



[IBM home](#) | [Products & services](#) | [Support & downloads](#) | [My account](#)

[IBM developerWorks](#) : [Web services](#) : [Web services articles](#)

developerWorks

Web Services Coordination (WS-Coordination)

9 August 2002

This version:

<http://www.ibm.com/developerworks/library/ws-coor/> (available in [PDF](#))

Authors:

Felipe Cabrera, Microsoft [<cabrera@microsoft.com>](mailto:cabrera@microsoft.com)
George Copeland, Microsoft [<gcope@microsoft.com>](mailto:gcope@microsoft.com)
Tom Freund, IBM [<tjfreund@uk.ibm.com>](mailto:tjfreund@uk.ibm.com)
Johannes Klein, Microsoft [<joklein@microsoft.com>](mailto:joklein@microsoft.com)
David Langworthy, Microsoft [<dlan@microsoft.com>](mailto:dlan@microsoft.com)
David Orchard, BEA Systems [<dorchard@bea.com>](mailto:dorchard@bea.com)
John Shewchuk, Microsoft [<johnshew@microsoft.com>](mailto:johnshew@microsoft.com)
Tony Storey, IBM [<tony_storey@uk.ibm.com>](mailto:tony_storey@uk.ibm.com)

Copyright© 2002 [BEA Systems](#), [International Business Machines Corporation](#), [Microsoft Corporation](#). All rights reserved. The presentation, distribution or other dissemination of the information contained in this specification is not a license, either expressly or impliedly, to any intellectual property owned or controlled by BEA or IBM or Microsoft and/or any other third party. BEA, IBM, Microsoft, and/or any other third party may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to BEA's or IBM's or Microsoft's or any other third party's patents, trademarks, copyrights, or other intellectual property. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred. This specification and the information contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, BEA, IBM and Microsoft provides the document AS IS AND WITH ALL FAULTS, and hereby disclaims all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the document. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE DOCUMENT. IN NO EVENT WILL BEA OR IBM OR MICROSOFT BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS DOCUMENT, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

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 transactions.

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.

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.

Status of this Document

WS-Coordination and related specifications are provided for use as-is and for review and evaluation only. Microsoft, BEA and IBM will solicit your contributions and suggestions in the near future. Microsoft, BEA and IBM make no warranties or representations regarding the specification in any manner whatsoever.

Acknowledgements

The following individuals have provided invaluable input into the design of the WS-Transaction specification:

Francisco Curbera, IBM

Don Ferguson, IBM

Frank Leymann, IBM

Jagan Peri, Microsoft

Satish Thatte, Microsoft

Sanjiva Weerawarana, IBM

We also wish to thank the technical writers and development reviewers who provided feedback to improve the readability of the specification.

Table of Contents

1. [Introduction](#)
 - 1.1. [The Model](#)
 - 1.2. [Extensibility](#)
 - 1.3. [Notational Conventions](#)
 - 1.4. [Namespace](#)
 - 1.5. [XSD and WSDL Files](#)
2. [CoordinationContext](#)
3. [Coordination Service](#)
 - 3.1. [Activation Service](#)
 - 3.1.1. [CreateCoordinationContext Message](#)
 - 3.1.2. [CreateCoordinationContextResponse Message](#)
 - 3.2. [Registration Service](#)
 - 3.2.1. [Register Message](#)
 - 3.2.2. [RegisterResponse Message](#)
 - 3.3. [Error Message](#)
4. [Security Considerations](#)
5. [Relationship to Other Web Services](#)

- 6. [Interoperability Considerations](#)
 - 7. [Glossary](#)
 - 8. [Appendices](#)
 - 8.1. [Appendix A. Port Reference](#)
 - 8.1.1. [XML Representation](#)
 - 8.1.2. [XML Example](#)
 - 8.1.3. [Security Considerations](#)
 - 8.2. [Appendix B. Context](#)
 - 8.2.1. [B.1 XML Representation](#)
 - 8.2.2. [XML Example](#)
 - 8.2.3. [Security Considerations](#)
 - 9. [References](#)
-

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.

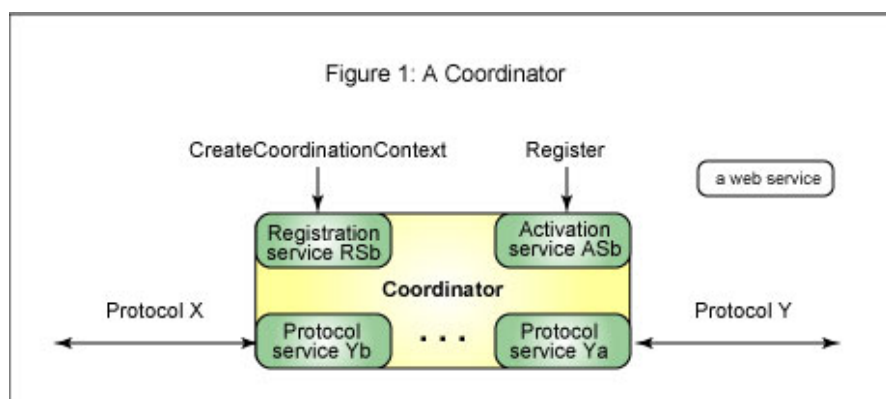
1.1. The Model

This specification describes a framework for a coordination service (or coordinator) which consists of the following 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 means are appropriate 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. 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 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.

1.3. Notational Conventions

The keywords "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 [\[KEYWORDS\]](#).

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 `<-- description -->` 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 [\[XML-Schema1\]\[WSDL\]](#)

1.4. Namespace

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

```
http://schemas.xmlsoap.org/ws/2002/08/wscoor
```

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/2001/12/soap-envelope
wsu	http://schemas.xmlsoap.org/ws/2002/07/utility
wscoor	http://schemas.xmlsoap.org/ws/2002/08/wscoor

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

```
http://schemas.xmlsoap.org/ws/2002/08/wscoor#Register
```

1.5. XSD and WSDL Files

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

<http://schemas.xmlsoap.org/ws/2002/08/wscoor/wscoor.xsd>

<http://schemas.xmlsoap.org/ws/2002/08/wscoor/wscoor.wsdl>

2. CoordinationContext

The `CoordinationContext` is a `Context` type, as described in [Appendix B](#), that is used to pass coordination information to parties involved in a coordination service.

`CoordinationContext` elements are placed within messages that are used by the parties. A `CoordinationContext` provides access to a coordination registration service, a coordination type, and relevant extensions.

[Listing 1](#) contains an example of a `CoordinationContext` supporting a transaction service.

Listing 1. Example CoordinationContext

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
  <S:Header>
    . . .
  <wscoor:CoordinationContext
    xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
    xmlns:wscoor="http://schemas.xmlsoap.org/ws/2002/08/wscoor"
    xmlns:myApp="http://fabrikam123.com/myApp">
  <wsu:Expires>
    2002-06-30T13:20:00.000-05:00
  </wsu:Expires>
  <wsu:Identifier>
    http://Fabrikam123.com/SS/1234
  </wsu:Identifier>
```

```

<wscoor:CoordinationType>
  http://schemas.xmlsoap.org/ws/2002/08/wstx
</wscoor:CoordinationType>
<wscoor:RegistrationService>
  <wsu:Address>
    http://Business456.com/mycoordination-service/registration
  </wsu:Address>
  <myApp:BetaMark> ... </myApp:BetaMark>
  <myApp:EBDCode> ... </myApp:EBDCode>
</wscoor:RegistrationService>
<myApp:IsolationLevel>
  RepeatableRead
</myApp:IsolationLevel>
</wscoor:CoordinationContext>
. . .
</S:Header>

```

When an application propagates an activity using a Coordination service, applications MUST include a `CoordinationContext` in the outgoing message.

3. Coordination Service

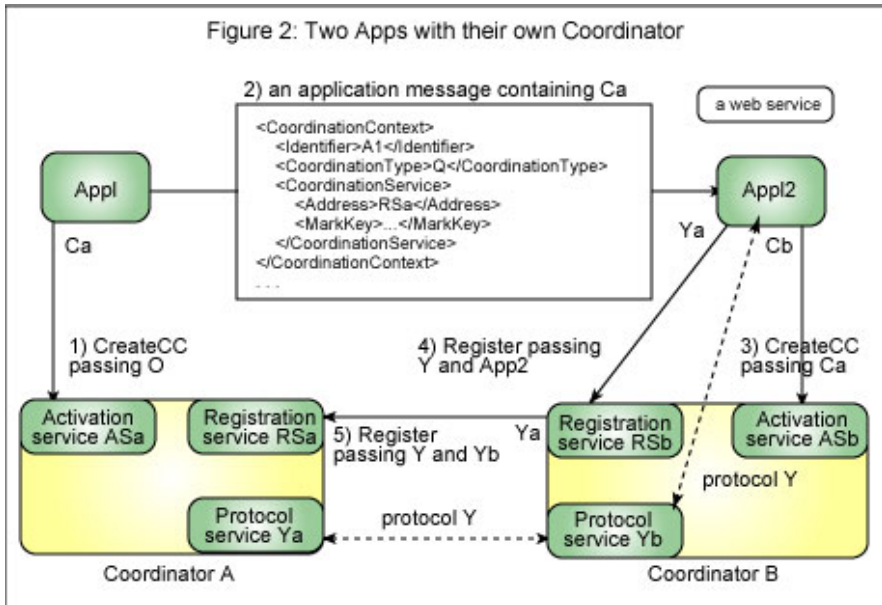
The Coordination service (or coordinator) is an aggregation of the following services:

- 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 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 a `PortReference` to CoordinatorA's Registration service RSa.
2. App1 then sends an application message to App2 containing the `Context Ca`.
3. App2 prefers CoordinatorB, 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 `PortReferences` for App2 and the protocol service Yb. This forms a logical connection between these `PortReferences` that the protocol Y can use.
5. This registration causes CoordinatorB to forward the registration onto CoordinatorA's Registration service RSa, exchanging `PortReferences` for Yb and the protocol service Ya. This forms a logical connection between these `PortReferences` that the protocol Y can use.

Figure 2. Two apps with their own coordinators



3.1. Activation Service

The Activation service definition requires a port type on the coordinator side (for the request) and on the requester side (for the response).

The coordinator's Activation service is defined in [Listing 2](#).

Listing 2. Coordinator's Activation service

```

<wsdl:portType name="ActivationCoordinatorPortType">
  <wsdl:operation name="CreateCoordinationContext">
    <wsdl:input message="wscoor:CreateCoordinationContext"/>
  </wsdl:operation>
</wsdl:portType>

```

The requester's Activation service is defined in [Listing 3](#).

Listing 3. Requester's Activation service

```

<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>

```

3.1.1. CreateCoordinationContext Message

This request is used to create a coordination context that supports a *coordination type* -- 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-Transaction).

The CreateCoordinationContext message is defined in [Listing 4](#).

Listing 4. CreateCoordinationContext message

```
<CreateCoordinationContext ...>
  <ActivationService> ... </ActivationService>
  <RequesterReference> ... </RequesterReference>
  <CoordinationType> ... </CoordinationType>
  <wsu:Expires> ... </wsu:Expires>?
  <CurrentContext> ... </CurrentContext>?
  <!-- extensibility element -- >*
</CreateCoordinationContext>
```

/CreateCoordinationContext/ActivationService

This provides the Activation service port reference.

/CreateCoordinationContext/RequesterReference

This provides the caller port reference. As described below, this enables the response to be sent when using one-way messaging.

/CreateCoordinationContext/CoordinationType

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

/CreateCoordinationContext/wsu:Expires

Optional. The date/time for the returned CoordinationContext.

/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.

The predefined error codes for the CreateCoordinationContext request are:

- `wscoor:InvalidCreateParameters` -- The parameters passed to the CreateCoordinationContext were invalid.

[Listing 5](#) is an example create message.

Listing 5. Example create message

```
<CreateCoordinationContext>
  <ActivationService>
    <wsu:Address>
      http://Business456.com/tm/activation
    </wsu:Address>
    <myapp:MyPrivateState>
      1234
    </myapp:MyPrivateState>
  </ActivationService>
  <RequesterReference>
    <wsu:Address>
      http://fabrikaml23.com/app1
```

```

    </wsu:Address>
  </RequesterReference>
  <CoordinationType>
    http://schemas.xmlsoap.org/ws/2002/08/wstx
  </CoordinationType>
</CreateCoordinationContext>

```

3.1.2. CreateCoordinationContextResponse Message

This returns the CoordinationContext that was created.

The CreateCoordinationContextResponse message is defined in [Listing 6](#).

Listing 6. CreateCoordinationContextResponse

```

<CreateCoordinationContextResponse ...>
  <RequesterReference> ... </RequesterReference>
  <CoordinationContext> ... </CoordinationContext>
  <!-- extensibility element -- >*
</CreateCoordinationContextResponse>

```

/CreateCoordinationContext/RequesterReference

This provides the port reference of the caller that invoked CreateCoordinationContext.

/CreateCoordinationContext/{any}

Extensibility elements may be used to convey additional information.

Extensibility attributes may be used to convey additional information.

[Listing 7](#) is an example create response message that does not contain additional information.

Listing 7. Example create response message

```

<CreateCoordinationContextResponse>
  <RequesterReference>
    <wsu:Address>
      http://fabrikaml23.com/appl
    </wsu:Address>
  </RequesterReference>
  <CoordinationContext>
    <wsu:Identifier>
      http://Business456.com/tm/context1234
    </wsu:Identifier>
    <CoordinationType>
      http://schemas.xmlsoap.org/ws/2002/08/wstx
    </CoordinationType>
    <RegistrationService>
      <wsu:Address>
        http://Business456.com/tm/registration
      <wsu:Address>
        <myapp:PrivateInstance>
          1234
        </myapp:PrivateInstance>
      </wsu:Address>
    </RegistrationService>
  </CoordinationContext>
</CreateCoordinationContextResponse>

```

```

        </myapp:PrivateInstance>
    </RegistrationService>
</CoordinationContext>
</CreateCoordinationContextResponse>

```

3.2. Registration Service

The Registration service definition requires a port type on the coordinator side (for the request) and on the requester side (for the response).

The coordinator's Registration service is defined in [Listing 8](#).

Listing 8. Coordinator's Registration service

```

<wsdl:portType name="RegistrationCoordinatorPortType">
  <wsdl:operation name="Register">
    <wsdl:input message="wscor:Register" />
  </wsdl:operation>
</wsdl:portType>

```

The requester's Registration service is defined in [Listing 9](#).

Listing 9. Requester's Registration service

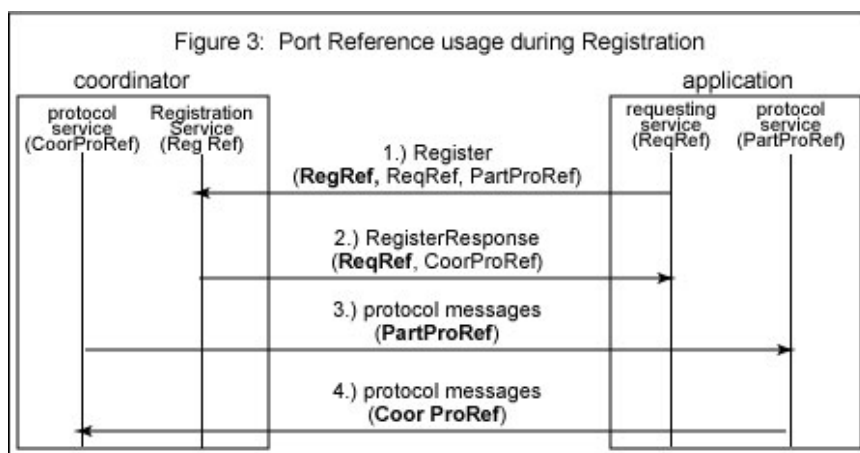
```

<wsdl:portType name="RegistrationRequesterPortType">
  <wsdl:operation name="RegisterResponse">
    <wsdl:input message="wscor:RegisterResponse" />
  </wsdl:operation>
  <wsdl:operation name="Error">
    <wsdl:input message="wscor:Error" />
  </wsdl:operation>
</wsdl:portType>

```

[Figure 3](#) illustrates the usage of `PortReferences` during registration. The target `PortReference` is indicated in boldface type. The coordinator provides the Registration `PortReference` in the `CoordinationContext` during the `CreateCoordinationContext` operation. The requesting service receives the Registration service `PortReference` (`RegRef`) in the `CoordinationContext` in an application message.

Figure 3. PortReference usage during Registration



- 1. The Register message targets this `PortReference`, also including the requesting service `PortReference` (`ReqRef`) and the participant protocol service `PortReference` as parameters.

- 2. The `RegisterResponse` message targets the requesting service (`ReqRef`), also including the coordinator's protocol service `PortReference`.
- 3 and 4. At this point, each side has the `PortReference` of the other's protocol service, so each protocol messages can target the other side.

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 port references. Each side of the coordination protocol (participant and coordinator) supplies a port 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 **Register** operation is defined in [Listing 10](#).

Listing 10. Register operation definition

```
< Register ...>
  <RegistrationService> ... </RegistrationService>
  <RequesterReference> ... </RequesterReference>
  <ProtocolIdentifier> ... </ProtocolIdentifier>
  <ParticipantProtocolService> ... </ParticipantProtocolService>
  <!-- extensibility element -- >*
</Register>
```

`/Register/RegistrationService`

This provides the registration port reference.

`/Register/RequesterReference`

The port reference to which the application wants the registration service to return status information.

`/Register/ProtocolIdentifier`

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

`/Register/ParticipantProtocolService`

The port reference that the registering participant wants the coordinator to use for the `Coordination` protocol.

`/Register/{any}`

Extensibility elements may be used to convey additional information.

`/Register/@{any}`

Extensibility attributes may be used to convey additional information.

The predefined error codes for the `Register` request are:

- `wscor:AlreadyRegistered` -- The participant has already registered for this coordination protocol under this activity identifier.
- `wscor:InvalidState` -- The state of the coordinator no longer allows registration for this coordination protocol.
- `wscor:InvalidProtocol` -- The coordination protocol is not supported.
- `wscor:NoActivity` -- The activity does not exist.

[Listing 11](#) is an example registration message:

Listing 11. Example registration message

```
<Register>
  <RegistrationService>
    <wsu:Address>
      http://Business456.com/mycoordinationservice/registration
    </wsu:Address>
    <EBCXCode> H23974922Z </EBCXCode>
  </RegistrationService>
  <RequesterReference>
    <wsu:Address>http://Schedule456.com</wsu:Address>
  </RequesterReference>
  <ProtocolIdentifier>
    http://schemas.xmlsoap.org/ws/2002/08/wstx/2PC
  </ProtocolIdentifier>
  <ParticipantProtocolService>
    <wsu:Address>
      http://Adventure456.com/participant2PCservice
    </wsu:Address>
    <BetaMark> AlphaBetaGamma </BetaMark>
  </ParticipantProtocolService>
</Register>
```

3.2.2. RegisterResponse Message

Receipt of the **RegisterResponse** message indicates that registration has completed successfully. The **RegisterResponse** message is defined in [Listing 12](#).

Listing 12. RegisterResponse message

```
<RegisterResponse>
  <RequesterReference> ... </RequesterReference>
  <CoordinatorProtocolService> ... </CoordinatorProtocolService>
  <!-- extensibility element -- >*
</RegisterResponse>
```

/Register/RequesterReference

The port reference to which the application wants the Registration service to return status information. This should include enough information to correlate a request with a response.

/RegisterResponse/CoordinatorProtocolService

The port 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 attribute may be used to convey additional information.

[Listing 13](#) is an example of a RegisterResponse message where the MarkKey element is additional address information private to an implementation used to correlate the request.

Listing 13. Example RegisterResponse message

```

<RegisterResponse>
  <RequesterReference>
    <wsu:Address>
      http://Schedule456.com
    </wsu:Address>
  </RequesterReference>
  <CoordinatorProtocolService>
    <wsu:Address>
      http://Business456.com/mycoordination-service/coordinator
    </wsu:Address>
    <myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>
  </CoordinatorProtocolService>
</RegisterResponse>

```

3.3. Error Message

Receipt of the **Error** message indicates that creation or registration request has failed.

The **Error** message is defined in [Listing 14](#).

Listing 14. Error message definition

```

<Error ...>
  <CoordinatorProtocolService> ... </CoordinatorControlService?>
  <Errorcode> ... </Errorcode>
  <!-- extensibility element -- >*
</Error>

```

/Error/CoordinatorProtocolService

The port reference that the Coordination service wants the registered participant to use for the Coordination protocol.

/Error/Errorcode

This required element is a QName that identifies the reason for the error.

/Error/{any}

Extensibility elements may be used to convey additional information.

/Error/@{any}

Extensibility attributes may be used to convey additional information.

4. Security Considerations

Because messages can be modified or forged, it is strongly RECOMMENDED that business process implementations use WS-Security to ensure messages have not been modified or forged while in transit or while residing at destinations. Similarly, invalid or expired messages could be re-used or message headers not specifically associated with the specific message could be referenced. Consequently, when using WS-Security, signatures MUST include the semantically significant headers and the message body (as well as any other relevant data) so that they cannot be independently separated and re-used.

In order to trust contexts, they SHOULD be signed using the mechanisms outlined in WS-Security. This allows readers of the context information to be certain that the contexts haven't been forged or altered in any way. It is strongly RECOMMENDED that contexts be signed. In addition to the mechanisms list above, contexts SHOULD include a message timestamp (as described in WS-Security).

It should also be noted that services implementing this are subject to various forms of denial-of-service attacks. Implementers should take this into account when building their services.

5. Relationship to Other Web Services

WS-Coordination depends on the <Expires> element defined in WS-Security and on 'Context' and 'PortReference' definitions contained in the appendices of this document.

6. Interoperability Considerations

In order for two parties to communicate, both parties will need to agree on the protocols provided. This specification facilitates this agreement and thus facilitates interoperability.

7. 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 port 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 an additional specification (e.g., WS-Transaction).

Coordination type

A defined set of coordination behaviors, including how the service accepts context creations and coordination protocol registrations; 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; it has no special presumption that the results of the computation are used primarily for display by a user agent.

8. Appendices

8.1. Appendix A. Port Reference

Editor's note: The TX working group provides the following minimal definition of port reference for purposes of enabling interoperability. Specifically, the following is what we need in order to enable the WS-Transaction specification. Our expectation is that a more complete definition will be provided and this specification will be updated to support it.

A port reference conveys the capability to invoke operations against an individual port. Instance-specific information may be added to reach a specific service instance in the target service, to scope a service interaction within a session, or to carry the values of context properties required for the invocation of the target.

A port reference contains at least the URI address of the port address.

A port reference can be extended with optional elements.

8.1.1. XML Representation

[Listing 16](#) is a schema outline for <PortReference>:

Listing 16. Port reference schema

```
<wsu:PortReference ...>
  <wsu:Address> ... </wsu:Address>
  <!-- extensibility element -- >*
</wsu:PortReference>
```

/PortReference/Address

Required. An absolute URI of a web services port.

/PortReference/{any}

This is an extensibility mechanism to allow additional elements to be added to the element. An implementation MAY ignore any element unless the element has a top level (child of <PortReference>)

wsu:MustUnderstand attribute with a value of *true*.

/PortReference/@{any}

This is an extensibility mechanism to allow additional attributes to be added to the element.

8.1.2. XML Example

[Listing 17](#) is a simple example. The reference describes the port at address `http://fabrikam123.com/accounting/auditPort` with instance specific information of 1234.

Listing 17. Port reference example

```
<wsu:PortReference xmlns:acct="http://Adventure456.com/auditing">
  <wsu:Address>
    http://fabrikam123.com/asvc/auditPort
  </wsu:Address>
  <acct:MyPrivateInfo>
    1234
  </acct:MyPrivateInfo>
</wsu:PortReference>
```

8.1.3. Security Considerations

In order to *trust* references, they SHOULD be signed using the mechanisms outlined in [\[WSSec\]](#). This allows readers of the reference information to be certain that the references haven't been forged or altered in any way. It is strongly RECOMMENDED that references be signed.

8.2. Appendix B. Context

The context is a container for sharing processing contexts between web service endpoints. This is used to allow different endpoints to correlate contexts between messages.

We define the type *Context* which specifies an optional expiration and an identifier. The identifier is a URI that is used to identify groups of related messages. It is expected that elements derived from this type will impose semantics about group membership. It should be noted that some derivations MAY omit the optional expiration element.

We further define a general purpose header `<Context>` based on this type.

8.2.1. B.1 XML Representation

The `Context` type is defined in [Listing 18](#).

Listing 18. Context type definition

```
<wsu:Context ...>
  <wsu:Identifier ...> ... </wsu:Identifier>
  <wsu:Expires> ... </wsu:Expires>?
  <!-- extensibility element -- >*
</wsu:Context>
```

`/Context/Expires`

This optional element indicates the expiration date of the context.

`/Context/Identifier`

This required element specifies a unique identifier for the context. This is a URI identifying a group of related messages.

`/Context/{any}`

This is an optional extensibility mechanism to allow additional elements to be specified.

`/Context/@{any}`

This is an extensibility mechanism to allow additional attributes to be added to the element.

8.2.2. XML Example

[Listing 19](#) illustrates the use of a context.

Listing 19. Example context definition

```
<wsu:Context>
  <wsu:Identifier> http://fabrikaml23.com/SS/34194 </wsu:Identifier>
  <wsu:Expires> 2001-10-13T09:00:00Z </wsu:Expires>
  <myapp:MarkKey> 400 </myapp:MarkKey>
</wsu:Context>
```

8.2.3. Security Considerations

In order to *trust* references, they SHOULD be signed using the mechanisms outlined in [\[WSSec\]](#). This allows readers of the reference information to be certain that the references haven't been forged or altered in any way. It is strongly RECOMMENDED that references be signed.

9. References

KEYWORDS

S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University, 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.

WSSEC

Microsoft, IBM, & VeriSign, [Web Services Security \(WS-Security\)](#)

WSTX

Microsoft, BEA & IBM, [Web Services Transactions](#)

WSDL

W3C Note, [Web Services Description Language \(WSDL\) 1.1](#)

[About IBM](#) | [Privacy](#) | [Legal](#) | [Contact](#)