



# D24.2v0.1. WSMO Grounding

WSMO Working Draft 19 January 2005

**This version:**

<http://www.wsmo.org/2005/d24/d24.2/v0.1/20050119/>

**Latest version:**

<http://www.wsmo.org/2005/d24/d24.2/v0.1/>

**Previous version:**

none

**Authors:**

Jacek Kopecký  
Dumitru Roman

This document is also available in non-normative [PDF](#) version.

Copyright © 2005 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

---

## Table of contents

[1. Introduction](#)

[2. Grounding WSMO Ontology to XML](#)

[2.1 Overview of Modeling in XML](#)

[2.2 Mapping XML to WSMO Ontology](#)

[2.3 Requirements for the Mapping Language](#)

[2.4 Mapping Language](#)

[3. WSMO Service Grounding to WSDL](#)

[3.1 WSDL Overview](#)

[3.2 Relation WSMO - WSDL](#)

[4. Conclusions and further work](#)

[References](#)

[Acknowledgement](#)

[Change Log](#)

---

## 1. Introduction

Currently, Web Services usually communicate with their clients using XML messages whose structure is described using XML Schema [\[XMLSchema\]](#), usually embedded in a Web Services Description Language [\[WSDL\]](#) document. Viewed from this point of view, Web Services and their clients are entities that produce and consume XML data that follows the given XML Schema structure. The semantic interpretation (understanding) of this data is implicit within the client or server code.

On the other hand, Web Services Modeling Ontology - WSMO [\[Roman et al. 2004\]](#) describes functional and behavioral aspects of Web Services, where the inputs and outputs are described in terms of ontologies as concepts, their instances and relations among them, with a great portion of the meaning of the data explicit in its ontology. From this point of view, Web Services and their clients are entities that produce and consume semantic data. There are multiple ways (syntaxes)

of representing this semantic data on the wire for exchange, including syntaxes in XML, but this is generally not a consideration when talking about the semantic Web Services and their (also semantic) clients.

Among the goals of Semantic Web Services is automatic goal fulfillment, which implies automatic service composition and execution. These systems will operate at the level of semantic data. The IT industry, however, is currently adopting Web Services technologies on the XML level, wary of unproven semantic approaches. Therefore Semantic Web Services systems must be able to interoperate with syntactical Web Services so that these services can also be automatically composed and executed. In the area of semantic description of Web Services it means that the ontological languages must be able to semantically annotate XML Web Services.

There are two independent areas of relationship between WSMO and the syntactical descriptions of a Web Services: data in WSMO ontologies has to be mapped to XML data, usually described using XML Schema ([Section 2](#)), and the functional and behavioral service descriptions in WSMO have to be related to the description construct present in WSDL ([Section 3](#)).

We refer to both these areas collectively as *WSMO grounding*.

## 2. Grounding WSMO Ontologies to XML

Semantically, Web Service inputs and outputs are described in WSMO using ontologies. Syntactically, they are described in WSDL using schemas. As already mentioned, we envision that semantic agents (e.g. an automatic goal fulfillment - service discovery and execution engine) must be able to communicate with plain syntactic Web Services (e.g. existing Amazon Web Services) because the industry is currently deploying heavily only the syntactical Web Services.

During the communication of a semantic-level client and a syntactic-level Web Service, two directions of data transformations are necessary: the client's semantic data must be written in an XML form that can be sent as a request to the service, and the response data coming back from the service must be interpreted semantically by the client. In other words, the request data must be transformed from an ontological form to XML and, conversely, the response data must be transformed from XML to an ontological form.

Since the semantics of XML data is only implicit, at best described in plain text in the specification of an XML language, a human designer is required to specify these data transformations so that they can be executed when a semantic client needs to communicate with a syntactic Web Service. This section details a language for describing such transformations.

### 2.1 Overview of Modeling in XML

WSMO ontologies use constructs like concepts, instances, attributes and relations to model data. In XML there is significant freedom in how these aspects of knowledge will be represented using elements, attributes and character data. The following table summarizes the most important constructs both in WSMO ontologies and in XML. Transformations between semantic and syntactic data must be able to map between these two sets of constructs:

Figure 1: Comparing WSMO ontology and XML modeling primitives

WSMO ontologies	XML
concepts	elements
instances	nested elements
attributes	order of elements
relations	attributes
unordered sets	character data

There are three important differences between modeling in WSMO ontologies and XML:

- In WSMO ontologies, there are no primitives for ordered lists; in XML, ordering is often significant among sibling elements or even between character data and elements in so called mixed content.
- Relationships in WSMO ontologies are explicit in the form of attributes and relations; in XML they may be implied by the ordering or nesting of elements.
- WSMO ontologies are general graphs and instances are often involved in multiple relationships - the form of reference serialization is only a syntax consideration; XML provides a tree structure and any relationships between elements that are not directly nested are either done using references (ID, XPath etc.) or are implied somehow.

**Note:** both WSMO ontologies and XML use the term "attribute". It is necessary to distinguish between these terms because the semantic mapping can map a WSMO attribute into both an XML attribute or element, and likewise an XML attribute can represent a value of a WSMO attribute or it can represent the name or type of the instance or attribute represented by the containing element. Therefore in the rest of this text XML attributes are qualified with the word "XML" and WSMO attributes are only called "attributes".

The following example XML data will illustrate some of these differences. Note that it is not a completely real RSS feed instance, it has been tweaked for the purposes of the example.

```

01 <feed>
02   <feedTitle>CNN.com - Technology</feedTitle>
03   <feedSource>http://rss.cnn.com/rss/cnn_tech.rss</feedSource>
04   <feedUpdateInterval>10</feedUpdateInterval>
05   <feedStatus>1</feedStatus>
06   <feedLastModified>Mon, 17 Jan 2005 14:37:57 GMT</feedLastModified>
07   <attributes>
08     <attribute name="language">en-us</attribute>
09     <attribute name="copyright">
10       © 2005 Cable News Network LP, LLLP.
11     </attribute>
12     <attribute name="pubDate">Mon, 17 Jan 2005 09:21:45 EST</attribute>
13   </attributes>
14   <item>
15     <title>Airbus to unveil A380 double-decker</title>
16     <description>Read full story for latest details.</description>
17     <source>
18       http://www.cnn.com/rssclic/2005/BUSINESS/01/17/
19       airbus.380.reut/index.html?section=cnn_tech
20     </source>
21     <id>http://example.com/9435</id>
22     <relatedStory>http://example.com/9372</relatedStory>
23     <readStatus>0</readStatus>
24     <attributes/>
25   </item>
26   <item>
27     <title>'Forgotten Realms' heavy on action</title>
28     <description>
29       Atari's "Forgotten Realms: Demon Stone" follows a band of adventurers
30       who inadvertently release two imprisoned demons.
31     </description>
32     <source>
33       http://www.cnn.com/rssclic/2005/TECH/fun.games/
34       01/14/dandd.review/index.html?section=cnn_tech
35     </source>
36     <id>http://example.com/9420</id>
37     <readStatus>0</readStatus>
38     <attributes/>
39   </item>
40 </feed>

```

The element `feed` (line 1) represents an instance, the following five elements (`feedTitle`, `feedSource`, `feedUpdateInterval`, `feedStatus`, `feedLastModified`, lines 2-6) directly represent attributes of that instance, whereas the elements `attribute` (lines 8-12), which represent additional attributes, identify the names of the attributes using the values of the XML attribute `name`. The element `attributes` (line 7) serves only as a container here and can be ignored, similarly to the empty containers on lines 24 and 38.

The elements `item` (lines 14 and 26) represent an ordered list of feed items, which would probably be interpreted in an ontology as a head-tail list:

```
instance memberOf feed
  ...
  firstItem hasValue <<http://example.com/9435>>
instance <<http://example.com/9435>> memberOf item
  ...
  nextItem hasValue <<http://example.com/9420>>
instance <<http://example.com/9420>> memberOf item
  ...
```

And finally the element `relatedStory` (line 22) would not represent a literal attribute (as most of the other elements would) but instead the value `http://example.com/9372` is a reference to another `item` instance, forming a graph of related stories.

## 2.2 Mapping XML to WSMO Ontology

As shown in the previous section, there is no direct mapping between an XML vocabulary and a WSMO ontology, mostly because relations can be represented in multiple ways in XML, or they can even be implied. As a result, we require a human operator to create the mapping.

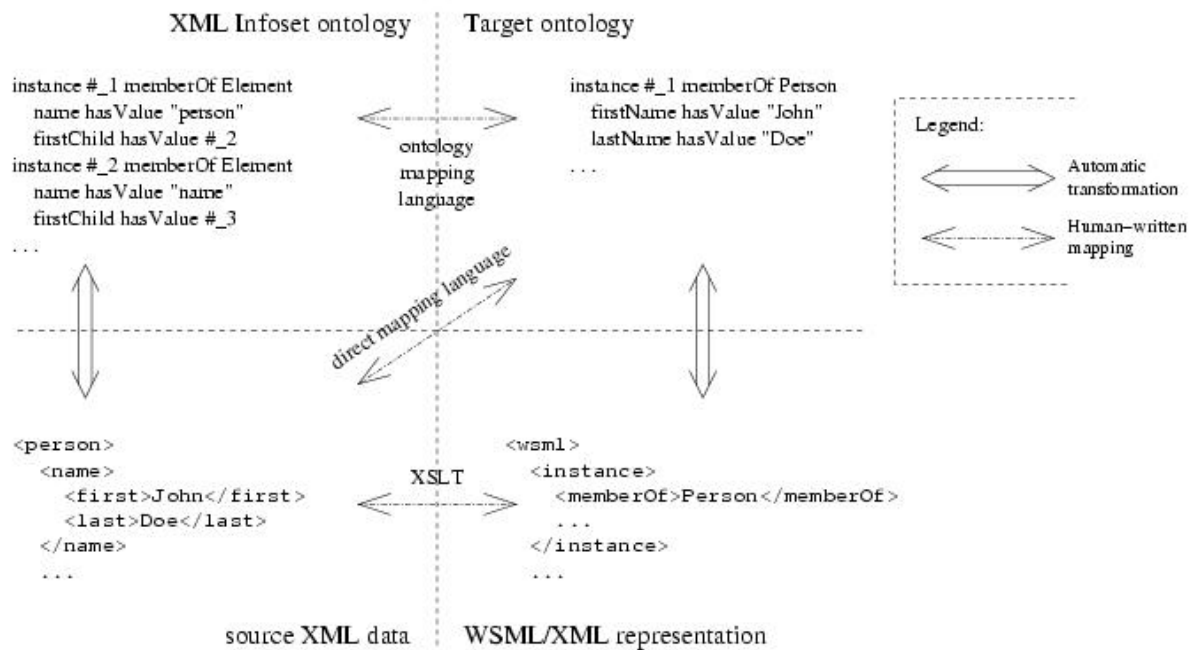
This section uses the terms *source XML* and *target ontology* to mean the XML data required or produced by a Web Service and the semantic data produced or received by the semantic client. Albeit the words "source" and "target" are used, the mapping transformations must go both ways, as discussed earlier.

Multiple technologies can be reused for this mapping; generally three approaches are possible for the mapping:

1. Transformations between the source XML data and an XML representation of the target ontology (for example, WSML/XML syntax). Any XML transformation tools and languages can be used for this purpose, but the most prominent language in this area is the XSL Transformations language [\[XSLT\]](#). WSML/XML can be automatically parsed into or generated from semantic data.
2. Ontological transformation between a semantic representation of the source XML tree (an envisioned ontology for XML Infoset [\[XMLInfoset\]](#)) and the target ontology, using an ontology mapping approach. The transformations between the XML data and the XML tree ontology would be automatic in this approach.
3. Direct mapping between the source XML data and the target ontology, using a mapping language specifically developed for this purpose.

The following figure illustrates these three approaches.

Figure 2: Mapping approaches



The first approach, with manual mapping using XSLT from XML to WSML/XML and back, is demonstrably working, but we find it unnatural due to the structure of WSML/XML documents, which is basically a flat listing of instances and relationship instances, among others. XML transformation languages are intended for transformations between hierarchical structures rather than flat ones.

The second approach, with manual mapping between XML Infoset ontology and WSMO ontology has not been tried yet, mostly because an ontology mapping for WSMO doesn't exist yet. We expect a similar problem as with the first approach, though, because of the structural mismatch between the XML Infoset (tree) ontology and any general (graph) target ontology. Ontology mapping approaches are generally based on the existence of inherent similarities in the two mapped models, similarities that cannot be expected between an XML Infoset ontology and a domain-specific ontology.

The following sections concentrate on the third approach, first analyzing the requirements for a direct XML-to-Ontology mapping language and then designing such a language.

## 2.3 Requirements for the Mapping Language

The mapping language must primarily be able to describe transformations between instances of an XML vocabulary (XML documents consisting of elements, XML attributes and character data) and instances of a WSMO ontology (knowledge graphs consisting of concept instances, their attributes and relation instances). For that we have to tease out from the XML data the ontological instances, attributes and relations. The following list describes a number of mapping patterns that we have identified, together with the explanation of a scenario demonstrating the pattern.

**Note:** both WSMO ontologies and XML use the term "attribute". It is necessary to distinguish between these terms because the semantic mapping can map a WSMO attribute into both an XML attribute or element, and likewise an XML attribute can represent a value of a WSMO attribute or it can represent the name or type of the instance or attribute represented by the containing element. Therefore in the rest of this text XML attributes are qualified with the word "XML" and WSMO attributes are only called "attributes".

### an element is an instance

An XML element, for example `<person ... />`, is mapped to an instance of a concept identified by the name or type of the element, e.g. `concept Person`. The identifier of this instance is unknown but can be contained in further data within or surrounding the XML

element.

### **a child element is an attribute with literal value**

An attribute is modeled in XML as a child element with character data inside it, for example

```
<person>
  <firstName>John</firstName>
</person>
```

the child element `firstName` is mapped to an attribute of the instance represented by the parent element. The name of the attribute is identified by the name or type of the element, e.g. `hasFirstName`, or by the implicit relationship between the element and its parent element.

### **a child element is an attribute with instance value**

An attribute whose value is another concept instance is modeled in XML as a child element, for example

```
<person>
  <daughter>
    <firstName>Jane</firstName>
  </daughter>
</person>
```

the child element `daughter` is mapped to an instance of a concept identified by the name or type of the element, and this instance is the value of an attribute of the instance represented by the parent element. The name of the attribute is also identified by the name or type of the element, e.g. `hasDaughter`, or by the implicit relationship between the element and its parent element.

### **an XML attribute is a literal attribute**

When a literal attribute is modeled as an XML attribute, e.g. `<person name="John Doe"/>`, it is mapped into an attribute of the instance represented by the owner element. The name of the attribute is implied by the name of the XML attribute.

### **an XML attribute is an attribute with instance value**

An attribute whose value is another concept instance is modeled as an XML attribute, e.g. `<article followUpTo="#article103"/>`, it is mapped into an attribute of the instance represented by the owner element. The name of the attribute is implied by the name of the XML attribute. The value of the attribute is represented indirectly by the value of the XML attribute.

### **relationship between siblings**

In WSMO, there is no native support for ordered lists, but in XML it's usual to model a list by the order of sibling elements. Therefore the second item in

```
<feed>
  <item id="i1"/>
  <item id="i2"/>
</feed>
```

maps to the value of an attribute (like `nextItem`) of the instance represented by the first item. The ontological representation is a list modeled as a head-tail structure like

```
instance memberOf feed
  firstItem hasValue i1

instance i1 memberOf item
  nextItem hasValue i2

instance i2 memberOf item
```

The above list of patterns only talks about ontological attributes but explicit relation instances can be mapped similarly.

## **2.4 Mapping Language**

todo

## 3. WSMO Service Grounding to WSDL

Web Services Description Language (WSDL) [[WSDL](#)] is an XML language for describing Web services. WSDL can be used to describe Web services based on an abstract model of what the service offers. WSDL enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as "how" and "where" that functionality is offered.

WSMO provides an abstract model for describing service functionalities and behaviours, however without concrete details such as "how" and "where" that functionality and the behavior are offered.

This section takes a closer look at the model provided by WSDL, and tries to relate it to WSMO in order to see where WSDL and WSMO can benefit one from each other. The rest of this section is organized as follows: [Section 3.1](#) provides a general overview of WSDL, to make the document self-contained, and [Section 3.2](#) presents various relations that could exist between WSDL and WSMO.

### 3.1 WSDL Overview

WSDL describes Web Services in two levels - an abstract model and its concrete realization.

At the abstract level, WSDL describes a Web service in terms of the messages it sends and receives; messages are described independent of a specific wire format using a *type* system, typically XML Schema. An *operation* associates a message exchange pattern with one or more messages. A message exchange pattern identifies the sequence and cardinality of messages sent and/or received as well as who they are logically sent to and/or received from. An *interface* groups together operations without any commitment to transport or wire format.

At the concrete level, a *binding* specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. And finally, a *service* groups together endpoints that implement a common interface.

### 3.2 Relation WSMO - WSDL

WSMO grounding can be approached from two directions: elements from WSMO can be used to extend WSDL, and conversely WSDL elements can be used to extend WSMO. In the current draft we consider the case of extending WSDL with WSMO; the way how WSMO can be extended with WSDL will be investigated in future drafts.

In order to identify what parts of WSDL can be extended with WSMO, a top down approach is taken: we analyse the top-level elements (and then their sub-elements) that are part of a WSDL specification and make some proposals where WSMO would make sense to fit in.

A WSDL specification consists of four main elements: *types*, *interface*, *binding*, *service*.

#### Types

The *types* element contains schemas that describe the kinds of messages that the service will send and receive. WSDL doesn't impose any language in which these types should be defined, although support for XML Schema is required and it is currently the most used schema language. Two alternatives would make sense here:

- map between the types and WSMO ontologies (see [section 2](#) for a mechanism to map between XML Schema and WSMO), or
- write types directly using a WSMO ontology language.

[todo: some examples]

## Interface

The *interface* element describes what abstract functionality the Web service provides by describing sequences of messages that a service sends and/or receives. It does this by grouping related messages into *operations*. An operation is a sequence of input and output messages, and an interface is a set of operations.

The following situations can be envisioned:

- The WSDL interface is extended with a WSMO web service. This means the interface has the capability of the web service and complies to the choreography of that web service.
- A WSDL operation inside the interface is extended with a WSMO web service. It doesn't make sense for that web service to contain a choreography because that is already described by the message exchange pattern of the operation.

[todo: treat also the case of the inherited interfaces]

[todo: some examples]

## Binding

The *binding* element describes how to access the service; it defines the implementation details necessary to accessing the service, i.e. a concrete message format and transmission protocol for an interface. All the extensions/transitions between WSMO and WSDL are done at the abstract interface level of WSDL, thus the WSMO Binding element is not directly extendable by WSMO elements.

## Service

The *service* element describes where to access the service. To indicate that this service only offers a subset of what is offered by its interface, a WSDL service can be extended with a WSMO web service, which must be a restriction of the WSMO web service associated with the interface of the WSDL service.

# 4. Conclusions and Further Work

This document defined WSMO Grounding by presenting two independent areas of relationship between WSMO and the syntactical descriptions of a Web Services: data in WSMO ontologies has to be mapped to XML data, usually described using XML Schema, and the functional and behavioral service descriptions in WSMO have to be related to the description construct present in WSDL. Further drafts of this document will refine the current document by providing a more detailed description of WSMO Grounding.

## References

[de Bruijn 2004] J. de Bruijn (editor): *WSML/XML - An XML Syntax for WSML*, available at <http://www.wsmo.org/2004/d16/d16.3/v0.1/>

[Roman et al. 2004] D. Roman, U. Keller, H. Lausen (editors): *Web Service Modeling Ontology - Standard (WSMO - Standard)*, version 1.0 available at <http://www.wsmo.org/2004/d2/v1.0/>

[WSDL] R. Chinnici, M. Gudgin, J-J. Moreau, J. Schlimmer, S. Weerawarana (editors): *Web Services Description Language Part 1: Core Language*, W3C Last Call Working Draft available at <http://www.w3.org/TR/wsd120/>

[XMLInfoSet] J. Cowan, R. Tobin (editors): *XML Information Set*, W3C Recommendation (Second Edition), 2004, available at <http://www.w3.org/TR/2004/REC-xml-infoSet-20040204/>

**[XMLSchema]** H. Thompson, D. Beech, M. Maloney, N. Mendelsohn (editors): *XML Schema part 1: Structures*, W3C Recommendation, 2001, available at <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>

**[XSLT]** J. Clark (editor): *XSL Transformations (XSLT)*, W3C Recommendation, 1999, available at <http://www.w3.org/TR/1999/REC-xslt-19991116>

## Acknowledgement

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperonto, and h-TechSight; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate programme.

The editors would like to thank to all the [members of the WSMO working group](#) for their advice and input to this document.

## Change Log

Date	Author	Description
2005/1/17	jacek, titi	Initial version
2005/1/19	titi, jacek	first draft



\$Date: Thursday 20 January 2005 - 16:06:32\$

[webmaster](#)