OASIS

Web Services Coordination (WS-Coordination) 1.1

Committee Draft 01, March 15, 2006

Document Identifier:

wstx-wscoor-1.1-spec-cd-01

Location:

http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-cd-01.pdf

Technical Committee:

OASIS WS-TX TC

Chair(s):

Eric Newcomer, Iona
Ian Robinson, IBM

Editor(s):

Max Feingold, Microsoft

Abstract:

This specification (WS-Coordination) describes an extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed activities.

The framework defined in this specification enables an application service to create a context needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment.

Additionally this specification describes a definition of the structure of context and the requirements for propagating context between cooperating services.

Status:

This document is published by the WS-TX TC as a "committee draft".

This document was last revised or approved by the WS-TX TC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at www.oasis-open.org/committees/ws-tx .

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (www.oasis-open.org/committees/ws-tx/ipr.php).

The non-normative errata page for this specification is located at www.oasis-open.org/committees/ws-tx .

Notices

Table of Contents

1 Introduction

The current set of Web service specifications [WSDL, SOAP] defines protocols for Web service interoperability. Web services increasingly tie together a large number of participants forming large distributed computational units – we refer to these computation units as activities.

The resulting activities are often complex in structure, with complex relationships between their participants. The execution of such activities often takes a long time to complete due to business latencies and user interactions.

This specification defines an extensible framework for coordinating activities using a coordinator and set of coordination protocols. This framework enables participants to reach consistent agreement on the outcome of distributed activities. The coordination protocols that can be defined in this framework can accommodate a wide variety of activities, including protocols for simple short-lived operations and protocols for complex long-lived business activities.

Note that the use of the coordination framework is not restricted to transaction processing systems; a wide variety of protocols can be defined for distributed applications.

1.1 Model

This 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.

A Registration service with an operation that enables an application to register for coordination protocols.

A coordination type-specific set of coordination protocols.

This is illustrated below in Figure 1.

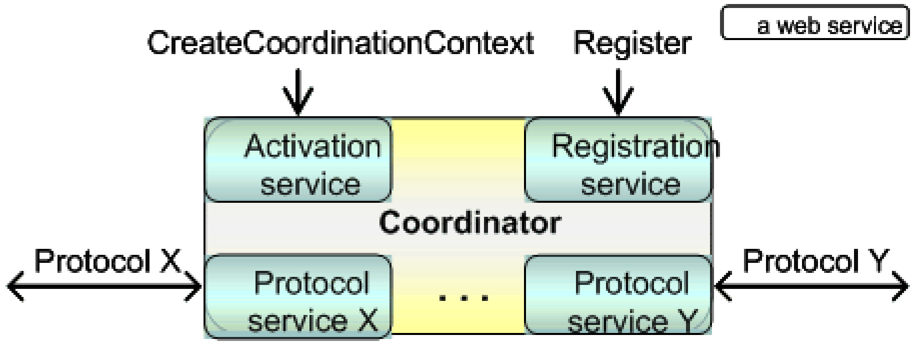


Figure 1: A Coordinator

Applications use the Activation service to create the coordination context for an activity. Once a coordination context is acquired by an application, it is then sent by whatever appropriate means to another application.

The context contains the necessary information to register into the activity specifying the coordination behavior that the application will follow.

Additionally, an application that receives a coordination context may use the Registration service of the original application or may use one that is specified by an interposing, trusted coordinator. In this manner an arbitrary collection of Web services may coordinate their joint operation.

1.2 Composable Architecture

By using the SOAP [SOAP] and WSDL [WSDL] extensibility model, SOAP-based and WSDL-based specifications are designed to be composed with each other to define a rich Web services environment. As such, WS-Coordination by itself does not define all the features required for a complete solution. WS-Coordination is a building block that is used in conjunction with other specifications and application-specific protocols to accommodate a wide variety of protocols related to the operation of distributed Web services.

The Web service protocols defined in this specification should be used when interoperability is needed across vendor implementations, trust domains, etc. Thus, the Web service protocols defined in this specification can be combined with proprietary protocols within the same application.

1.3 Extensibility

The specification provides for extensibility and flexibility along two dimensions. The framework allows for:

The publication of new coordination protocols.

The selection of a protocol from a coordination type and the definition of extension elements that can be added to protocols and message flows.

Extension elements can be used to exchange application-specific data on top of message flows already defined in this specification. This addresses the need to exchange such data as isolation-level supported signatures or other information related to business-level coordination protocols. The data can be logged for auditing purposes, or evaluated to ensure that a decision meets certain business-specific constraints.

To understand the syntax used in this specification, you should be familiar with the WSDL [WSDL] specification, including its HTTP and SOAP binding styles. All WSDL port type definitions provided here assume the existence of corresponding SOAP and HTTP bindings.

Terms introduced in this specification are explained in the body of the specification and summarized in the glossary.

1.4 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

Namespace URIs of the general form "some-URI" represents some application-dependent or context-dependent URI as defined in RFC2396 [URI].

This specification uses an informal syntax to describe the XML grammar of the XML fragments below:

- The syntax appears as an XML instance, but the values indicate the data types instead of values.
- Element names ending in "..." (such as <element.../> or <element...>) indicate that elements/attributes irrelevant to the context are being omitted.
- Attributed names ending in "..." (such as name=...) indicate that the values are specified below.
- Grammar in bold has not been introduced earlier in the document, or is of particular interest in an example.
- <-- description --> is a placeholder for elements from some "other" namespace (like ##other in XSD).
- Characters are appended to elements, attributes, and <!-- descriptions --> as follows: "?" (0 or 1), "*" (0 or more), "+" (1 or more). The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to the "?", "*", or "+" characters.
- The XML namespace prefixes (defined below) are used to indicate the namespace of the element being defined.
- Examples starting with <?xml contain enough information to conform to this specification; others examples are fragments and require additional information to be specified in order to conform.

XSD schemas and WSDL definitions are provided as a formal definition of grammars [xml-schema1] [WSDL].

1.5 Namespace

The XML namespace [XML-ns] URI that MUST be used by implementations of this specification is:

```
          http://docs.oasis-open.org/ws-tx/wscoor/2006/03
```

The namespace prefix "wscoor" used in this specification is associated with this URI.

The following namespaces are used in this document:

Prefix	Namespace
S	http://www.w3.org/2003/05/soap-envelope
wscoor	http://docs.oasis-open.org/ws-tx/wscoor/2006/03
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing

If an action URI is used, then the action URI MUST consist of the coordination namespace URI concatenated with the '/' character and the element name. For example:

```
          http://docs.oasis-open.org/ws-tx/wscoor/2006/03/Register
```

1.6 XSD and WSDL Files

The following links hold the XML schema and the WSDL declarations defined in this document.

http://docs.oasis-open.org/ws-tx/wscoor/2006/03/wscoor.xsd

http://docs.oasis-open.org/ws-tx/wscoor/2006/03/wscoor.wsdl

Soap bindings for the WSDL documents defined in this specification MUST use "document" for the *style* attribute.

1.7 Coordination Protocol Elements

The protocol elements define various extensibility points that allow other child or attribute content. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. If a receiver does not recognize an extension, the receiver SHOULD ignore the extension.

1.8 Normative References

1.9 Non-normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

[URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[XML-ns] W3C Recommendation, "Namespaces in XML," 14 January 1999.

[XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001.

[XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.

[WSADDR] Web Services Addressing (WS-Addressing), Microsoft, IBM, Sun, BEA Systems, SAP, Sun, August 2004

[WSAT] Web Services Atomic Transaction (WS-AtomicTransaction) http://docs.oasis-open.org/ws-tx/wsat/2006/03.

[WSDL] Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[WSPOLICY] Web Services Policy Framework (WS-Policy), VeriSign, Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004

[WSSec] OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)"

[WSSecPolicy] Web Services Security Policy Language (WS-SecurityPolicy), Microsoft, VeriSign, IBM, and RSA Security Inc., July 2005

[WSSecConv Web Services Secure Conversation Language (WS-SecureConversation), OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, IBM, VeriSign, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, February 2005

[WSTrust] Web Services Trust Language (WS-Trust), OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, February 2005

2 Coordination Context

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, 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 provides access to a coordination registration service, a coordination type, and relevant extensions.

The following is an example of a CoordinationContext supporting a transaction service:

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
    <S:Header>
        . . .
        <wscoor:CoordinationContext
            xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
            xmlns:wscoor="http://docs.oasis-open.org/ws-tx/wscoor/2006/03"
            xmlns:myApp="http://fabrikam123.com/myApp"
            S: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/2006/03
            </wscoor:CoordinationType>
            <wscoor:RegistrationService>
                <wsa:Address>
                 http://Business456.com/mycoordinationservice/registration
                </wsa:Address>
                <wsa:ReferenceProperties>
                  <myApp:BetaMark> ... </myApp:BetaMark>
                  <myApp:EBDCode> ... </myApp:EBDCode>
                </wsa:ReferenceProperties>
            </wscoor:RegistrationService>
            <myApp:IsolationLevel>
                  RepeatableRead
            </myApp:IsolationLevel>
        </wscoor:CoordinationContext>
        . . .
    </S:Header>
    </S:Body>
        . . .
    </S:Body >
</S:Envelope>

```

When an application propagates an activity using a coordination service, applications MUST include a Coordination context in the message.

When a context is exchanged as a SOAP header, the mustUnderstand attribute MUST be present and its value MUST be true.

3 Coordination Service

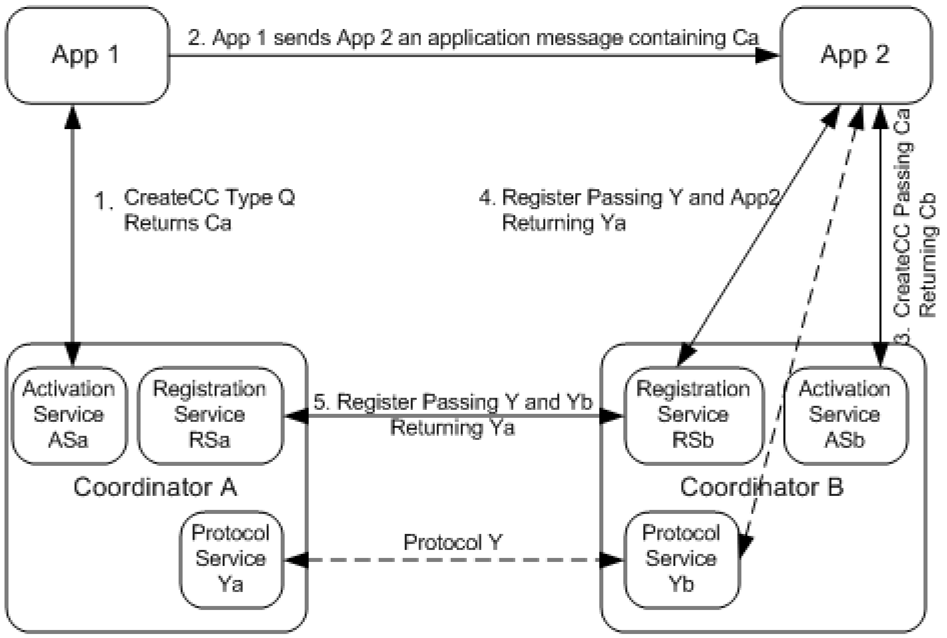
The Coordination service (or coordinator) is an aggregation of the following services:

- Activation service: Defines a CreateCoordinationContext operation that allows a CoordinationContext to be created. The exact semantics are defined in the specification that defines the coordination type. The Coordination service MAY support the Activation service.
- Registration service: Defines a Register operation that allows a Web service to register to participate in a coordination protocol. The Coordination service MUST support the Registration service.
- A set of coordination protocol services for each supported coordination type. These are defined in the specification that defines the coordination type.

Figure 2 illustrates an example of how two application services (App1 and App2) with their own coordinators (CoordinatorA and CoordinatorB) interact as the activity propagates between them. The protocol Y and services Ya and Yb are specific to a coordination type, which are not defined in this specification.

1. App1 sends a CreateCoordinationContext for coordination type Q, getting back a Context Ca that contains the activity identifier A1, the coordination type Q and an Endpoint Reference to CoordinatorA's Registration service RSa.
2. App1 then sends an application message to App2 containing the Context Ca.
3. App2 prefers to use CoordinatorB instead of CoordinatorA, so it uses CreateCoordinationContext with Ca as an input to interpose CoordinatorB. CoordinatorB creates its own CoordinationContext Cb that contains the same activity identifier and coordination type as Ca but with its own Registration service RSb.
4. App2 determines the coordination protocols supported by the coordination type Q and then Registers for a coordination protocol Y at CoordinatorB, exchanging Endpoint References for App2 and the protocol service Yb. This forms a logical connection between these Endpoint References that the protocol Y can use.
5. This registration causes CoordinatorB to decide to immediately forward the registration onto CoordinatorA's Registration service RSa, exchanging Endpoint References for Yb and the protocol service Ya. This forms a logical connection between these Endpoint References that the protocol Y can use.

Figure 2: Two applications with their own coordinators


App 1
2. App 1 sends App 2 an application message containing Ca
App 2
1. CreateCC Type Q Returns Ca
4. Register Passing Y and App2 Returning Ya
3. CreateCC Passing Ca Returning Cb
Activation Service ASa
Registration Service RSa
5. Register Passing Y and Yb Returning Ya
Registration Service RSb
Activation Service ASb
Coordinator A
Coordinator B
Protocol Service Ya
Protocol Y
Protocol Service Yb

It should be noted that in this example several actions are taken that are not required by this specification, but which may be defined by the coordination type specification or are implementation or configuration choices. Specifications of coordination types and coordination protocols that need to constrain the sub-coordination behavior of implementations should state these requirements in their specification.

3.1 Activation Service

The Activation service creates a new activity and returns its coordination context.

An application sends:

CreateCoordinationContext

The structure and semantics of this message is defined in Section 3.1.1.

The activation service returns:

CreateCoordinationContextResponse

The structure and semantics of this message is defined in Section 3.1.2

3.1.1 CreateCoordinationContext

This request is used to create a coordination context that supports a coordination type (i.e., a service that provides a set of coordination protocols). This command is required when using a network-accessible Activation service in heterogeneous environments that span vendor implementations. To fully understand the semantics of this operation it is necessary to read the specification where the coordination type is defined (e.g. WS-AtomicTransaction).

The following pseudo schema defines this element:

```
<CreateCoordinationContext ...>
    <Expires> ... </Expires>?
    <CurrentContext> ... </CurrentContext>?
    <CoordinationType> ... </CoordinationType>
    ...
</CreateCoordinationContext>

```

/CreateCoordinationContext/CoordinationType

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

/CreateCoordinationContext/Expires

Optional. The expiration for the returned CoordinationContext expressed as an unsigned integer in milliseconds.

/CreateCoordinationContext/CurrentContext

Optional. The current CoordinationContext. This may be used for a variety of purposes including recovery and subordinate coordination environments.

/CreateCoordinationContext /{any}

Extensibility elements may be used to convey additional information.

/CreateCoordinationContext /@{any}

Extensibility attributes may be used to convey additional information.

A CreateCoordinationContext message can be as simple as the following example.

```
<CreateCoordinationContext>
    <CoordinationType>
         http://docs.oasis-open.org/ws-tx/wsat/2006/03    </CoordinationType>
</CreateCoordinationContext>
```

3.1.2 CreateCoordinationContextResponse

This returns the CoordinationContext that was created.

The following pseudo schema defines this element:

```
<CreateCoordinationContextResponse ...>
    <CoordinationContext> ... </CoordinationContext>
    ...
</CreateCoordinationContextResponse>
```

/CreateCoordinationContext/CoordinationContext

This is the created coordination context.

/CreateCoordinationContext /{any}

Extensibility elements may be used to convey additional information.

/CreateCoordinationContext /@{any}

Extensibility attributes may be used to convey additional information.

The following example illustrates a response:

```
<CreateCoordinationContextResponse>
    <CoordinationContext>
        <Identifier>
             http://Business456.com/tm/context1234
        </Identifier>
        <CoordinationType>
             http://docs.oasis-open.org/ws-tx/wsat/2006/03
        </CoordinationType>
        <RegistrationService>
             <wsa:Address>
                  http://Business456.com/tm/registration
             </wsa:Address>
             <wsa:ReferenceProperties>
               <myapp:PrivateInstance>
                  1234
               </myapp:PrivateInstance>
             </wsa:ReferenceProperties>
        </RegistrationService>
    </CoordinationContext>
</CreateCoordinationContextResponse>
```

3.2 Registration Service

Once an application has a coordination context from its chosen coordinator, it can register for the activity. 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:

Register

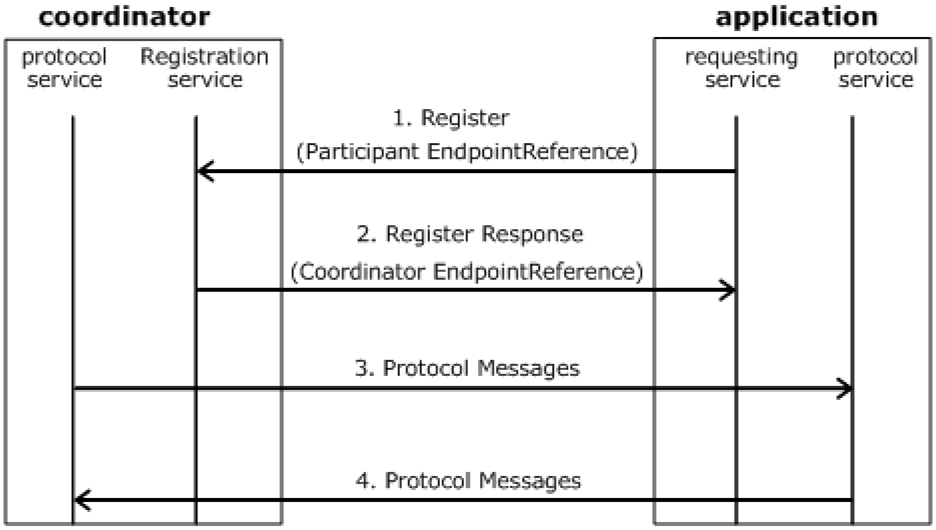
The syntax and semantics of this message are defined in Section 3.2.1.

The coordinator's registration service responds with:

Registration Response

The syntax and semantics of this message are defined in Section 3.2.2.

Figure 3: The usage of Endpoint References during registration



In Figure 3, the coordinator provides the Registration Endpoint Reference in the CoordinationContext during the CreateCoordinationContext operation. The requesting service receives the Registration service Endpoint Reference in the CoordinationContext in an application message.

1.) The Register message targets this Endpoint Reference and includes the participant protocol service Endpoint Reference as a parameter.

2.) The RegisterResponse includes the coordinator's protocol service Endpoint Reference.

3. & 4.) At this point, both sides have the Endpoint References of the other's protocol service, so the protocol messages can target the other side.

These Endpoint References may contain (opaque) wsa:ReferenceProperties to fully qualify the target protocol service endpoint. According to the mapping rules defined in the WS-Addressing specification, all such reference properties must be copied literally as headers in any message targeting the endpoint.

A Registration service is not required to detect duplicate Register requests and MAY treat each Register message as a request to register a distinct participant.

A participant MAY send multiple Register requests to a Registration service. For example, it may retry a Register request following a lost RegisterResponse, or it may fail and restart after registering successfully but before performing any recoverable work.

If a participant sends multiple Register requests for the same activity, the participant MUST be prepared to correctly handle duplicate protocol messages from the coordinator. One simple strategy for accomplishing this is for the participant to generate a unique reference parameter for each participant Endpoint Reference that it provides in a Register request. The manner in which the participant handles duplicate protocol messages depends on the specific coordination type and coordination protocol.

3.2.1 Register Message

The Register request is used to do the following:

- Participant selection and registration in a particular Coordination protocol under the current coordination type supported by the Coordination Service.
- Exchange Endpoint References. Each side of the coordination protocol (participant and coordinator) supplies an Endpoint Reference.

Participants can register for multiple Coordination protocols by issuing multiple Register operations. WS-Coordination assumes that transport protocols provide for message batching if required.

The following pseudo schema defines this element:

```
<Register ...>
    <ProtocolIdentifier> ... </ProtocolIdentifier>
    <ParticipantProtocolService> ... </ParticipantProtocolService>
    ...
</Register>
```

/Register/ProtocolIdentifier

This URI provides the identifier of the coordination protocol selected for registration.

/Register/ParticipantProtocolService

The Endpoint Reference that the registering participant wants the coordinator to use for the Coordination protocol (See WS-Addressing [WSADDR]).

/Register/{any}

Extensibility elements may be used to convey additional information.

/ Register/@{any}

Extensibility attributes may be used to convey additional information.

The following is an example registration message:

```
<Register>
    <ProtocolIdentifier>
        http://docs.oasis-open.org/ws-tx/wsat/2006/03/Volatile2PC
    </ProtocolIdentifier>
    <ParticipantProtocolService>
        <wsa:Address>
             http://Adventure456.com/participant2PCservice
        </wsa:Address>
        <wsa:ReferenceProperties>
            <BetaMark> AlphaBetaGamma </BetaMark>
        </wsa:ReferenceProperties>
    </ParticipantProtocolService>
</Register>
```

3.2.2 RegistrationResponse Message

The response to the registration message contains the coordinators Endpoint Reference.

The following pseudo schema defines this element:

```
<RegisterResponse ...>
    <CoordinatorProtocolService> ... </CoordinatorProtocolService>
    ...
</RegisterResponse>
```

/RegisterResponse/CoordinatorProtocolService

The Endpoint Reference that the Coordination service wants the registered participant to use for the Coordination protocol.

/RegisterResponse/{any}

Extensibility elements may be used to convey additional information.

/RegisterResponse /@{any}

Extensibility attributes may be used to convey additional information.

The following is an example of a RegisterResponse message:

```
<RegisterResponse>
  <CoordinatorProtocolService>
    <wsa:Address>
       http://Business456.com/mycoordinationservice/coordinator
    </wsa:Address>
    <wsa:ReferenceProperties>
      <myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>
    </wsa:ReferenceProperties>
  </CoordinatorProtocolService>
</RegisterResponse>
```

.

4 Coordination Faults

WS-Coordination faults MUST include as the [action] property the following fault action URI:

```
http://docs.oasis-open.org/ws-tx/wscoor/2006/03/fault
```

The faults defined in this section are generated if the condition stated in the preamble is met. Faults are targeted at a destination endpoint according to the fault handling rules defined in [WSADDR].

The definitions of faults in this section use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

For SOAP 1.2, the [Code] property MUST be either "Sender" or "Receiver". These properties are serialized into text XML as follows:

SOAP Version	Sender	Receiver
SOAP 1.2	S:Sender	S:Receiver

The properties above bind to a SOAP 1.2 fault as follows:

```
<S:Envelope>
 <S:Header>
   <wsa:Action>
      http://docs.oasis-open.org/ws-tx/wscoor/2006/03/fault
   </wsa:Action>
   <!-- Headers elided for clarity.  -->
 </S:Header>
 <S:Body>
  <S:Fault>
   <S:Code>
     <S:Value>[Code]</S:Value>
     <S:Subcode>
      <S:Value>[Subcode]</S:Value>
     </S:Subcode>
   </S:Code>
   <S:Reason>
     <S:Text xml:lang="en">[Reason]</S:Text>
   </S:Reason>
   <S:Detail>
     [Detail]
     ...
   </S:Detail>
  </S:Fault>
 </S:Body>
</S:Envelope>
```

The properties bind to a SOAP 1.1 fault as follows:

```
<S11:Envelope>
 <S11:Body>
  <S11:Fault>
   <faultcode>[Subcode]</faultcode>
   <faultstring xml:lang="en">[Reason]</faultstring>
  </S11:Fault>
 </S11:Body>
</S11:Envelope>
```

4.1 Invalid State

This fault is sent by either the coordinator or a participant to indicate that the endpoint that generates the fault has received a message that is not valid for its current state. This is an unrecoverable condition.

Properties:

[Code] Sender

[Subcode] wscoor:InvalidState

[Reason] The message was invalid for the current state of the activity.

[Detail] unspecified

4.2 Invalid Protocol

This fault is sent by either the coordinator or a participant to indicate that the endpoint that generates the fault received a message from an invalid protocol. This is an unrecoverable condition.

Properties:

[Code] Sender

[Subcode] wscoor:InvalidProtocol

[Reason] The protocol is invalid or is not supported by the coordinator.

4.3 Invalid Parameters

This fault is sent by either the coordinator or a participant to indicate that the endpoint that generated the fault received invalid parameters on or within a message. This is an unrecoverable condition.

Properties:

[Code] Sender

[Subcode] wscoor:InvalidParameters

[Reason] The message contained invalid parameters and could not be processed.

4.4 Cannot Create Context

This fault is sent by the Activation Service to the sender of a CreateCoordinationContext to indicate that a context could not be created.

Properties:

[Code] Sender

[Subcode] wscoor:CannotCreateContext

[Reason] CoordinationContext could not be created.

[Detail] unspecified

4.5 Cannot Register Participant

This fault is sent by the Registration Service to the sender of a Register to indicate that the Participant could not be registered.

Properties:

[Code] Sender

[Subcode] wscoor:CannotRegisterParticipant

[Reason] Participant could not be registered.

[Detail] unspecified

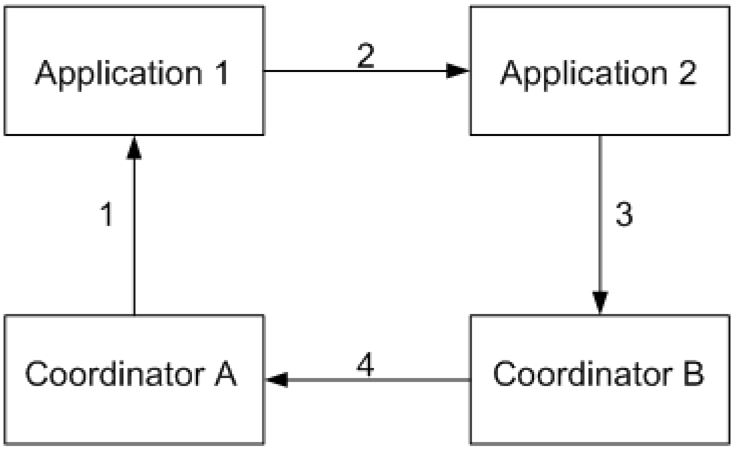
5 Security Model

The primary goals of security with respect to WS-Coordination are to:

1. ensure only authorized principals can create coordination contexts
2. ensure only authorized principals can register with an activity
3. ensure only legitimate coordination contexts are used to register
4. enable existing security infrastructures to be leveraged
5. allow principal authorization to be based on federated identities

These goals build on the general security requirements for integrity, confidentiality, and authentication, each of which is provided by the foundations built using the Web service security specifications such as WS-Security [WSSec] and WS-Trust [WSTrust].

The following figure illustrates a fairly common usage scenario:



In the figure above, step 1 involves the creation and subsequent communication between the creator of the context and the coordinator A (root). It should be noted that this may be a private or local communication. Step 2 involves the delegation of the right to register with the activity using the information from the coordination context and subsequent application messages between two applications (and may include middleware involvement) which are participants in the activity. Step 3 involves delegation of the right to register with the activity to coordinator B (subordinate) that manages all access to the activity on behalf of the second, and possibly other parties. Again note that this may also be a private or local communication. Step 4 involves registration with the coordinator A by the coordinator B and proof that registration rights were delegated.

It should be noted that many different coordination topologies may exist which may leverage different security technologies, infrastructures, and token formats. Consequently an appropriate security model must allow for different topologies, usage scenarios, delegation requirements, and security configurations.

To achieve these goals, the security model for WS-Coordination leverages the infrastructure provided by WS-Security [WSSec], WS-Trust [WSTrust], WS-Policy [WSPOLICY], and WS-SecureConversation [WSSecConv]: Services have policies specifying their requirements and requestors provide claims (either implicit or explicit) and the requisite proof of those claims.

There are a number of different mechanisms which can be used to affect the previously identified goals. However, this specification RECOMMENDS a simple mechanism, which is described here, for use in interoperability scenarios.

5.1 CoordinationContext Creation

When a coordination context is created (step 1 above) the message is secured using the mechanisms described in WS-Security. If the required claims are proven, as described by WS-Policy [WSPOLICY], then the coordination context is created.

A set of claims, bound to the identity of the coordination context's creator, and maintained by the coordinator, are associated with the creation of the coordination context. The creator of the context must obtain these claims from the coordinator. Before responding with the claims, the coordinator requires proof of the requestor's identity.

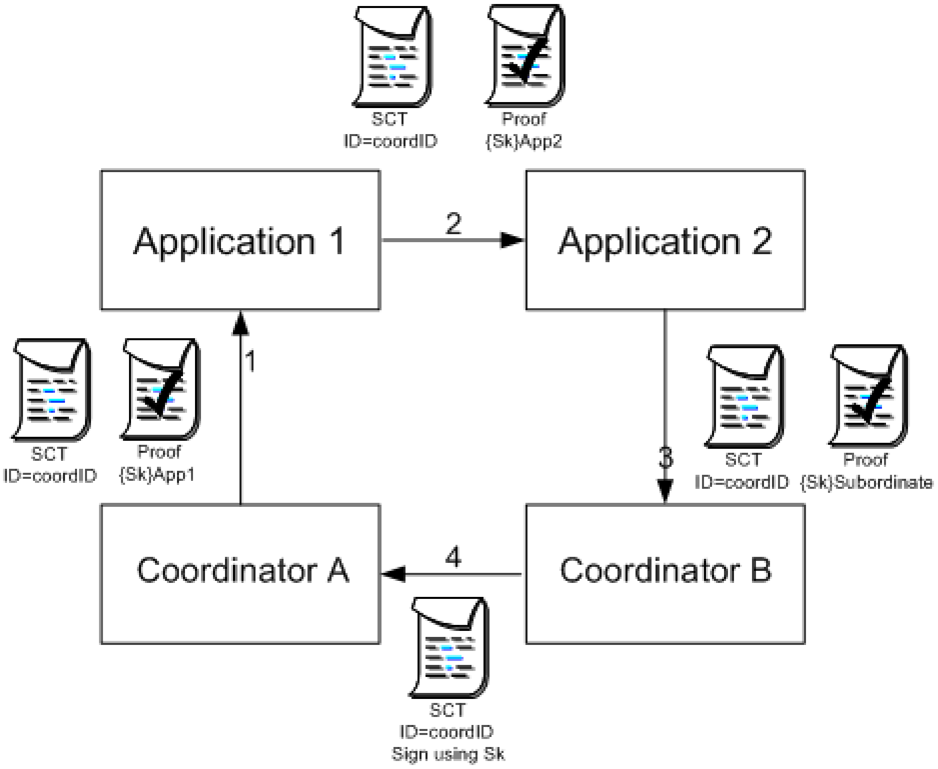
Additionally, the coordinator provides a shared secret which is used to indicate authorization to register with the coordination context by other parties. The secret is communicated using a security token and a <wst:RequestSecurityTokenResponse> element inside a <wst:IssuedTokens> header. The security token and hence the secret is scoped to a particular coordination context using the textual value of a <wscoor:Identifier> element in a <wsp:AppliesTo> element in the <wst:RequestSecurityTokenResponse> using the mechanisms described in WS-Trust [WSTrust]. This secret may be delegated to other parties as described in the next section.

5.2 Registration Rights Delegation

Secret delegation is performed by propagation of the security token that was created by the root Coordinator. This involves using the <wst:IssuedTokens> header containing a <wst:RequestSecurityTokenResponse> element. The entire header SHOULD be encrypted for the new participant.

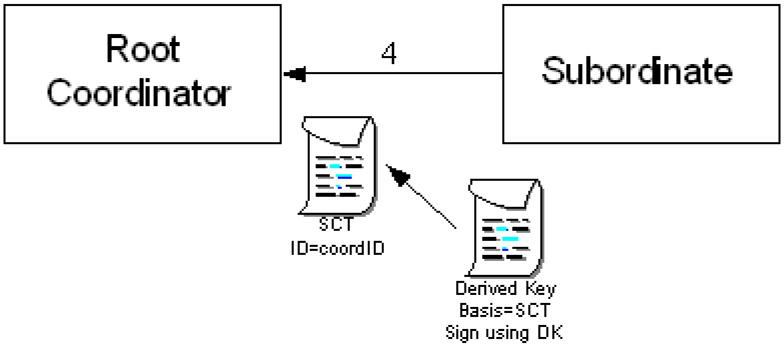
The participants can then use the shared secret using WS-Security by providing a signature based on the key/secret to authenticate and authorize the right to register with the activity that created the coordination context.

The figure below illustrates this simple key delegation model:



As illustrated in the figure above, the coordinator A, root in this case, (or its delegate) creates a security context token (cordID) representing the right to register and returns (using the mechanisms defined in WS-Trust [WSTrust]) that token to Application 1 (or its delegate) (defined in WS-SecureConversation [WSSecConv]) and a session key (Sk) encrypted for Application 1 inside of a proof token. This key allows Application 1 (or its delegate) to prove it is authorized to use the SCT. Application 1 (or its delegate) decrypts the session key (Sk) and encrypts it for Application 2 its delgate. Application 2 (or its delegate) performs the same act encrypting the key for the subordinate. Finally, coordinator B, subordinate in this case, proves its right to the SCT by including a signature using Sk.

It is RECOMMENDED by this specification that the key/secret never actually be used to secure a message. Instead, keys derived from this secret SHOULD be used to secure a message, as described in WS-SecureConversation [WSSecConv]. This technique is used to maximize the strength of the key/secret as illustrated in the figure below:



6 Security Considerations

It is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [WSSec]. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the <wscoor:CoordinationContext> header needs to be signed with the body and other key message headers in order to "bind" the two together. This will ensure that the coordination context is not tampered. In addition the reference properties within an Endpoint Reference may be encrypted to ensure their privacy.

In the event that a participant communicates frequently with a coordinator, it is RECOMMENDED that a security context be established using the mechanisms described in WS-Trust [WSTrust] and WS-SecureConversation [WSSecConv] allowing for potentially more efficient means of authentication.

It is common for communication with coordinators to exchange multiple messages. As a result, the usage profile is such that it is susceptible to key attacks. For this reason it is strongly RECOMMENDED that the keys used to secure the channel be changed frequently. This "re-keying" can be effected a number of ways. The following list outlines four common techniques:

- Attaching a nonce to each message and using it in a derived key function with the shared secret
- Using a derived key sequence and switch "generations"
- Closing and re-establishing a security context
- Exchanging new secrets between the parties

It should be noted that the mechanisms listed above are independent of the SCT and secret returned when the coordination context is created. That is, the keys used to secure the channel may be independent of the key used to prove the right to register with the coordination context.

The security context MAY be re-established using the mechanisms described in WS-Trust [WSTrust] and WS-SecureConversation [WSSecConv]. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret SHOULD NOT be used to encrypt the new shared secret. Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation.

The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

- **Message alteration** – Alteration is prevented by including signatures of the message information using WS-Security [WSSec].
- **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security.
- **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy [WSPOLICY] and WS-SecurityPolicy [WSSecPolicy]).
- **Authentication** – Authentication is established using the mechanisms described in WS-Security [WSSec] and WS-Trust [WSTrust]. Each message is authenticated using the mechanisms described in WS-Security.
- **Accountability** – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- **Availability** – Many services are subject to a variety of availability attacks. Replay is a common attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification. That said, care should be taken to ensure that minimal processing be performed prior to any authenticating sequences.

- **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security [WSSec]. Alternatively, and optionally, other technologies, such as sequencing, can also be used to prevent replay of application messages.

7 Use of WS-Addressing Headers

The messages defined in WS-Coordination can be classified into two types:

- Request messages: **CreateCoordinationContext** and **Register**.
- Reply messages: **CreateCoordinationContextResponse** and **RegisterResponse** and faults

Request and reply messages follow the standard "Request Reply" pattern as defined in WS-Addressing.

The following statements define addressing interoperability requirements for the respective WS-Coordination message types:

Request messages

- MUST include a wsa:MessageID header.
- MUST include a wsa:ReplyTo header.

Reply messages

- MUST include a wsa:RelatesTo header, specifying the MessageID from the corresponding Request message.
- MUST include the reference parameter elements from the request's wsa:ReplyTo header in their header blocks.

All messages are delivered using connections initiated by the sender. Endpoint References MUST contain physical addresses and MUST NOT use the well-known "anonymous" endpoint defined in WS-Addressing.

8 Glossary

The following definitions are used throughout this specification:

Activation service: This supports a CreateCoordinationContext operation that is used by participants to create a CoordinationContext.

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.

Coordination protocol: The definition of the coordination behavior and the messages exchanged between the coordinator and a participant playing a specific role within a coordination type. WSDL definitions are provided, along with sequencing rules for the messages. The definition of coordination protocols are provided in additional specification (e.g., WS-AtomicTransaction).

Coordination type: A defined set of coordination behaviors, including how the service accepts context creations and coordination protocol registrations, and drives the coordination protocols associated with the activity.

Coordination service (or Coordinator): This service consists of an activation service, a registration service, and a set of coordination protocol services.

Participant: A service that is carrying out a computation within the activity. A participant receives the CoordinationContext and can use it to register for coordination protocols.

Registration service: This supports a Register operation that is used by participants to register for any of the coordination protocols supported by a coordination type, such as Atomic Transaction 2PC or Business Agreement NestedScope.

Web service: A Web service is a computational service, accessible via messages of definite, programming-language-neutral and platform-neutral format, and which has no special presumption that the results of the computation are used primarily for display by a user-agent.

Appendix A. Acknowledgements

This document is based on initial contribution to OASIS WS-TX Technical Committee by the following authors: Luis Felipe Cabrera, Microsoft, George Copeland, Microsoft, Max Feingold, Microsoft,(Editor), Robert W Freund, Hitachi, Tom Freund, IBM, Jim Johnson, Microsoft, Sean Joyce, IONA, Chris Kaler, Microsoft, Johannes Klein, Microsoft, David Langworthy, Microsoft, Mark Little, Arjuna Technologies, Anthony Nadalin, IBM, Eric Newcomer, IONA, David Orchard, BEA Systems, Ian Robinson, IBM, John Shewchuk, Microsoft, Tony Storey, IBM.

The following individuals have provided invaluable input into the initial contribution: Francisco Curbera, IBM, Sanjay Dalal, BEA Systems, Doug Davis, IBM, Don Ferguson, IBM, Kirill Gavrylyuk, Microsoft, Dan House, IBM, Oisin Hurley, IONA, Frank Leymann, IBM, Thomas Mikalsen, IBM, Jagan Peri, Microsoft, Alex Somogyi, BEA Systems, Stefan Tai, IBM, Satish Thatte, Microsoft, Gary Tully, IONA, Sanjiva Weerawarana, IBM.

The following individuals were members of the committee during the development of this specification:

TBD

[Participant Name, Affiliation | Individual Member]
[Participant Name, Affiliation | Individual Member]

Appendix B. Revision History

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
01	2005-11-22	Max Feingold	Initial Working Draft
02	2006-02-20	Max Feingold	References have been made non-normative. Refer to Section Non-normative References. [TC Issue i017] Change copyright year to 2006 both in the copyright notice and the footer.
03	2006-03-06	Max Feingold	Added new fault CannotCreateContext, CannotRegisterParticipant. [TC Issues i004, i005] Modified document Identifier, location, and footer to reflect the working draft version 03. Also modified the status description. Removed faults NoActivity, AlreadyRegistered, ContextRefused. [TC Issues i006, i008, i013] Added additional description to section “Registration Service”. [Issue i007] Updated description in Section “Coordination Context”. [Issue i012] Updated description in Section “Coordination Service”. [Issues i018, i019, i020, i021] Changed namespace and action URIs. [Issue i015]
04	2006-03-10	Max Feingold	Added new Section “Use of WS-Addressing Headers”. [Issue i009] Updated text in Section “Coordination Context”. [Issue i022] Updated Section “Non-normative References”. [Issue i024]

Appendix C. Non-normative Text