

# SWAD–Europe: WP6.1 Pragmatic Methods for Mapping Semantics from XML Schemas: Implementation Guide

Project name:

Semantic Web Advanced Development for Europe (SWAD-Europe)

Project Number:

IST-2001-34732

Workpackage name:

SWAD-Europe: XML and Semantic Web Integration research prototypes

Workpackage description:

<http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-6.html>

Deliverable title:

Pragmatic Methods for Mapping Semantics from XML Schemas: Implementation Guide

URI:

Authors:

[Brian Matthews](#), CCLRC.

[Ian Johnson](#), CCLRC.

Abstract:

This document provides user documentation for the Mapping Tool, mapping between XML Schema and OWL Ontologies.

Status:

Completed report 2004-06-06

Comments on this document should be sent to the public SWAD-Europe mailing list, [public-esw@w3.org](mailto:public-esw@w3.org).

## THE SWAD–E ONTOLOGY–SCHEMA MAPPING TOOL

*INTEGRATING LEGACY XML INTO THE SEMANTIC WEB* - This paper is part of SWAD-E Workpackage 6.1.

This workpackage builds on the findings of WP5, Integration with XML. It provides an approach to defining mappings between an ontology covering a subject area and a schema defining an XML document. This workpackage has developed a tool to allow this mapping to be defined graphically, supporting Web Ontology Language (OWL) and W3C XML Schema 1.0.

This tool allows one to graphically define mappings between an ontology and an XML schema. The file formats the tool supports are Web Ontology Language (OWL) and XML Schema Definition. This note describes how to install the application, run and use the application.

## DOWNLOAD

The mapping tool can be downloaded from: <http://www.w3c.rl.ac.uk/SWAD/wp6/legacy-integration.html>.

The tool is available in the file mapper.zip. This contains a bundle of files:

mapper-dist.jar	A Jar file containing the Java class files for the application in the package swad.mapper. The main class is swad.mapper.Mapper
build.xml	An Ant build file describing tasks to run the application from the mapper-dist.jar file, or to compile the files under the src directory and run the application from the compiled location.
build.properties	Contains property definitions to control the operation of Ant tasks
src	A directory containing the Java source files which comprise the application.

## PRE-REQUISITES

The Jena Semantic Web framework version 2 from HP Laboratories - available at: <http://jena.sourceforge.net/downloads.html> The Ant build system, version 1.6.1 or later - this is available at <http://ant.apache.org/bindownload.cgi> The Java 2 SE SDK is available at <http://java.sun.com/j2se/corejava/index.jsp> (strictly, only the Java Runtime Environment is necessary - this is available at <http://www.java.com>.)

## INSTALLING THE APPLICATION

Unzip the mapper.zip file - use a tool such as FreeZip under Microsoft Windows, or the GNU unzip utility under Linux or Solaris.

Edit the build.properties file to reflect your settings - a sample build.properties file takes the form:

```
# Sample ResourceBundle properties file

jena.dir=/space/SemWeb/Jena-2.1

#Whether to use an HTTP proxy or not
proxySet=false

proxyHost=wwwcache.rl.ac.uk

proxyPort=8080

build.dir=build
```

1. Change the value of the property jena.dir to reflect the location of your Jena installation.
2. If you use an HTTP proxy, set the property proxySet to true, set the property proxyHost to the hostname of the HTTP proxy server, and set proxyPort to the appropriate port number. These settings come into play if an ontology or schema imports resources from an HTTP URI.

## RUNNING THE APPLICATION

Using Ant is the easiest way to run the application. To run the application from the Jar file mapper-dist.jar, invoke Ant as 'ant' - the default task, run-from-jar, runs the application from the Jar file.

To build the application from source and run the result, invoke 'ant run'.

If you wish to run the application without using Ant, set a variable (e.g. SWADPATH) to contain the pathnames of all the Jar files under the Jena lib directory and the file mapper-dist.jar. Invoke java by a command line similar to:

```
java -cp %SWADPATH% swad.mapper.Mapper # Windows
java -cp $SWADPATH swad.mapper.Mapper # Linux or Solaris
```

## USING THE APPLICATION

The application presents a main window containing File menus to open an ontology file and a schema file. Opening either of these files presents a simple Swing JTree browser, allowing the user to explore the structure and contents of the files.

## BROWSING AN ONTOLOGY

The ontology browser presents an OWL file in a number of tabbed panes:

Classes:	Shows the asserted class hierarchy in a Swing JTree.
Object Properties:	Shows the compositional relationship between classes in a Swing JTable.
Datatype Properties:	Shows the lowest-level datatypes in a Swing JTable.

The displays in the panes are linked, such that selecting a class in the Class pane will highlight any associated Object Properties and Datatype Properties.

Selecting a class displays in the tree its name in the text area below the ontology - this serves as a drag source for use in the mapping process (see THE MAPPING PROCESS below).

Right-selecting in either the Object Properties or Datatype Properties tables will preserve the multiple selection status (if any), and similarly displays the name of the property below the table.

## BROWSING A SCHEMA

The schema browser presents a simple tree view of the XSD schema, representing the Document Object Model view of the schema. The labels on the tree nodes show the XSD type of the DOM node and its attributes.

Selecting a node in the tree displays an XPath expression identifying that node in the schema. This XPath expression serves as a drop target for use in the mapping process.

## THE MAPPING PROCESS

The user can map between the ontology and the schema by dragging from the drag source in the ontology browser to the drop target in the schema browser. Dragging a class from the drag source in the ontology and dropping it onto

the drop target area in the schema browser adds a representation of the mapping into a scrolling list in the application window.

For mapping a property in the ontology, the drop operation selects the target of the mapping. The application then prompts the user to select the domain of the target and the range of target by selecting the appropriate nodes in the schema and clicking the 'Select Node' button. If the user decides to cancel this property mapping operation, they can click the 'Cancel Selection' button instead and start again.

The domain of the target represents the part of the schema representing the part of the XML instance document which 'contains' the target. The range of the target is its datatype.

## SAVING A MAPPING

When the user makes the first mapping, a new menu item 'Save' appears on the File menu. This allows the user to write the list of mappings to a file by presenting a file chooser dialogue.

## OUTPUT FILE FORMAT

The output file contains a `map:mapping` element which serves to define the XML namespace `xmlns:map`. The elements `map:sourceNamespace` and `map:targetNamespace` define the namespaces of the source ontology and target schema respectively.

The `map:classMap` element contains nested `map:source` and `map:target` elements to define a mapping from a class in the ontology to the schema. The `map:propertyMap` element contains nested `map:source`, `map:target`, `map:domain` and `map:range` elements to define the source property in the ontology, the target node in the schema, and the domain node - the node in the XML instance document associated with the target - , and the range of the target node.

An example mapping file appears below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<map:mapping
  xmlns:map="http://www.w3c.rl.ac.uk/xml/SchemaMap"
  >

  <map:sourceNamespace value="http://a.com/ontology#" />
  <map:targetNamespace value="http://www.w3c.rl.ac.uk/orders" />

  <map:classMap>
    <map:source class="http://a.com/ontology#Customer" />
    <map:target path='/xsd:schema/xsd:element[@name="customer"]' />
  </map:classMap>

  <map:classMap>
    <map:source class="http://a.com/ontology#Order" />
    <map:target path='/xsd:schema/xsd:complexType[@name="orderType"]' />
  </map:classMap>

  <map:classMap>
    <map:source class="http://a.com/ontology#OrderStatus" />
    <map:target path='/xsd:schema/xsd:simpleType[@name="allowableOrderStatusType"]' />
  </map:classMap>

  <map:propertyMap>
    <map:source property="http://a.com/ontology#customerName" />
    <map:target
      path='/xsd:schema/xsd:complexType[@name="orderType"] /
      xsd:attribute[@name="orderID"]' />
    <map:domain path='/xsd:schema/xsd:element[@name="customer"]' />
    <map:range path='/xsd:schema/xsd:complexType[@name="customerType"] /
      xsd:attribute[@name="customerName"]' />
  </map:propertyMap>

  <map:propertyMap>
    <map:source property="http://a.com/ontology#shippingPostCode" />
    <map:target
      path='/xsd:schema/xsd:complexType[@name="customerType"] /
      xsd:attribute[@name="shippingPostalCode"]' />
    <map:domain path='/xsd:schema/xsd:element[@name="customer"]' />
    <map:range path='/xsd:schema/xsd:simpleType[@name="postalCode"]' />
  </map:propertyMap>

  <map:propertyMap>
```

```

    <map:source property="http://a.com/ontology#shippingAddress" />
    <map:target
path='/xsd:schema/xsd:complexType[@name="customerType"] /
xsd:attribute[@name="shippingAddress"]' />
    <map:domain path='/xsd:schema/xsd:element[@name="customer"]' />
    <map:range path='/xsd:schema/xsd:simpleType[@name="description"]' />
</map:propertyMap>

<map:propertyMap>
    <map:source property="http://a.com/ontology#orderStatus" />
    <map:target
path='/xsd:schema/xsd:complexType[@name="orderType"] /
xsd:attribute[@name="ordersStatusType"]' />
    <map:domain
path='/xsd:schema/xsd:complexType[@name="customerType"] /
xsd:sequence/xsd:element[@name="order"]' />
    <map:range
path='/xsd:schema/xsd:complexType[@name="orderType"] /
xsd:attribute[@name="ordersStatusType"]' />
</map:propertyMap>

<map:propertyMap>
    <map:source property="http://a.com/ontology#customer" />
    <map:target
path='/xsd:schema/xsd:complexType[@name="customerType"]
/xsd:sequence/xsd:element[@name="order"]' />
    <map:domain path='/xsd:schema/xsd:complexType[@name="orderType"]' />
    <map:range path='/xsd:schema/xsd:complexType[@name="customerType"]' />
</map:propertyMap>

<map:propertyMap>
    <map:source property="http://a.com/ontology#orderStatusDescription" />
    <map:target
path='/xsd:schema/xsd:simpleType[@name="allowableOrderStatusType"]' />
    <map:domain
path='/xsd:schema/xsd:simpleType[@name="allowableOrderStatusType"]' />
    <map:range path='/xsd:schema/xsd:complexType[@name="customerType"]' />
</map:propertyMap>

<map:propertyMap>
    <map:source property="http://a.com/ontology#customerName" />
    <map:target
path='/xsd:schema/xsd:complexType[@name="customerType"]
/xsd:attribute[@name="customerName"]' />
    <map:domain path='/xsd:schema/xsd:element[@name="customer"]' />
    <map:range path='/xsd:schema/xsd:complexType[@name="customerType"]' />
</map:propertyMap>

</map:mapping>

```