



# D15v0.1. Orchestration in WSMO

WSMO Working Draft 24 January 2005

**This version:**

<http://www.wsmo.org/2004/d15/v0.1/20040529/>

**Latest version:**

<http://www.wsmo.org/2004/d15/v0.1/>

**Previous version:**

<http://www.wsmo.org/2004/d15/v0.1/20040529/>

**Editors:**

Dumitru Roman  
James Scicluna

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

Copyright © 2004 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

---

## Table of contents

- [1. Introduction](#)
  - [2. State Signature](#)
  - [3. State](#)
  - [4. Guarded Transitions](#)
  - [5. Orchestration description example](#)
  - [6. Conclusions and Further Work](#)
  - [References](#)
  - [Acknowledgment](#)
- 

## 1. Introduction

The *orchestration* of a WSMO service defines how the overall functionality of the service is achieved by the cooperation of other WSMO service providers. It describes how the service works from the provider's perspective (i.e. how a service makes use of other WSMO services or goals in order to achieve its capability). This complies with the W3C definition of Web Service Orchestration [W3C Working Group]: *An orchestration defines the sequence and conditions in which one Web Service invokes other Web Services in order to realize some useful function. That is, an orchestration is the pattern of interactions that a Web Service agent must follow in order to achieve its goal.*

As in WSMO Choreography [Roman et al., 2005], the conceptual model of WSMO orchestrations is based on Abstract State Machines [Gurevich, 1995].

Listing 1: WSMO Orchestration Definition

```
ClasswsmoOrchestration
  hasStateSignature type stateSignature
  hasState type state
  hasGuardedTransitiontypeguardedTransition
```

## State Signature

A state signature defines the invariant elements of the state description.

## State

A state is described by a set of instance statements ("memberOf" statements).

## Guarded transitions

Transition rules that express changes of states by changing the set of instances.

The rest of the document is organized as follows. In Section 2 we define how a state signature is described. Section 3 provides a definition of a state. In Section 4 we describe the guarded transitions and how they differ from the ones defined in WSMO choreographies and finally in Section 5 we conclude with some remarks and future work.

## 2. State Signature

The signature of a state is defined by the elements of a WSMO ontology and it is unchanged for all the states specified in the orchestration.

Listing 2. State Signature for states in WSMO Orchestrations

```
ClassstateSignature
  hasNonFunctionalProperties type nonFunctionalProperties
  importsOntology type ontology
  usesMediator type oooMediator
  hasConceptInOrchestration type conceptInOrchestration
  hasRelationInOrchestration type relationInOrchestration
  hasFunctionInOrchestration type functionInOrchestration
  hasAxiom type axiom
  hasIdentifiers type WSMOIdentifiers
```

### Non Functional Properties

As defined in WSMO [[Roman et. al.](#)], [Section 4.1](#)

### Imported Ontologies

As defined in WSMO [[Roman et. al.](#)], [Section 4.2](#)

### Used Mediators

As defined in WSMO [[Roman et. al.](#)], [Section 4.3](#)

### Concepts in Orchestration

Concepts in orchestration are a sub-class of concepts as defined in WSMO [[Roman et. al.](#)], [Section 4.4](#). However, their non functional properties are extended to support the attribute *mode* which can take the values *static*, *controlled*, *in*, *shared* or *out*.

### Relations in Orchestration

Relations in orchestration are a sub-class of relations as defined in WSMO [[Roman et. al.](#)], [Section 4.5](#). As for concepts in orchestration, their non functional properties are extended with the attribute *mode*.

### Functions in Orchestration

Functions in orchestration are a sub-class of functions as defined in WSMO [[Roman et. al.](#)], [Section 4.6](#). As for concepts in orchestration, their non functional properties are extended with the attribute *mode*.

### Axioms

As defined in WSMO [[Roman et. al.](#)], [Section 4.8](#)

The value of the attribute *mode* for non functional properties of concepts, relations and functions in orchestrations must take one of the following values:

- *static* - meaning that the extension of the concept, relation or function can be changed. As in choreographies, if the attribute mode takes this value by default if not explicitly specified otherwise.
- *controlled* - meaning that the extension of concept, relation or function can only be changed by the service.
- *in* - meaning that the extension of concept, relation or function can only be changed by the environment. An **invocation** mechanism for this item must be provided that implements *write* access for the environment.
- *shared* - meaning that the extension of the concept, relation or function can be changed by the service and the environment. An **invocation** mechanism for this item must be provided that implements *read/write* access for the environment
- *out* - meaning that the extension of the concept, relation, or function can only be changed by the service. An **invocation** mechanism for this item must be provided that implements *read* access for the environment.

### 3. State

A state is described by a set of explicitly defined instances and values of their attributes or through a link to an instance store.

### 4. Guarded Transitions

Guarded Transitions define state changes and are expressed as follows:

**if** <cond>**theninvoke** <goal, web service or mediator ID>

A *cond* is an arbitrary WSMML axiom describing the state for which the transition rule should fire. The *invoke* keyword is used for invocation purposes of either a goal, web service or a mediator. In the case of a goal, the appropriate service (or services) has to be discovered and joined in the orchestration. If a Web Service ID is specified, then the service to be used as part of the orchestration is already known and hence there is no need to discover the goal but rather invoke it directly. The mediators can be of two types in this case, that is, *wgMediator* and *wwMediators*. A *wgMediator* is used in case a goal based on some different semantic annotation needs to be discovered. Again, once the required service (or services) is discovered, it is joined with the rest of the orchestration. We can consider the use of the *wwMediator* for two purposes. In the first case, we may want to reference an already know web service but which is described in some different notation other than WSMO. Another purpose is that of using a mediator for a possible grounding. Rather than referencing directly an invocation mechanism from the orchestration, we use a mediator to link the service to another grounded service such as WSDL.

### 5. Orchestration description example

Our service makes reservations for trips, for which the starting and ending points are located in Austria or Germany. The service request route and credit card information. In case it can provide the requested service it will perform the booking for the client.

[Listing 3](#) describes the Trip Reservation Ontology, containing concepts, relations and functions needed for making a trip reservation. This ontology uses concepts already defined in The [Dublin Core Element Set v1.1](#), the [OWL Person Ontology](#), (imported by using the [owlPersonMediator](#)), the [XML Schema Namespace](#), and the [Train Connection](#) and [Purchase](#) ontologies, the last two being developed by WSMO working group.

Listing 3. Trip Reservation Ontology

```
namespace <<http://www.wsmo.org/ontologies/tripReservationOntology#>>
  dc:<<http://purl.org/dc/elements/1.1#>>
  prs:<<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/owlPersonMe
  xsd: <<http://www.w3.org/2001/XMLSchema#>>
  tc:<<http://www.wsmo.org/ontologies/trainConnection#>>
  po: <<http://www.wsmo.org/ontologies/purchase#>>
  targetnamespace:<<http://www.wsmo.org/ontologies/tripReservationOntology#>>
```

**ontology** <<http://www.wsmo.org/ontologies/tripReservationOntology#>>

**nonFunctionalProperties**

dc:title **hasValue** "Trip Reservation Ontology"

dc:creator **hasValue** "DERI Innsbruck"

dc:description **hasValue** "an ontology for describing trip reservation re  
knowledge"

dc:publisher **hasValue** "DERI International"

dc:contributor **hasValue** "Titi"

dc:date **hasValue** "2004-10-22"

dc:type **hasValue** <<http://www.wsmo.org/2004/d2/#ontology>>

dc:format **hasValue** "text/html"

dc:language **hasValue** "en-us"

dc:rights **hasValue** <<http://deri.at/privacy.html>>

version **hasValue** "\$Revision 1.17 \$"

**endNonFunctionalProperties**

**concept** route

**nonFunctionalProperties**

dc:description **hasValue** "concept of a route between two stations"

**endNonFunctionalProperties**

sourceLocation **type** tc:station

destinationLocation **type** tc:station

**concept** reservation

**nonFunctionalProperties**

dc:description **hasValue** "concept of reservation, containing a reservat

**endNonFunctionalProperties**

reservationNumber **type** xsd:integer

reservedRoute **type** route

reservationHolder **type** prs:person

booked **type** xsd:boolean

**relation** connectionExists

**nonFunctionalProperties**

dc:description **hasValue** "connection existence relationship between tv

**endNonFunctionalProperties**

sourceLocation **type** tc:station

destincationLocation **type** tc:station

**relation** validCreditCard

**nonFunctionalProperties**

dc:description **hasValue** "credit card is valid"

```
endNonFunctionalProperties  
creditCard type prs:CreditCard
```

[Listing 4](#) contains the definition of the Trip Reservation Service. The capability offered by it is Reservation Service capability, presented in [Listing 5](#), and its orchestration (Trip Reservation Service Orchestration) is described in [Listing 6](#). Note that listings 5 to 8 are assumed to be in the same namespace (i.e. the trip reservation service namespace).

Listing 4. Trip Reservation Service definition.

```
namespace <<http://www.wsmo.org/ontologies/tripReservationService#>>  
  dc:<<http://purl.org/dc/elements/1.1#>>  
  prs:<<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/owl/PersonMediator.wsml>>  
  xsd: <<http://www.w3.org/2001/XMLSchema#>>  
  tc:<<http://www.wsmo.org/ontologies/trainConnection#>>  
  po: <<http://www.wsmo.org/ontologies/purchase#>>  
  ts: <<http://www.wsmo.org/ontologies/tripReservationOntology#>>  
  targetnamespace:<<http://www.wsmo.org/ontologies/tripReservationService#>>  
service <<http://www.wsmo.org/ontologies/tripReservationService.wsml>>  
  nonFunctionalProperties  
    dc:title hasValue "Trip Reservation Service"  
    dc:creator hasValue "DERI Innsbruck"  
    dc:description hasValue "service for online trip reservations for Austria and Germany"  
    dc:publisher hasValue "DERI International"  
    dc:contributor hasValue "Titi"  
    dc:date hasValue "2004-10-22"  
    dc:type hasValue <<http://www.wsmo.org/2004/d2/#service>>  
    dc:format hasValue "text/html"  
    dc:language hasValue "en-us"  
    dc:coverage hasValues {tc:austria, tc:germany}  
    dc:rights hasValue <<http://deri.at/privacy.html>>  
    version hasValue "$Revision 1.17 $"  
  endNonFunctionalProperties  
  capability reservationServiceCapability  
  interface reservationServiceInterface  
    choreography reservationServiceChoreography  
    orchestration reservationServiceOrchestration
```

[Listing 5](#) below presents the capability of the service by defining its precondition and postcondition. The assumption requests a valid credit card. The precondition expresses the fact that the service requires a route, for which the start and end location have to be in Austria or in Germany, and a connection exists. The postcondition expresses the fact that, in case of a successful execution of a service, a reservation is sent to the user.

Listing 5. Trip Reservation Service Capability definition.

**capability** reservationServiceCapability

**nonFunctionalProperties**

dc:title **hasValue** "Trip Reservation Service Capability"  
dc:creator **hasValue** "DERI Innsbruck"  
dc:description **hasValue** "description of the capability of the reservation  
service of providing trip reservations for Austria and Germany"  
dc:publisher **hasValue** "DERI International"  
dc:contributor **hasValue** "Titi"  
dc:date **hasValue** "2004-10-22"  
dc:format **hasValue** "text/html"  
dc:language **hasValue** "en-us"  
dc:rights **hasValue** <<http://deri.at/privacy.html>>  
version **hasValue** "\$Revision 1.17 \$"

**endNonFunctionalProperties**

**sharedVariables** ?route, ?creditCard

**assumption**

**nonFunctionalProperties**

dc:description **hasValue** "In order to purchase a trip, the credit card  
given by the user must be valid"

**endNonFunctionalProperties**

**definedBy**

validCreditCard(?creditCard)

**precondition**

**nonFunctionalProperties**

dc:description **hasValue** "the service receives a route, for which the  
start and end location have to be in Austria or in Germany,  
and a credit card which has to be valid."

**endNonFunctionalProperties**

**definedBy**

((?route **memberOf** ts:route  
?route.startLocation **hasValue** ?start  
?route.endLocation **hasValue** ?end) and  
(connectionExists(?start,?end)) and  
(?creditCard **memberOf** prs:CreditCard))

**postcondition**

**nonFunctionalProperties**

dc:description **hasValue** "in case of a successful execution of a service,  
a confirmation is sent to the user a confirmation is sent to the user"

**endNonFunctionalProperties**

**definedBy**

(?reservation **memberOf** ts:reservation) and

(?reservation.route = ?route)

**effect**

**nonFunctionalProperties**

dc:description **hasValue** "If the credit card is valid and the route is available,  
then we need to charge the credit card"

**endNonFunctionalProperties**

**definedBy**

(?creditCard **memberOf** prs:CreditCard

balance **hasValue** ?currentBalance) and

(?reservation **memberOf** ts:reservation

price **hasValue** ?reservationCost) and

(?currentBalance.amount = ?currentBalance.amount - ?reservationCost)

We now define the orchestration of the web service by specifying the state signature ([Listing 7](#)) and the guarded transitions ([Listing 8](#)).

Listing 6. Trip Reservation Service Orchestration definition.

```
orchestration reservationServiceOrchestration  
  
stateSignature reservationServiceOrchestrationSignature  
  
guardedTransitions reservationServiceOrchestrationRules
```

The signature of the states is given by elements of the WSMO Ontology, and it remains unchanged during the execution of the orchestration. All its elements are inherited from the already defined elements of the ontology, additionally having the attribute mode, described in [Section 2](#).

Listing 7. State signature in the Orchestration of the Trip Reservation Service.

```
stateSignature reservationServiceOrchestrationSignature
```

```
nonFunctionalProperties
```

```
dc:title hasValue "State signature"
```

```
dc:description hasValue "The definition of the elements that are part of  
the state signature of Trip Reservation Service orchestration."
```

```
dc:date hasValue "2004-10-22"
```

```
dc:format hasValue "text/plain"
```

```
dc:language hasValue "en-US"
```

```
version hasValue "$Revision 0.1 $"
```

```
endNonFunctionalProperties
```

```
conceptInOrchestration route subClass ts:route
```

```
nonFunctionalProperties
```

```
dc:description hasValue "a route as defined in 'ts' ontology and  
adopted for orchestration"
```

```
mode hasValue in
```

```
grounding hasValue reservationWSDL:reserveRoute
```

```
endNonFunctionalProperties
```

```
conceptInOrchestration creditcard subClass prs:CreditCard
```

```
nonFunctionalProperties
```

```
dc:description hasValue "a credit card as defined in the 'prs' ontology  
and adopted in orchestration"
```

```
mode hasValue in
```

```
grounding hasValue reservationWSDL:reserveRoute
```

```
endNonFunctionalProperties
```

```
conceptInOrchestration reservation subClass ts:reservation
```

```
nonFunctionalProperties
```

```
dc:description hasValue "concept of credit card as defined in the  
'po' ontology and adapted for orchestration-"
```

```
mode hasValue out
```

```
grounding hasValue reservationWSDL:reserveRoute
```

```
endNonFunctionalProperties
```

[Listing 8](#) presents the guarded transitions for the Trip Reservation Service orchestration. In this case we only need one guarded transition since we essentially need only to discover a service that provides a route.

Listing 8. Guarded Transitions in the Orchestration of the Trip Reservation Service.

```
guardedTransitions reservationServiceOrchestrationRules`
```

```
if((routeInstance memberOf ts:route
```

```
sourceLocation hasValue ?start
```

```

        destinationLocation hasValue ?end) and
    (?start.locatedIn = austria or ?start.locatedIn = germany) and
    (?end.locatedIn = austria or ?end.locatedIn = germany) and
    (connection (?start,?end))
then
    invoke findRoute

```

Since our service does not provide a route by itself, it needs to use a goal in order to find the service needed. This is done using the keyword *invoke* and the required goal is specified in [Listing 9](#).

Listing 9. Goal description which is invoked in order to find the required route from another service.

```

namespace <<http://www.wsmo.org/ontologies/findRouteGoal#>>
    dc:<<http://purl.org/dc/elements/1.1#>>
    po: <<http://www.wsmo.org/ontologies/purchase#>>
    prs:<<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/owlPersonMediator.wsml>>
    targetnamespace:<<http://www.wsmo.org/ontologies/findRouteGoal#>>
goal findRoute
    nonFunctionalProperties
        dc:title hasValue "Find Route"
        dc:creator hasValue "DERI Innsbruck"
        dc:description hasValue "A goal that requires a train route between Germany and Austria"
        dc:publisher hasValue "DERI International"
        dc:contributor hasValue "James"
        dc:date hasValue "2004-10-22"
        dc:format hasValue "text/html"
        dc:language hasValue "en-us"
        dc:rights hasValue <<http://deri.at/privacy.html>>
        version hasValue "$Revision 1.17 $"
    endNonFunctionalProperties
    usedMediators ooMediator
        <<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/owlPersonMediator.wsml>>
    capability
        sharedVariables ?route
        postcondition
            nonFunctionalProperties
                dc:description hasValue "the service receives a route, for which the
                    start and end location have to be in Austria or in Germany"

```

### **endNonFunctionalProperties**

#### **definedBy**

```
((?route memberOf ts:route
    ?route.startLocation hasValue ?start
    ?route.endLocation hasValue ?end)) and
(?start = Germany or ?start = Austria) and
(?end = Germany or ?end = Austria)
```

#### **effect**

#### **nonFunctionalProperties**

```
dc:description hasValue "There should be a trade if a route is available"
```

### **endNonFunctionalProperties**

#### **definedBy**

```
((?trade memberOf po:trade
    item hasValue ?route
    buyer hasValue ?routeBuyer) and
(?routeBuyer.creditCard hasValue prs:CreditCard))
```

## 6. Conclusions and Further Work

This document presented a preliminary draft related to orchestration in WSMO. It is yet based on a formal model which is Abstract State Machines. There are many aspects in orchestration which are still being considered. When invoking other services and/or goals, there has to be a better specification of how the discovered services are to be combined in the orchestration. This could be as trivial as joining the guarded transitions and concepts of the ASMs describing the relevant web services but it can also be a very complex operation if mediation and parameter binding matching is needed. Another point is that of defining proper semantics to the *invoke* keyword, that is, a further detailed description of its behaviour with respect to the different types of parameters it can take should be defined.

## References

[AsmL] Abstract State Machine Language. Available at <http://research.microsoft.com/fse/asml/>

[Gurevich, 1995] E. Börger (ed.): Yuri Gurevich: Evolving Algebras 1993: Lipari Guide, Specification and Validation Methods, Oxford University Press, 1995, 9--36.

[Roman et al., 2004] D. Roman, H. Lausen, U. Keller (eds.): Web Service Modelling Ontology (WSMO), WSMO deliverable D2 version 1.1. available at <http://www.wsmo.org/2004/d2/v1.1/>

[Roman et al., 2005] D. Roman, J. Scicluna (eds.): Choreography in WSMO, WSMO deliverable D14 version 0.1. available at <http://www.wsmo.org/2004/d14/v0.1/20041112/>

[W3C Working Group] H. Haas, A. Brown (eds.): Web Services Glossary, W3C Working Group Note 11 February 2004 available at <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>

## Acknowledgement

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

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

[webmaster](#)