WS-Coordination Overview

by Simon Zambrovski

Table of contents

1. Abstract

Note:
This content is based on Arjuna Technology Report "Introducing WS-Coordination" (see Resources)

In general terms, coordination is the act of one entity (the known as the coordinator) disseminating information to a number of participants for some domain-specific reason. This reason could be in order to reach consensus on a decision like in a distributed transaction protocol, or simply to guarantee that all participants obtain a specific message, as occurs in a reliable multicast environment. When parties are being coordinated, information known as the coordination context is propagated to tie together operations which are logically part of the same coordinated work or activity.

This context information may flow with normal application messages, or may be an explicit part of a message exchange and is specific to the type of coordination being performed. For example a security coordination service will propagate differently formed contexts than a transaction coordinator. The fundamental idea underpinning WS-Coordination is that there is indeed a generic need for propagating context information in a Web services environment which is a shared requirement irrespective of the applications being executed. The WS-Coordination specification defines a framework that allows different coordination protocols to be plugged-in to coordinate work between clients, services and participants. Whatever coordination protocol is used, and in whatever domain it is deployed, the same generic requirements are present:

- Instantiation (or activation) of a new coordinator for the specific coordination protocol, for a particular application instance
- Registration of participants with the coordinator, such that they will receive that coordinator's protocol messages during (some part of) the application's lifetime
- Propagation of contextual information between Web services that comprise the application
- An entity to drive the coordination protocol through to completion

2. Activation

The WS-Coordination framework exposes an Activation Service which supports the creation of coordinators for specific protocols and their associated contexts. The process of invoking an activation service is done asynchronously, and so the specification defines both the interface of the activation service itself, and that of the invoking service, so that the activation service can call back to deliver the results of the activation - namely a context that identifies the protocol type and coordinator location.

```
<!-- Activation Service portType Declaration -->
<wsdl:portType name="ActivationCoordinatorPortType">
        <wsdl:operation name="CreateCoordinationContext">
                <wsdl:input message="wscoor:CreateCoordinationContext"/>
        </wsdl:operation>
</wsdl:portType>

<!-- Activation Requester portType Declaration -->
<wsdl:portType name="ActivationRequesterPortType">
        <wsdl:operation name="CreateCoordinationContextResponse">
                <wsdl:input
message="wscoor:CreateCoordinationContextResponse"/>
        </wsdl:operation>
        <wsdl:operation name="Error">
                <wsdl:input message="wscoor:Error"/>
        </wsdl:operation>
</wsdl:portType>
```

3. Registration

Once a coordinator has been instantiated and a corresponding context created by the activation service, a Registration Service is created and exposed. This service allows participants to register to receive protocol messages associated with a particular coordinator. Like the activation service, the registration service assumes asynchronous communication and so specifies WSDL for both registration service and registration requester

```
<!-- Registration Service portType Declaration -->
<wsdl:portType name="RegistrationCoordinatorPortType">
        <wsdl:operation name="Register">
                <wsdl:input message="wscoor:Register"/>
        </wsdl:operation>
</wsdl:portType>

<!-- Registration Requester portType Declaration -->
<wsdl:portType name="RegistrationRequesterPortType">
        <wsdl:operation name="RegisterResponse">
                <wsdl:input message="wscoor:RegisterResponse"/>
        </wsdl:operation>
        <wsdl:operation name="Error">
                <wsdl:input message="wscoor:Error"/>
        </wsdl:operation>
</wsdl:portType>
```

When a participant is registered with a coordinator through the registration service, it receives messages that the coordinator sends (for example, 'completed' and 'compensate' messages if a BusinessAgreement protocol is used); where the coordinator's protocol supports it, participants can also send messages back to the coordinator.

4. Completion

The role of terminator is generally played by the client application, which at an appropriate point will ask the coordinator to perform its particular coordination function with any registered participants - to drive to protocol through to its completion. On completion, the client application may be informed of an outcome for the activity which may vary from simple succeeded-failed notification through to complex structured data detailing the activity's status.

5. Context

The context is critical to coordination since it contains the information necessary for services to participate in the protocol. It provides the glue to bind all of the application's constituent Web services together into a single coordinated application whole. Since WS-Coordination is a generic coordination framework, contexts have to be tailored to meet the needs of specific coordination protocols that are plugged into the framework. The format of a WS-Coordination context is specifically designed to be third-party extensible and its contents are as follows:

- A coordination identifier with guaranteed global uniqueness for an individual coordinator in the form of a URI
- An address of a registration service endpoint where parties receiving a context can register participants into the protocol
- A TTL value which indicates for how long the context should be considered valid
- Extensible protocol-specific information particular to the actual coordination protocol supported by the coordinator

While the first three points are common sense, the fourth is somewhat more interesting. Since WS-Coordination is generic, it is of very little use to applications without augmentation, and this is reflected in the part of the WS-Coordination XML schema for contexts.

```
<xs:complexType name="CoordinationContextType" abstract="false">
       <xs:complexContent>
              <xs:extension base="wsu:ContextType">
                     <xs:sequence>
                            <xs:element name="CoordinationType"
type="xs:anyURI" />
                            <xs:element name="RegistrationService"
type="wsu:PortReferenceType" />
                            <xs:any namespace="##any"
processContents="lax"
                                                   minOccurs="0"
maxOccurs="unbounded" />
                     </xs:sequence>
```

```
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

6. Summary

WS-Coordination looks set to become the adopted standard for activity coordination on the Web. Out of the box, WS-Coordination provides only activity and registration services, and is extended through protocol plug-ins that provide domain-specific coordination facilities. In addition to its generic nature, the WS-Coordination model is also scales efficiently via interposed coordination which allows arbitrary collections of Web services to coordinate their operation in a straightforward and scalable manner. Though WS-Coordination is generically useful, at time of writing only two protocols that leverages WS-Coordination has been made public: WS-AtomicTransactions and WS-BusinessActivities.

7. Resources

- WS-Coordination specification
- Arjuna Technology Report : Introducing WS-Coordination