

XML Schema for WSMML

Table of Contents

- [Schema Document Properties](#)
- [Global Schema Components](#)
 - [Element: **wsml**](#)
 - [Element: **importOntology**](#)
 - [Element: **usedMediator**](#)
 - [Element: **preCondition**](#)
 - [Element: **assumption**](#)
 - [Element: **postCondition**](#)
 - [Element: **effect**](#)
 - [Element: **choreography**](#)
 - [Element: **orchestration**](#)
 - [Element: **version**](#)
 - [Element: **performance**](#)
 - [Element: **reliability**](#)
 - [Element: **security**](#)
 - [Element: **scalability**](#)
 - [Element: **robustness**](#)
 - [Element: **accuracy**](#)
 - [Element: **transactional**](#)
 - [Element: **trust**](#)
 - [Element: **financial**](#)
 - [Element: **networkRelatedQoS**](#)
 - [Element: **goal**](#)
 - [Element: **nonFunctionalProperties**](#)
 - [Element: **webService**](#)
 - [Element: **capability**](#)
 - [Element: **interface**](#)
 - [Element: **ooMediator**](#)
 - [Element: **ggMediator**](#)
 - [Element: **wgMediator**](#)
 - [Element: **wwMediator**](#)
 - [Element: **ontology**](#)
 - [Element: **axiom**](#)
 - [Complex Type: **axiomType**](#)
 - [Element: **concept**](#)
 - [Element: **attribute**](#)
 - [Element: **relation**](#)
 - [Complex Type: **relationType**](#)
 - [Element: **parameter**](#)
 - [Element: **instance**](#)
 - [Element: **attributeValue**](#)
 - [Element: **relationInstance**](#)
 - [Element: **parameterValue**](#)
 - [Element: **function**](#)
 - [Complex Type: **logicalExpressionType**](#)

[top](#)

Schema Document Properties

**Target
Namespace**

<http://www.wsmo.org/2004/wsml>

Element and Attribute Namespaces	<ul style="list-style-type: none"> • Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations have no namespace.
Schema Composition	<ul style="list-style-type: none"> • This schema imports schema(s) from the following namespace(s): <ul style="list-style-type: none"> ◦ http://purl.org/dc/elements/1.1/ (at http://dublincore.org/schemas/xmls/qdc/2003/04/02/dc.xsd) • This schema includes components from the following schema document(s): <ul style="list-style-type: none"> ◦ http://www.wsmo.org/2004/d16/d16.1/v0.2/xml-syntax/wsml-expr.xsd ◦ http://www.wsmo.org/2004/d16/d16.1/v0.2/xml-syntax/wsml-identifiers.xsd
Documentation	<p>version: \$Revision: 1.6 \$ date: \$Date: 2004/11/24 14:07:30 \$ author: Jos de Bruijn This document provides an XML Schema for the XML version of the WSML syntax. The syntax presented in this document can be used for all WSML variants. The WSML variant is identified through the 'variant' attribute of the root element 'wsml'. A modular approach has been chosen when creating this schema. This schema contains the conceptual syntax. Separate schemas have been developed for the WSML identifiers and for the WSML logic expressions. These schemas are therefore included in this schema.</p>

Declared Namespaces

Prefix	Namespace
Default namespace	http://www.wsmo.org/2004/wsml
xml	http://www.w3.org/XML/1998/namespace
dc	http://purl.org/dc/elements/1.1/
xs	http://www.w3.org/2001/XMLSchema

Schema Component Representation

```

<xs:schema targetNamespace="http://www.wsmo.org/2004/wsml "
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:include
    schemaLocation="http://www.wsmo.org/2004/d16/d16.1/v0.2/xml-syntax/wsml-expr
  <xs:include
    schemaLocation="http://www.wsmo.org/2004/d16/d16.1/v0.2/xml-syntax/wsml-ider
  <xs:import namespace="http://purl.org/dc/elements/1.1/"
    schemaLocation="http://dublincore.org/schemas/xmls/qdc/2003/04/02/dc.xsd" />
  ...
</xs:schema>

```

[top](#)

Global Schema Components

Element: **wsml**

Name	wsml
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	The element 'wsml' is the preferred root element for any WSML specification. It allows the user to explicitly specify, through the 'variant' attribute, which WSML variant is used. If the WSML variant has not been specified, the application has to "guess" the variant and in the worst case, WSML-Full will be assumed.

XML Instance Representation

```
<wsml
  variant=" xs:anyURI [0..1]">
  Start Sequence [0..*]
    Allow any elements from any namespace (strict validation). [1]
  End Sequence
</wsml>
```

Schema Component Representation

```
<xs:element name="wsml">
  <xs:complexType>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:any/>
    </xs:sequence>
    <xs:attribute name="variant" type=" xs:anyURI "/>
  </xs:complexType>
</xs:element>
```

[top](#)

Element: **importOntology**

Name	importOntology
Type	wsmlID
Nilable	no
Abstract	no
Documentation	This element is used for indicating which ontologies to import. Importing ontologies is the most basic (and inflexible) modularization mechanism. The result of the import is merely the union of the axioms in the importing and the imported ontology.

XML Instance Representation

```
<importOntology> wsmlID </importOntology>
```

Schema Component Representation

```
<xs:element name="importOntology" type=" wsmlID "/>
```

[top](#)

Element: **usedMediator**

Name	usedMediator
Type	wsmlID
Nilable	no
Abstract	no
Documentation	Mediators can be used to mediate between any top-level WSMO elements, i.e. ontologies, goals, and web services. The most frequent use of mediators is to resolve heterogeneity between ontologies. Thus, mediators provide a more flexible way for modularization of ontologies.

XML Instance Representation

```
<usedMediator> wsmlID </usedMediator>
```

Schema Component Representation

```
<xs:element name="usedMediator" type=" wsmlID "/>
```

[top](#)

Element: **preCondition**

Name	preCondition
Type	axiomType
Nilable	no
Abstract	no
Documentation	A pre-condition is an axiom describing the state of the information space before executing the web service.

XML Instance Representation

```
<preCondition
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <definedBy> logicalExpressionType </definedBy> [1]
  End Sequence
</preCondition>
```

Schema Component Representation

```
<xs:element name="preCondition" type=" axiomType "/>
```

[top](#)

Element: **assumption**

Name	assumption
Type	axiomType
Nilable	no
Abstract	no
Documentation	An assumption is an axiom describing the state of the world before executing the web service.

XML Instance Representation

```
<assumption
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <definedBy> logicalExpressionType </definedBy> [1]
  End Sequence
</assumption>
```

Schema Component Representation

```
<xs:element name="assumption" type=" axiomType "/>
```

[top](#)

Element: **postCondition**

Name	postCondition
Type	axiomType
Nilable	no
Abstract	no
Documentation	A post-condition is an axiom describing the state of the information space after executing the web service.

XML Instance Representation

```
<postCondition
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <definedBy> logicalExpressionType </definedBy> [1]
  End Sequence
</postCondition>
```

Schema Component Representation

```
<xs:element name="postCondition" type=" axiomType "/>
```

[top](#)

Element: **effect**

Name	effect
Type	axiomType
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	An effect is an axiom describing the state of the world after executing the web service.

XML Instance Representation

```
<effect
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <definedBy> logicalExpressionType </definedBy> [1]
  End Sequence
</effect>
```

Schema Component Representation

```
<xs:element name="effect" type=" axiomType "/>
```

[top](#)

Element: choreography

Name	choreography
Type	wsmlID
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A reference to a choreography description.

XML Instance Representation

```
<choreography> wsmlID </choreography>
```

Schema Component Representation

```
<xs:element name="choreography" type=" wsmlID "/>
```

[top](#)

Element: orchestration

Name	orchestration
Type	wsmlID
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A reference to an orchestration description.

XML Instance Representation

```
<orchestration> wsmlID </orchestration>
```

Schema Component Representation

```
<xs:element name="orchestration" type=" wsmlID "/>
```

[top](#)

Element: **version**

Name	version
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<version  
  type=" wsmlID [0..1]"  
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})  
  [0..1]">  
  wsmlIDorLiteral-basic  
</version>
```

Schema Component Representation

```
<xs:element name="version" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: **performance**

Name	performance
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<performance  
  type=" wsmlID [0..1]"  
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})  
  [0..1]">  
  wsmlIDorLiteral-basic  
</performance>
```

Schema Component Representation

```
<xs:element name="performance" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: **reliability**

Name	reliability
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<reliability
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
  [0..1]">
  wsmlIDorLiteral-basic
</reliability>
```

Schema Component Representation

```
<xs:element name="reliability" type=" wsmlIDorLiteral " />
```

[top](#)

Element: **security**

Name	security
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<security
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
  [0..1]">
  wsmlIDorLiteral-basic
</security>
```

Schema Component Representation

```
<xs:element name="security" type=" wsmlIDorLiteral " />
```

[top](#)

Element: **scalability**

Name	scalability
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<scalability
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
```

```
[0..1]">
  wsmlIDorLiteral-basic
</scalability>
```

Schema Component Representation

```
<xs:element name="scalability" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: **robustness**

Name	robustness
Type	wsmlIDorLiteral
Nilable	no
Abstract	no

XML Instance Representation

```
<robustness
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
[0..1]">
  wsmlIDorLiteral-basic
</robustness>
```

Schema Component Representation

```
<xs:element name="robustness" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: **accuracy**

Name	accuracy
Type	wsmlIDorLiteral
Nilable	no
Abstract	no

XML Instance Representation

```
<accuracy
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
[0..1]">
  wsmlIDorLiteral-basic
</accuracy>
```

Schema Component Representation

```
<xs:element name="accuracy" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: **transactional**

Name	transactional
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<transactional
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
  [0..1]">
  wsmlIDorLiteral-basic
</transactional>
```

Schema Component Representation

```
<xs:element name="transactional" type=" wsmlIDorLiteral " />
```

[top](#)

Element: **trust**

Name	trust
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<trust
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
  [0..1]">
  wsmlIDorLiteral-basic
</trust>
```

Schema Component Representation

```
<xs:element name="trust" type=" wsmlIDorLiteral " />
```

[top](#)

Element: **financial**

Name	financial
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<financial
  type=" wsmlID [0..1]"
  kind=" xs:string (value comes from list: {'wsmlID'|'literal'})
```

```
[0..1]">
  wsmlIDorLiteral-basic
</financial>
```

Schema Component Representation

```
<xs:element name="financial" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: networkRelatedQoS

Name	networkRelatedQoS
Type	wsmlIDorLiteral
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```
<networkRelatedQoS
  type=" wsmlID [0..1]"
  kind="xs:string (value comes from list: {'wsmlID'|'literal'})
[0..1]">
  wsmlIDorLiteral-basic
</networkRelatedQoS>
```

Schema Component Representation

```
<xs:element name="networkRelatedQoS" type=" wsmlIDorLiteral "/>
```

[top](#)

Element: goal

Name	goal
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A goal specification.

XML Instance Representation

```
<goal
  name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <importOntology> ... </importOntology> [0..*]
  <ooMediator> ... </ooMediator> [0..*]
  <ggMediator> ... </ggMediator> [0..*]
  <postCondition> ... </postCondition> [0..*]
  <effect> ... </effect> [0..*]
</goal>
```

Schema Component Representation

```
<xs:element name="goal">
  <xs:complexType>
    <xs:sequence>
```

```

<xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
<xs:element ref=" importOntology " minOccurs="0"
maxOccurs="unbounded"/>
<xs:element ref=" ooMediator " minOccurs="0"
maxOccurs="unbounded"/>
<xs:element ref=" ggMediator " minOccurs="0"
maxOccurs="unbounded"/>
<xs:element ref=" postCondition " minOccurs="0"
maxOccurs="unbounded"/>
<xs:element ref=" effect " minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="name" type=" wsmlID " use="optional"/>
</xs:complexType>
</xs:element>

```

[top](#)

Element: **nonFunctionalProperties**

Name	nonFunctionalProperties
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	Any element can be used inside the non-functional properties. Non-functional properties function as annotation for the containing element. The recommended set of non-functional properties is the set of elements offered by Dublin Core. WSMML further defines one element, named 'version' and a number of web-service specific elements, named 'performance', 'reliability', 'security', 'scalability', 'robustness', 'accuracy', 'transactional', 'trust', 'financial', and 'networkRelatedQoS'.

XML Instance Representation

```

<nonFunctionalProperties>
  Start Choice [0..*]
    Allow any elements from any namespace (strict validation). [1]
  End Choice
</nonFunctionalProperties>

```

Schema Component Representation

```

<xs:element name="nonFunctionalProperties">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:any/>
    </xs:choice>
  </xs:complexType>
</xs:element>

```

[top](#)

Element: **webService**

Name	webService
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A web service specification.

XML Instance Representation

```

<webService
  name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <importOntology> ... </importOntology> [0..*]
  <ooMediator> ... </ooMediator> [0..*]
  <capability> ... </capability> [1]
  <interface> ... </interface> [0..*]
</webService>

```

Schema Component Representation

```

<xs:element name="webService">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element ref=" importOntology " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref=" ooMediator " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref=" capability "/>
      <xs:element ref=" interface " minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type=" wsmlID " use="optional"/>
  </xs:complexType>
</xs:element>

```

[top](#)

Element: **capability**

Name	capability
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A web service capability specification.

XML Instance Representation

```

<capability
  name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <importOntology> ... </importOntology> [0..*]
  <ooMediator> ... </ooMediator> [0..*]
  <wgMediator> ... </wgMediator> [0..*]
  <preCondition> ... </preCondition> [0..*]
  <assumption> ... </assumption> [0..*]
  <postCondition> ... </postCondition> [0..*]
  <effect> ... </effect> [0..*]
</capability>

```

Schema Component Representation

```
<xs:element name="capability">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="nonFunctionalProperties" minOccurs="0"/>
      <xs:element ref="importOntology" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="ooMediator" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="wgMediator" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="preCondition" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="assumption" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="postCondition" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="effect" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="wsmlID" use="optional"/>
  </xs:complexType>
</xs:element>
```

[top](#)

Element: **interface**

Name	interface
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A web service interface specification.

XML Instance Representation

```
<interface
name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <importOntology> ... </importOntology> [0..*]
  <ooMediator> ... </ooMediator> [0..*]
  <choreography> ... </choreography> [1]
  <orchestration> ... </orchestration> [1]
</interface>
```

Schema Component Representation

```
<xs:element name="interface">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="nonFunctionalProperties" minOccurs="0"/>
      <xs:element ref="importOntology" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="ooMediator" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref="choreography"/>
      <xs:element ref="orchestration"/>
    </xs:sequence>
    <xs:attribute name="name" type="wsmlID" use="optional"/>
  </xs:complexType>
</xs:element>
```

Element: ooMediator

Name	ooMediator
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	The element ooMediator is used for both declaration of ooMediators and reference to existing ooMediators. In case no elements are nested inside the ooMediator, the element is interpreted as a reference to the ooMediator indicated with the attribute 'name'. If elements are nested inside the ooMediator, it is interpreted as a specification of a new ooMediator whose identifier is specified in the attribute 'name'.

XML Instance Representation

```

<ooMediator
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <importOntology> ... </importOntology> [0..*]
    Start Choice [0..*]
      <sourceOntology> wsmlID </sourceOntology> [1]
      <sourceooMediator> wsmlID </sourceooMediator> [1]
    End Choice
    Start Choice [0..1]
      <targetOntology> wsmlID </targetOntology> [1]
      <targetGoal> wsmlID </targetGoal> [1]
      <targetWebService> wsmlID </targetWebService> [1]
      <targetMediator> wsmlID </targetMediator> [1]
    End Choice
    Start Choice [0..1]
      <goal> wsmlID </goal> [1]
      <wwMediator> wsmlID </wwMediator> [1]
      <webService> wsmlID </webService> [1]
    End Choice
  End Sequence
</ooMediator>

```

Schema Component Representation

```

<xs:element name="ooMediator">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element ref=" importOntology " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="sourceOntology" type=" wsmlID "/>
        <xs:element name="sourceooMediator" type=" wsmlID "/>
      </xs:choice>
      <xs:choice minOccurs="0">
        <xs:element name="targetOntology" type=" wsmlID "/>
        <xs:element name="targetGoal" type=" wsmlID "/>
        <xs:element name="targetWebService" type=" wsmlID "/>
        <xs:element name="targetMediator" type=" wsmlID "/>
      </xs:choice>
      <xs:choice minOccurs="0">
        <xs:element name="goal" type=" wsmlID "/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        <xs:element name="wwMediator" type=" wsmlID "/>
        <xs:element name="webService" type=" wsmlID "/>
    </xs:choice>
</xs:sequence>
<xs:attribute name="name" type=" wsmlID " use="optional"/>
</xs:complexType>
</xs:element>

```

[top](#)

Element: ggMediator

Name	ggMediator
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A ggMediator specification.

XML Instance Representation

```

<ggMediator
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <importOntology> ... </importOntology> [0..*]
    <usedMediator> ... </usedMediator> [0..*]
    Start Choice [0..*]
      <sourceGoal> wsmlID </sourceGoal> [1]
      <sourceggMediator> wsmlID </sourceggMediator> [1]
    End Choice
    Start Choice [0..1]
      <targetGoal> wsmlID </targetGoal> [1]
      <targetggMediator> wsmlID </targetggMediator> [1]
    End Choice
    Start Choice [0..1]
      <goal> wsmlID </goal> [1]
      <wwMediator> wsmlID </wwMediator> [1]
      <webService> wsmlID </webService> [1]
    End Choice
  End Sequence
</ggMediator>

```

Schema Component Representation

```

<xs:element name="ggMediator">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element ref=" importOntology " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref=" usedMediator " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="sourceGoal" type=" wsmlID "/>
        <xs:element name="sourceggMediator" type=" wsmlID "/>
      </xs:choice>
      <xs:choice minOccurs="0">
        <xs:element name="targetGoal" type=" wsmlID "/>
        <xs:element name="targetggMediator" type=" wsmlID "/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

    <xs:element name="goal" type=" wsmlID "/>
    <xs:element name="wwMediator" type=" wsmlID "/>
    <xs:element name="webService" type=" wsmlID "/>
  </xs:choice>
</xs:sequence>
  <xs:attribute name="name" type=" wsmlID " use="optional"/>
</xs:complexType>
</xs:element>

```

[top](#)

Element: **wgMediator**

Name	wgMediator
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A wgMediator specification.

XML Instance Representation

```

<wgMediator
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <importOntology> ... </importOntology> [0..*]
    <usedMediator> ... </usedMediator> [0..*]
  Start Choice [0..1]
    <sourceWebService> wsmlID </sourceWebService> [1]
    <sourcewgMediator> wsmlID </sourcewgMediator> [1]
  End Choice
  Start Choice [0..1]
    <targetGoal> wsmlID </targetGoal> [1]
    <targetggMediator> wsmlID </targetggMediator> [1]
  End Choice
  Start Choice [0..1]
    <goal> wsmlID </goal> [1]
    <wwMediator> wsmlID </wwMediator> [1]
    <webService> wsmlID </webService> [1]
  End Choice
  End Sequence
</wgMediator>

```

Schema Component Representation

```

<xs:element name="wgMediator">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element ref=" importOntology " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref=" usedMediator " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:choice minOccurs="0">
        <xs:element name="sourceWebService" type=" wsmlID "/>
        <xs:element name="sourcewgMediator" type=" wsmlID "/>
      </xs:choice>
      <xs:choice minOccurs="0">
        <xs:element name="targetGoal" type=" wsmlID "/>
        <xs:element name="targetggMediator" type=" wsmlID "/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

    <xs:choice minOccurs="0">
      <xs:element name="goal" type=" wsmlID "/>
      <xs:element name="wwMediator" type=" wsmlID "/>
      <xs:element name="webService" type=" wsmlID "/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type=" wsmlID " use="optional"/>
</xs:complexType>
</xs:element>

```

[top](#)

Element: **wwMediator**

Name	wwMediator
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A wwMediator specification.

XML Instance Representation

```

<wwMediator
  name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <importOntology> ... </importOntology> [0..*]
  <usedMediator> ... </usedMediator> [0..*]
  Start Choice [0..1]
    <sourceWebService> wsmlID </sourceWebService> [1]
    <sourcewwMediator> wsmlID </sourcewwMediator> [1]
  End Choice
  Start Choice [0..1]
    <targetWebService> wsmlID </targetWebService> [1]
    <targetwwMediator> wsmlID </targetwwMediator> [1]
  End Choice
  Start Choice [0..1]
    <goal> wsmlID </goal> [1]
    <wwMediator> wsmlID </wwMediator> [1]
    <webService> wsmlID </webService> [1]
  End Choice
</wwMediator>

```

Schema Component Representation

```

<xs:element name="wwMediator">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element ref=" importOntology " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element ref=" usedMediator " minOccurs="0"
maxOccurs="unbounded"/>
      <xs:choice minOccurs="0">
        <xs:element name="sourceWebService" type=" wsmlID "/>
        <xs:element name="sourcewwMediator" type=" wsmlID "/>
      </xs:choice>
      <xs:choice minOccurs="0">
        <xs:element name="targetWebService" type=" wsmlID "/>
        <xs:element name="targetwwMediator" type=" wsmlID "/>
      </xs:choice>
      <xs:choice minOccurs="0">

```

```

    <xs:element name="goal" type=" wsmlID "/>
    <xs:element name="wwMediator" type=" wsmlID "/>
    <xs:element name="webService" type=" wsmlID "/>
  </xs:choice>
</xs:sequence>
  <xs:attribute name="name" type=" wsmlID " use="optional"/>
</xs:complexType>
</xs:element>

```

[top](#)

Element: **ontology**

Name	ontology
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	An ontology specification.

XML Instance Representation

```

<ontology
  name=" wsmlID [0..1]">
  Start Choice [0..*]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [1]
    <importOntology> ... </importOntology> [1]
    <ooMediator> ... </ooMediator> [1]
    <concept> ... </concept> [1]
    <relation> ... </relation> [1]
    <instance> ... </instance> [1]
    <relationInstance> ... </relationInstance> [1]
    <function> ... </function> [1]
    <axiom> ... </axiom> [1]
  End Choice
</ontology>

```

Schema Component Representation

```

<xs:element name="ontology">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref=" nonFunctionalProperties "/>
      <xs:element ref=" importOntology "/>
      <xs:element ref=" ooMediator "/>
      <xs:element ref=" concept "/>
      <xs:element ref=" relation "/>
      <xs:element ref=" instance "/>
      <xs:element ref=" relationInstance "/>
      <xs:element ref=" function "/>
      <xs:element ref=" axiom "/>
    </xs:choice>
    <xs:attribute name="name" type=" wsmlID " use="optional"/>
  </xs:complexType>
</xs:element>

```

[top](#)

Element: **axiom**

Name	axiom
Type	axiomType
Nilable	no
Abstract	no

XML Instance Representation

```
<axiom
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <definedBy> logicalExpressionType </definedBy> [1]
  End Sequence
</axiom>
```

Schema Component Representation

```
<xs:element name="axiom" type=" axiomType "/>
```

[top](#)

Complex Type: **axiomType**

<i>Super-types:</i>	None
<i>Sub-types:</i>	None

Name	axiomType
Abstract	no
Documentation	An axiom reference or specification. In case no elements are nested inside the axiom, it is merely a reference to an axiom definition. Otherwise, it is an axiom specification, consisten of a number of annotations in the form of non-functional properties and the actual logical expression, contained in the definedBy keyword.

XML Instance Representation

```
<...
  name=" wsmlID [0..1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <definedBy> logicalExpressionType </definedBy> [1]
  End Sequence
</...>
```

Schema Component Representation

```
<xs:complexType name="axiomType">
  <xs:sequence minOccurs="0">
    <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
    <xs:element name="definedBy" type=" logicalExpressionType "/>
  </xs:sequence>
  <xs:attribute name="name" type=" wsmlID " use="optional"/>
</xs:complexType>
```

[top](#)

Element: **concept**

Name	concept
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A concept specification or reference. In case no elements are nested inside the concept, it is a reference to the concept identified by the name. Otherwise, it is a concept specification.

XML Instance Representation

```
<concept
  name=" wsmlID [1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <superConcept> wsmlID </superConcept> [0..*]
    <attribute> ... </attribute> [0..*]
    <definedBy> logicalExpressionType </definedBy> [0..1]
  End Sequence
</concept>
```

Schema Component Representation

```
<xs:element name="concept">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element name="superConcept" type=" wsmlID " minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref=" attribute " minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="definedBy" type=" logicalExpressionType "
        minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type=" wsmlID " use="required"/>
  </xs:complexType>
</xs:element>
```

[top](#)

Element: **attribute**

Name	attribute
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A concept can have zero or more attributes. The type of the attribute can be inferring or constraining. A constraining attribute (corresponding to 'ofType') is used to check the type of parameter values; an inferring parameter (corresponding to 'impliesType') is used to derive the type of attribute values. By default an attribute is constraining. Furthermore, an attribute can be symmetric, transitive, reflexive or the inverse of another attribute. Finally, it is possible to specify the minimal and maximal cardinality of an attribute. By default the minimal cardinality is 0 and the maximal cardinality is n (i.e. not restricted).

XML Instance Representation

```

<attribute
  name=" wsmlID [1]"
  type=" xs:string (value comes from list:
  {'constraining'|'inferring'}) [0..1]"
  symmetric=" xs:boolean [0..1]"
  transitive=" xs:boolean [0..1]"
  reflexive=" xs:boolean [0..1]"
  inverseOf=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <range> wsmlID </range> [0..1]
  <minCardinality> xs:integer </minCardinality> [0..1]
  <maxCardinality> xs:integer </maxCardinality> [0..1]
</attribute>

```

Schema Component Representation

```

<xs:element name="attribute">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element name="range" type=" wsmlID " minOccurs="0"/>
      <xs:element name="minCardinality" type=" xs:integer "
        minOccurs="0"/>
      <xs:element name="maxCardinality" type=" xs:integer "
        minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type=" wsmlID " use="required"/>
    <xs:attribute name="type" default="constraining">
      <xs:simpleType>
        <xs:restriction base=" xs:string ">
          <xs:enumeration value="constraining"/>
          <xs:enumeration value="inferring"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="symmetric" type=" xs:boolean "
      default="false"/>
    <xs:attribute name="transitive" type=" xs:boolean "
      default="false"/>
    <xs:attribute name="reflexive" type=" xs:boolean "
      default="false"/>
    <xs:attribute name="inverseOf" type=" wsmlID " use="optional"/>
  </xs:complexType>
</xs:element>

```

Element: relation

Name	relation
Type	relationType
Nilable	no
Abstract	no

XML Instance Representation

```

<relation
  name=" wsmlID [1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <superRelation> wsmlID </superRelation> [0..*]
    <parameter> ... </parameter> [0..*]
    <definedBy> logicalExpressionType </definedBy> [0..1]
  End Sequence
</relation>

```

Schema Component Representation

```
<xs:element name="relation" type=" relationType " />
```

Complex Type: relationType

<i>Super-types:</i>	None
<i>Sub-types:</i>	None

Name	relationType
Abstract	no
Documentation	A relation specification or reference. In case no elements are nested inside the relation, it is a reference to the relation identified by the name. Otherwise, it is a relation specification.

XML Instance Representation

```

<...
  name=" wsmlID [1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <superRelation> wsmlID </superRelation> [0..*]
    <parameter> ... </parameter> [0..*]
    <definedBy> logicalExpressionType </definedBy> [0..1]
  End Sequence
</...>

```

Schema Component Representation

```

<xs:complexType name="relationType">
  <xs:sequence minOccurs="0">
    <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
    <xs:element name="superRelation" type=" wsmlID " minOccurs="0"

```

```

maxOccurs="unbounded"/>
<xs:element ref=" parameter " minOccurs="0"
maxOccurs="unbounded"/>
<xs:element name="definedBy" type=" logicalExpressionType "
minOccurs="0"/>
</xs:sequence>
<xs:attribute name="name" type=" wsmlID " use="required"/>
</xs:complexType>

```

[top](#)

Element: **parameter**

Name	parameter
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A relation can have zero or more parameters. The type of the parameter can be inferring or constraining. A constraining parameters is used to check the type of parameter values; an inferring parameter is used to derive the type of parameter values. By default a parameter is constraining.

XML Instance Representation

```

<parameter
name=" wsmlID [1]"
type=" xs:string (value comes from list:
{'constraining'|'inferring'}) [0..1]">
  <range> wsmlID </range> [1]
</parameter>

```

Schema Component Representation

```

<xs:element name="parameter">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="range" type=" wsmlID "/>
    </xs:sequence>
    <xs:attribute name="name" type=" wsmlID " use="required"/>
    <xs:attribute name="type" default="constraining">
      <xs:simpleType>
        <xs:restriction base=" xs:string ">
          <xs:enumeration value="constraining"/>
          <xs:enumeration value="inferring"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

[top](#)

Element: **instance**

Name	instance
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	An instance can be member of zero or more concepts. Furthermore, zero or more attribute values can be associated with this particular instance.

XML Instance Representation

```
<instance
  name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <memberOf> wsmlID </memberOf> [0..*]
  <attributeValue> ... </attributeValue> [0..*]
</instance>
```

Schema Component Representation

```
<xs:element name="instance">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element name="memberOf" type=" wsmlID " minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref=" attributeValue " minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type=" wsmlID " use="optional"/>
  </xs:complexType>
</xs:element>
```

[top](#)

Element: **attributeValue**

Name	attributeValue
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	The value of an attribute is either the identifier of another WSMML element or a literal.

XML Instance Representation

```
<attributeValue>
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <name> wsmlID </name> [1]
  <value> wsmlIDorLiteral </value> [1..*]
</attributeValue>
```

Schema Component Representation

```
<xs:element name="attributeValue">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element name="name" type=" wsmlID "/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```

    <xs:element name="value" type=" wsmlIDorLiteral "
      maxOccurs="unbounded"/>
    <!-- can be instance, ID or literal -->
  </xs:sequence>
</xs:complexType>
</xs:element>

```

[top](#)

Element: **relationInstance**

Name	relationInstance
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no

XML Instance Representation

```

<relationInstance
  name=" wsmlID [0..1]">
  <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
  <memberOf> wsmlID </memberOf> [0..*]
  <parameterValue> ... </parameterValue> [0..*]
</relationInstance>

```

Schema Component Representation

```

<xs:element name="relationInstance">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref=" nonFunctionalProperties " minOccurs="0"/>
      <xs:element name="memberOf" type=" wsmlID " minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref=" parameterValue " minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type=" wsmlID " use="optional"/>
  </xs:complexType>
</xs:element>

```

[top](#)

Element: **parameterValue**

Name	parameterValue
Type	Locally-defined complex type
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	A parameter value refer to a parameter in case the relation has named parameters. In case the relation does not have named parameters, the name in the parameterValue is omitted. This corresponds with the symbol '_#' in the standard WSML syntax.

XML Instance Representation

```

<parameterValue>
  <name> wsmlID </name> [0..1]
  <value> wsmlID </value> [1]
</parameterValue>

```

Schema Component Representation

```

<xs:element name="parameterValue">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type=" wsmlID " minOccurs="0"/>
      <xs:element name="value" type=" wsmlID "/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

[top](#)

Element: **function**

Name	function
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A function is a special kind of relation with an additional parameter, namely the range of the function.

XML Instance Representation

```

<function
  name=" wsmlID [1]">
  Start Sequence [0..1]
    <nonFunctionalProperties> ... </nonFunctionalProperties> [0..1]
    <superRelation> wsmlID </superRelation> [0..*]
    <parameter> ... </parameter> [0..*]
    <definedBy> logicalExpressionType </definedBy> [0..1]
  End Sequence
  <range> wsmlID </range> [1]
</function>

```

Schema Component Representation

```

<xs:element name="function">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base=" relationType ">
        <xs:sequence>
          <xs:element name="range" type=" wsmlID "/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

```

[top](#)

Complex Type: **logicalExpressionType**

Super-types: [complexExpressionType](#) < **logicalExpressionType** (by extension)

Sub-types: None

Name	logicalExpressionType
Abstract	no
Documentation	A logical expression. This logical expression is a formula from a certain syntax. Currently, the only allowed syntax is the WSML-Full syntax. The XML syntax for logical expressions is inspired by the RuleML FOL (First-Order Logic) syntax (http://www.ruleml.org/fol/).

XML Instance Representation

```
<...>  
  <!-- ' complexExpressionType ' super type was not found in this  
  schema. Some elements and attributes may be missing. -->  
</...>
```

Schema Component Representation

```
<xs:complexType name="logicalExpressionType">  
  <xs:complexContent>  
    <xs:extension base=" complexExpressionType "/>  
  </xs:complexContent>  
</xs:complexType>
```

[top](#)