Web Service Coordination (WS-Coordination)

1) The WS-Coordination specification describes a framework for a coordination service (or coordinator) which consists of these component services:

- An activation service with an operation that enables an application to create a coordination instance or context. An application sends a CreateCoordinationContext message and the activation service returns a CreateCoordinationContextResponse message.

- A registration service with an operation that enables an application to register for coordination protocols. The interface provided to an application registering for an activity and for an interposed coordinator registering for an activity is the same. The requester sends a Register message and the coordinator's registration service responds a RegistrationResponse message.

- A coordination type-specific set of coordination protocols.

2) The CoordinationContext is used by applications to pass Coordination information to parties involved in an activity. CoordinationContext elements are propagated to parties which may need to register Participants for the activity. Context propagation may be accomplished using application-defined mechanisms – e.g. as a header element of a SOAP application message sent to such parties. (Conveying a context in an application message is commonly referred to as flowing the context.) A CoordinationContext contains the activity identifier, its coordination type that represents the collection of behaviors supported by the activity and a registration service endpoint reference that participants can use to register for one or more of the protocols supported by that activity's coordination type.
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    ...<wscoor:CoordinationContext
      xmlns:wsa="http://www.w3.org/2005/08/addressing"
      xmlns:wscoor="http://docs.oasis-open.org/ws-tx/wscoor/2006/06"
      xmlns:myApp="http://www.example.com/myApp"
      soap:mustUnderstand="true">
      <wscoor:Identifier>
        http://Fabrikam123.com/SS/1234
      </wscoor:Identifier>
      <wscoor:Expires>3000</wscoor:Expires>
      <wscoor:CoordinationType>
        http://docs.oasis-open.org/ws-tx/wsat/2006/06
      </wscoor:CoordinationType>
      <wscoor:RegistrationService>
        <wsa:Address>
          http://Business456.com/mycoordinator/registration
        </wsa:Address>
        <wsa:ReferenceParameters>
          <myApp:BetaMark> ... </myApp:BetaMark>
          <myApp:EBDCode> ... </myApp:EBDCode>
        </wsa:ReferenceParameters>
      </wscoor:RegistrationService>
      <myApp:IsolationLevel>
        RepeatableRead
      </myApp:IsolationLevel>
      </wscoor:CoordinationContext>
    ...</soap:Header>
  </soap:Body>
</soap:Envelope>

Table 1: Sample Coordination Context in a SOAP message header

/CoordinationContext/Identifier
Unique identifier of the activity.

/CoordinationContext/CoordinationType
This provides the unique identifier for the desired coordination type for the activity (e.g., a URI to the Atomic Transaction coordination type).

/CoordinationContext/Expires (optional)
The expiration for the returned CoordinationContext expressed as an unsigned integer in milliseconds.

/CoordinationContext/CurrentContext (optional)
If absent, the activation service creates a coordination context representing a new, independent activity. If present, the Activation Service creates a coordination context representing a new activity which is related to the existing activity identified by the current coordination context contained in this element. Some examples of potential uses of this type of relationship include interposed subordinate coordination, protocol bridging and coordinator replication.
This provides an endpoint reference of the registration service. Participants use this endpoint reference to register for an activity.

Extensibility elements may be used to convey additional information.

When a context is exchanged as a SOAP header, the mustUnderstand attribute must be present and its value must be true. In this way, it is assured that all participants support the coordination protocol.

**Web Service Atomic Transaction (WS-AtomicTransaction) and Web Service Business Activity (WS-BusinessActivity)**

1)  
   - Application  
     - Creates a coordination context for the atomic transaction coordination type (activation service).  
     - Registers for the completion protocol (registration service).  
   - Coordinator  
     - Offers the activation service, the registration service, and the completion protocol service.  
   - Airline web service, hotel web service, and car rental agency web service  
     - Register for the durable two-phase commit protocol (registration service) and optionally for the volatile two-phase commit protocol if required.
Application

Activation
Service

Registration
Service

Completion
Protocol
Service

Airline
Web
Service

Hotel
Web
Service

Car Rental
Agency Web
Service

CreateCoordinationContext

CreateCoordinationContextResponse

Register (Completion Protocol)

RegisterResponse

book a flight

Register (Durable 2PC)

RegisterResponse

book a hotel

Register (Durable 2PC)

RegisterResponse

book a rental car

Register (Durable 2PC)

RegisterResponse

Commit

Prepare

Prepare

Prepared

Prepared

Commit

Commit

Committed

Committed

Committed
The overall transaction fails.
3) Atomic transactions require a high level of trust between participants and are short in duration.

- A single system can maliciously abort the overall transaction.
- Holding exclusive locks across trust domains is dangerous.
- In the sample scenario the airplane seats are “reserved” and may still be canceled by the client later without charge. A company might not be willing to offer these booking conditions.
4)

CreateCoordinationContext

CreateCoordinationContextResponse

book a flight

Register (BusinessAgreementWithParticipantCompletion)

RegisterResponse

book a hotel

Register (BusinessAgreementWithParticipantCompletion)

RegisterResponse

Completed (hotel booked)

book a rental car

Register (BusinessAgreementWithParticipantCompletion)

RegisterResponse

Completed (rental car booked)

Completed (flight booked)

Close

Closed

Closed

Close

Closed
They progress of the business activity is governed by business logic.
<table>
<thead>
<tr>
<th>Duration of activities</th>
<th>WS-AtomicTransaction</th>
<th>WS-BusinessActivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short transactions</td>
<td></td>
<td>Long-running activities</td>
</tr>
<tr>
<td>Trust</td>
<td>High level of trust between participants</td>
<td>Spans trust domains</td>
</tr>
<tr>
<td>Undo action</td>
<td>Rollback</td>
<td>Compensation</td>
</tr>
<tr>
<td>Completion protocol</td>
<td>2PC</td>
<td>Governed by business logic. Mixed outcome is possible.</td>
</tr>
</tbody>
</table>

**Web-based Information Systems**

1) Electronic guest book of a private web site
   - Low access rate
   - No user authentication or authorization required
   - Security is not an issue

A server-side approach such as CGI programs, Server APIs, Server-Side-Includes seems reasonable. The guestbook does not need to scale up to many users; therefore a two-tier architecture is sufficient.

2) Online book store
   - High access rate
   - Shopping cart requires session tracking across http request
   - User may need to register to purchase books
   - Online payment demands for security

A server-side approach such as Java Servlets in combination with an application server seems reasonable here. Note, that session tracking is required here!

3) An intranet application with a rich user interface
   - Rich user interface (frequent page reloads are painful)
   - Trusted environment

Rich user interfaces can easily be realized with Java Applets (using Java Swing for instance). Signed applets can access local system resources.