful Web Application Description Language

## B - CMTS RESTful Web Application Description Language

This chapter explains how to use Web Application Description Language (WADL) to access the RESTful service Application Programming Interface (API) for the caAdapter Common Mapping and Transformation Service (CMTS) data transformation portal.

Topics in this chapter include:

* WADL Elements

### WADL Elements

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.

```xml
<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>
```