



Web Services Coordination (WS-Coordination) Version 1.1

Committee Specification

8 February 2007

Specification URIs:

This Version:

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-cs-03/wstx-wscoor-1.1-spec-cs-03.html>

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-cs-03.pdf>

Previous Version:

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-pr-01/wstx-wscoor-1.1-spec-pr-01.html>

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

Latest Approved Version:

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec/wstx-wscoor-1.1-spec.html>

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

Technical Committee:

OASIS WS-TX TC

Chair(s):

Eric Newcomer, Iona

Ian Robinson, IBM

Editor(s):

Max Feingold, Microsoft

Ram Jeyaraman, Microsoft

Declared XML Namespaces:

<http://docs.oasis-open.org/ws-tx/wscoor/2006/06>

Abstract:

The WS-Coordination specification 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 was last revised or approved by the WS-TX TC on the above date. The level of approval is also listed above. Check the "Latest Approved Version" location noted above for possible later revisions of this document.

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

Copyright © OASIS Open 2007. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

Table of Contents

1	Introduction.....	5
1.1	Model.....	5
1.2	Composable Architecture.....	6
1.3	Extensibility.....	6
1.4	Terminology.....	6
1.5	Namespace.....	7
1.5.1	Prefix Namespace.....	7
1.6	XSD and WSDL Files.....	7
1.7	Coordination Protocol Elements.....	7
1.8	Normative References.....	7
1.9	Non-normative References.....	8
2	Coordination Context.....	9
3	Coordination Service.....	10
3.1	Activation Service.....	11
3.1.1	CreateCoordinationContext.....	11
3.1.2	CreateCoordinationContextResponse.....	12
3.2	Registration Service.....	13
3.2.1	Register Message.....	14
3.2.2	RegistrationResponse Message.....	15
4	Coordination Faults.....	16
4.1	Invalid State.....	17
4.2	Invalid Protocol.....	17
4.3	Invalid Parameters.....	17
4.4	Cannot Create Context.....	17
4.5	Cannot Register Participant.....	17
5	Security Model.....	19
5.1	CoordinationContext Creation.....	20
5.2	Registration Rights Delegation.....	20
6	Security Considerations.....	22
7	Use of WS-Addressing Headers.....	24
8	Glossary.....	25
	Appendix A. Acknowledgements.....	26

1 Introduction

The current set of Web service specifications (SOAP [SOAP 1.1] [SOAP 1.2] and WSDL [WSDL]) 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. For example, WS-AtomicTransaction [WSAT] and WS-BusinessActivity [WSBA] specifications use and build upon this specification.

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

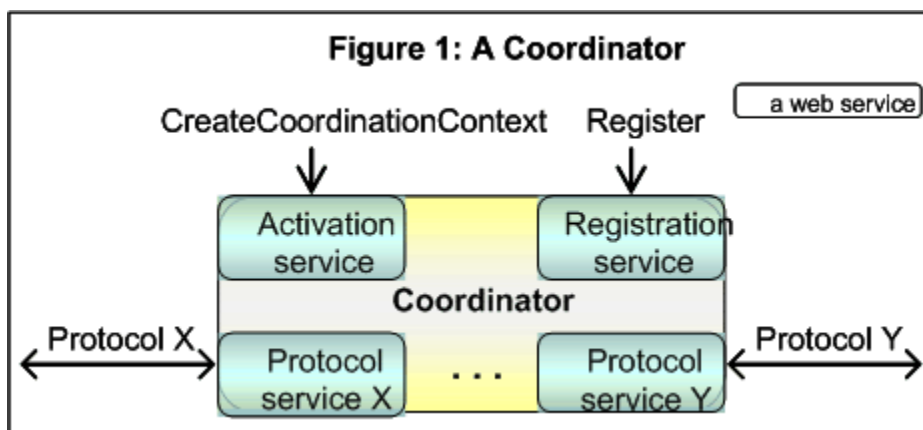
This specification describes a framework for a coordination service (or coordinator) which consists of these component services:

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.



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.

35 1.2 Composable Architecture

36 By using the XML [**XML**], SOAP [**SOAP 1.1**] [**SOAP 1.2**] and WSDL [**WSDL**] extensibility model, SOAP-
37 based and WSDL-based specifications are designed to be composed with each other to define a rich
38 Web services environment. As such, WS-Coordination by itself does not define all the features required
39 for a complete solution. WS-Coordination is a building block that is used in conjunction with other
40 specifications and application-specific protocols to accommodate a wide variety of protocols related to the
41 operation of distributed Web services.

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

45 1.3 Extensibility

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

- 47 • The publication of new coordination protocols.
- 48 • The selection of a protocol from a coordination type and the definition of extension elements that
49 can be added to protocols and message flows.

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

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

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

59 1.4 Terminology

60 The uppercase key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
61 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as
62 described in [**RFC2119**].

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

- 64 • The syntax appears as an XML instance, but the values indicate the data types instead of values.
- 65 • Element names ending in "..." (such as <element.../> or <element...>) indicate that
66 elements/attributes irrelevant to the context are being omitted.
- 67 • Attributed names ending in "..." (such as name=...) indicate that the values are specified below.
- 68 • Grammar in bold has not been introduced earlier in the document, or is of particular interest in an
69 example.
- 70 • <!-- description --> is a placeholder for elements from some "other" namespace (like ##other in
71 XSD).
- 72 • Characters are appended to elements, attributes, and <!-- descriptions --> as follows: "?" (0 or 1),
73 "*" (0 or more), "+" (1 or more). The characters "[" and "]" are used to indicate that contained
74 items are to be treated as a group with respect to the "?", "*", or "+" characters.
- 75 • The XML namespace prefixes (defined below) are used to indicate the namespace of the element
76 being defined.
- 77 • Examples starting with <?xml contain enough information to conform to this specification; others
78 examples are fragments and require additional information to be specified in order to conform.

79 1.5 Namespace

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

81 `http://docs.oasis-open.org/ws-tx/wscoor/2006/06`

82 1.5.1 Prefix Namespace

83 The following namespaces are used in this document:

Prefix	Namespace
S11	http://schemas.xmlsoap.org/soap/envelope
S12	http://www.w3.org/2003/05/soap-envelope
wscoor	http://docs.oasis-open.org/ws-tx/wscoor/2006/06
wsa	http://www.w3.org/2005/08/addressing

84 1.6 XSD and WSDL Files

85 Dereferencing the XML namespace defined in section 1.5 will produce the Resource Directory
86 Description Language (RDDL) **[RDDL]** document that describes this namespace, including the XML
87 schema **[XML-Schema1][XML-Schema2]** and WSDL **[WSDL]** declarations associated with this
88 specification.

89 SOAP bindings for the WSDL **[WSDL]**, referenced in the RDDL **[RDDL]** document, MUST use
90 "document" for the *style* attribute.

91 1.7 Coordination Protocol Elements

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

96 1.8 Normative References

- 97 **[RDDL]** Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language
98 (RDDL) 2.0", <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>,
99 January 2004.
- 100 **[RFC2119]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels",
101 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 102 **[SOAP 1.1]** W3C Note, "SOAP: Simple Object Access Protocol 1.1",
103 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>, 08 May 2000.
- 104 **[SOAP 1.2]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework",
105 <http://www.w3.org/TR/soap12-part1>, June 2003.
- 106 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth
107 Edition)",<http://www.w3.org/TR/2006/REC-xml-20060816>, 16 August 2006.
- 108 **[XML-ns]** W3C Recommendation, "Namespaces in XML 1.0 (Second Edition),"
109 <http://www.w3.org/TR/2006/REC-xml-names-20060816>, 16 August 2006.
- 110 **[XML-Schema1]** W3C Recommendation, "XML Schema Part 1: Structures Second Edition,"
111 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>, 28 October 2004.
- 112 **[XML-Schema2]** W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition,"
113 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>, 28 October 2004.

114	[WSADDR]	Web Services Addressing (WS-Addressing) 1.0, W3C Recommendation, http://www.w3.org/2005/08/addressing .
115		
116	[WSDL]	Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/2001/NOTE-wsdl-20010315 .
117		
118	[WSPOLICY]	Web Services Policy Framework (WS-Policy), http://schemas.xmlsoap.org/ws/2004/09/policy , VeriSign, Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004.
119		
120		
121	[WSSec]	OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf .
122		
123		
124	[WSSecPolicy]	Web Services Security Policy Language (WS-SecurityPolicy), http://schemas.xmlsoap.org/ws/2005/07/securitypolicy , Microsoft, VeriSign, IBM, and RSA Security Inc., July 2005.
125		
126		
127	[WSSecConv]	Web Services Secure Conversation Language (WS-SecureConversation), http://schemas.xmlsoap.org/ws/2005/02/sc , OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, IBM, VeriSign, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, February 2005.
128		
129		
130		
131	[WSTrust]	Web Services Trust Language (WS-Trust), http://schemas.xmlsoap.org/ws/2005/02/trust , OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, February 2005.
132		
133		
134		

135 1.9 Non-normative References

136		
137	[WSAT]	Web Services Atomic Transaction (WS-AtomicTransaction) http://docs.oasis-open.org/ws-tx/wsat/2006/06 .
138		
139	[WSBA]	Web Services Business Activity (WS-BusinessActivity) http://docs.oasis-open.org/ws-tx/wsba/2006/06 .
140		

2 Coordination Context

141

142 The CoordinationContext is used by applications to pass Coordination information to parties involved in
143 an activity. CoordinationContext elements are propagated to parties which may need to register
144 Participants for the activity. Context propagation may be accomplished using application-defined
145 mechanisms -- e.g. as a header element of a SOAP application message sent to such parties.
146 (Conveying a context in an application message is commonly referred to as flowing the context.) A
147 CoordinationContext provides access to a coordination registration service, a coordination type, and
148 relevant extensions.

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

```
150 <?xml version="1.0" encoding="utf-8"?>
151 <S11:Envelope xmlns:S11="http://www.w3.org/2003/05/soap-envelope">
152   <S11:Header>
153     . . .
154     <wscoor:CoordinationContext
155       xmlns:wsa="http://www.w3.org/2005/08/addressing"
156       xmlns:wscoor="http://docs.oasis-open.org/ws-tx/wscoor/2006/06"
157       xmlns:myApp="http://www.example.com/myApp"
158       S11:mustUnderstand="true">
159       <wscoor:Identifier>
160         http://Fabrikam123.com/SS/1234
161       </wscoor:Identifier>
162       <wscoor:Expires>3000</wscoor:Expires>
163       <wscoor:CoordinationType>
164         http://docs.oasis-open.org/ws-tx/wsac/2006/06
165       </wscoor:CoordinationType>
166       <wscoor:RegistrationService>
167         <wsa:Address>
168           http://Business456.com/mycoordinationservice/registration
169         </wsa:Address>
170         <wsa:ReferenceParameters>
171           <myApp:BetaMark> ... </myApp:BetaMark>
172           <myApp:EBDCode> ... </myApp:EBDCode>
173         </wsa:ReferenceParameters>
174       </wscoor:RegistrationService>
175       <myApp:IsolationLevel>
176         RepeatableRead
177       </myApp:IsolationLevel>
178     </wscoor:CoordinationContext>
179     . . .
180   </S11:Header>
181   </S11:Body>
182   . . .
183 </S11:Body >
184 </S11:Envelope>
185
```

186 When an application propagates an activity using a coordination service, applications **MUST** include a
187 CoordinationContext in the message.

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

190 3 Coordination Service

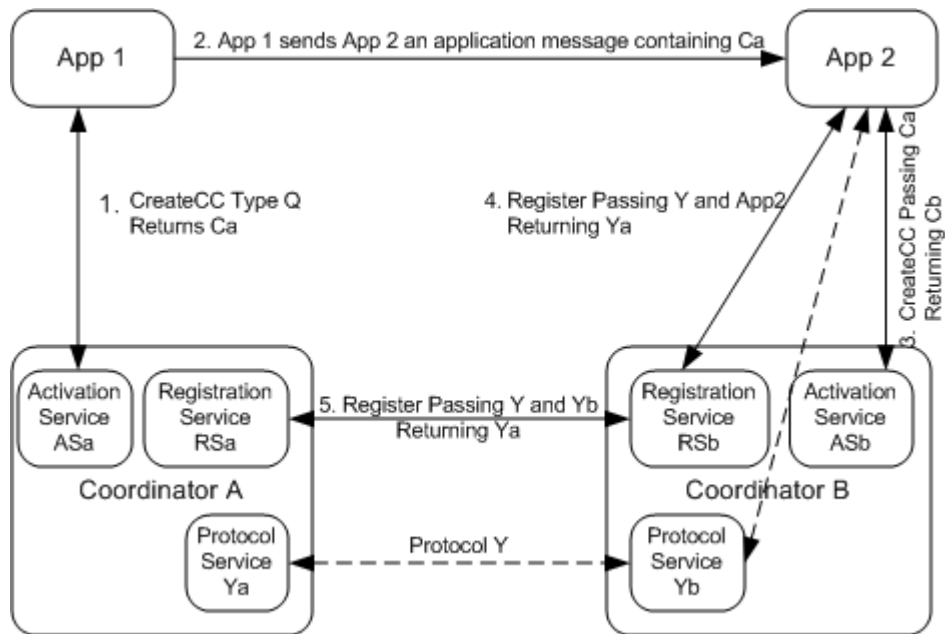
191 The Coordination service (or coordinator) is an aggregation of the following services:

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

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

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

220 Figure 2: Two applications with their own coordinators



221

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

226 3.1 Activation Service

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

228 An application sends:

229 CreateCoordinationContext

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

231 The activation service returns:

232 CreateCoordinationContextResponse

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

234 3.1.1 CreateCoordinationContext

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

240 The following pseudo schema defines this element:

```

241 <CreateCoordinationContext ...>
242   <Expires> ... </Expires>?
243   <CurrentContext> ... </CurrentContext>?
244   <CoordinationType> ... </CoordinationType>
245   ...
246 </CreateCoordinationContext>
247
  
```

248 Expires is an optional element which represents the remaining expiration for the CoordinationContext as
 249 an unsigned integer in milliseconds to be measured from the point at which the context was first received.

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

253 /CreateCoordinationContext/Expires
254 Optional. The expiration for the returned CoordinationContext expressed as an unsigned integer
255 in milliseconds.

256 /CreateCoordinationContext/CurrentContext
257 Optional. If absent, the Activation Service creates a coordination context representing a new,
258 independent activity. If present, the Activation Service creates a coordination context representing
259 a new activity which is related to the existing activity identified by the current coordination context
260 contained in this element. Some examples of potential uses of this type of relationship include
261 interposed subordinate coordination, protocol bridging and coordinator replication.

262 /CreateCoordinationContext /{any}
263 Extensibility elements may be used to convey additional information.

264 /CreateCoordinationContext /@{any}
265 Extensibility attributes may be used to convey additional information.

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

```
267 <CreateCoordinationContext>  
268   <CoordinationType>  
269     http://docs.oasis-open.org/ws-tx/wsat/2006/06  
270   </CoordinationType>  
271 </CreateCoordinationContext>
```

272 **3.1.2 CreateCoordinationContextResponse**

273 This returns the CoordinationContext that was created.

274 The following pseudo schema defines this element:

```
275 <CreateCoordinationContextResponse ...>  
276   <CoordinationContext> ... </CoordinationContext>  
277   ...  
278 </CreateCoordinationContextResponse>
```

279 /CreateCoordinationContext/CoordinationContext

280 This is the created coordination context.

281 /CreateCoordinationContext /{any}
282 Extensibility elements may be used to convey additional information.

283 /CreateCoordinationContext /@{any}
284 Extensibility attributes may be used to convey additional information.

285 The following example illustrates a response:

```
286 <CreateCoordinationContextResponse>  
287   <CoordinationContext>  
288     <Identifier>  
289       http://Business456.com/tm/context1234  
290     </Identifier>  
291     <CoordinationType>  
292       http://docs.oasis-open.org/ws-tx/wsat/2006/06  
293     </CoordinationType>  
294     <RegistrationService>  
295       <wsa:Address>
```

```

296         http://Business456.com/tm/registration
297     </wsa:Address>
298     <wsa:ReferenceParameters>
299         <myapp:PrivateInstance>
300             1234
301         </myapp:PrivateInstance>
302     </wsa:ReferenceParameters>
303 </RegistrationService>
304 </CoordinationContext>
305 </CreateCoordinationContextResponse>

```

306 3.2 Registration Service

307 Once an application has a coordination context from its chosen coordinator, it can register for the activity.
 308 The interface provided to an application registering for an activity and for an interposed coordinator
 309 registering for an activity is the same.

310 The requester sends:

311 Register

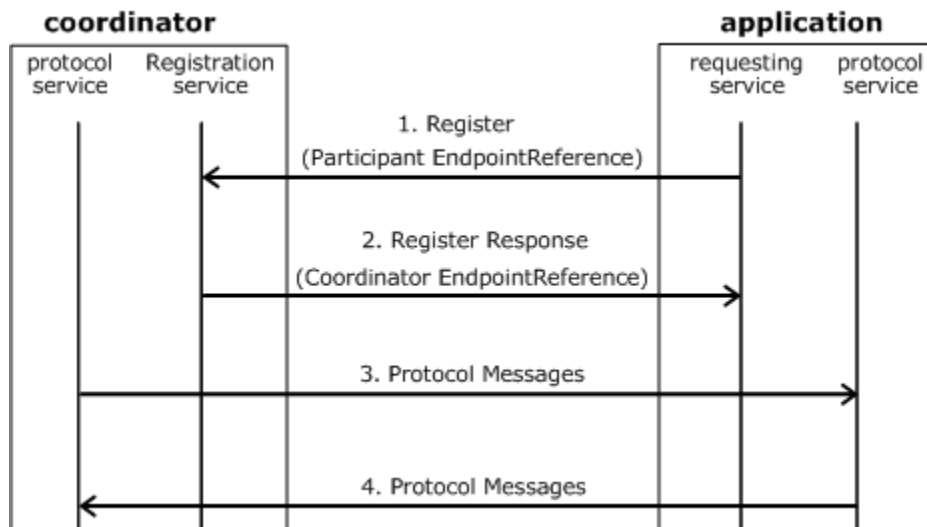
312 The syntax and semantics of this message are defined in Section 3.2.1.

313 The coordinator's registration service responds with:

314 Registration Response

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

316 Figure 3: The usage of Endpoint References during registration



317
 318 In Figure 3, the coordinator provides the Registration Endpoint Reference in the CoordinationContext
 319 during the CreateCoordinationContext operation. The requesting service receives the Registration service
 320 Endpoint Reference in the CoordinationContext in an application message.

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

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

324 3. & 4.) At this point, both sides have the Endpoint References of the other's protocol service, so the
 325 protocol messages can target the other side.

326 These Endpoint References may contain (opaque) wsa:ReferenceParameters to fully qualify the target
327 protocol service endpoint. Endpoint References MUST be interpreted according to the rules defined in
328 WS-Addressing 1.0 Core **[WSADDR]**.

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

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

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

339 **3.2.1 Register Message**

340 The Register request is used to do the following:

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

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

347 The following pseudo schema defines this element:

```
348 <Register ...>  
349   <ProtocolIdentifier> ... </ProtocolIdentifier>  
350   <ParticipantProtocolService> ... </ParticipantProtocolService>  
351   ...  
352 </Register>
```

353 /Register/ProtocolIdentifier

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

355 /Register/ParticipantProtocolService

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

358 /Register/{any}

359 Extensibility elements may be used to convey additional information.

360 / Register/@{any}

361 Extensibility attributes may be used to convey additional information.

362 The following is an example registration message:

```
363 <Register>  
364   <ProtocolIdentifier>  
365     http://docs.oasis-open.org/ws-tx/wsat/2006/06/Volatile2PC  
366   </ProtocolIdentifier>  
367   <ParticipantProtocolService>  
368     <wsa:Address>  
369       http://Adventure456.com/participant2PCservice  
370     </wsa:Address>  
371     <wsa:ReferenceParameters>
```

```
372         <BetaMark> AlphaBetaGamma </BetaMark>
373         </wsa:ReferenceParameters>
374     </ParticipantProtocolService>
375 </Register>
```

376 3.2.2 RegistrationResponse Message

377 The response to the registration message contains the coordinator's Endpoint Reference.

378 The following pseudo schema defines this element:

```
379 <RegisterResponse ...>
380   <CoordinatorProtocolService> ... </CoordinatorProtocolService>
381   ...
382 </RegisterResponse>
```

383 /RegisterResponse/CoordinatorProtocolService

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

386 /RegisterResponse/{any}

387 Extensibility elements may be used to convey additional information.

388 /RegisterResponse /@{any}

389 Extensibility attributes may be used to convey additional information.

390 The following is an example of a RegisterResponse message:

```
391 <RegisterResponse>
392   <CoordinatorProtocolService>
393     <wsa:Address>
394       http://Business456.com/mycoordinationservice/coordinator
395     </wsa:Address>
396     <wsa:ReferenceParameters>
397       <myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>
398     </wsa:ReferenceParameters>
399   </CoordinatorProtocolService>
400 </RegisterResponse>
```

401 .

4 Coordination Faults

402

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

404

```
http://docs.oasis-open.org/ws-tx/wscoor/2006/06/fault
```

405 The protocol faults defined in this section are generated if the condition stated in the preamble is met.
406 When used by a specification that references this specification, these faults are targeted at a destination
407 endpoint according to the protocol fault handling rules defined for that specification.

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

409 [Code] The fault code.

410 [Subcode] The fault subcode.

411 [Reason] A human readable explanation of the fault.

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

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

415

SOAP Version	Sender	Receiver
SOAP 1.2	S12:Sender	S12:Receiver

416

417 The properties above bind to a SOAP 1.2 [**SOAP 1.2**] fault as follows:

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

```
<S12:Envelope>
  <S12:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-tx/wscoor/2006/06/fault
    </wsa:Action>
    <!-- Headers elided for clarity. -->
  </S12:Header>
  <S12:Body>
    <S12:Fault>
      <S12:Code>
        <S12:Value> [Code] </S12:Value>
        <S12:Subcode>
          <S12:Value> [Subcode] </S12:Value>
        </S12:Subcode>
      </S12:Code>
      <S12:Reason>
        <S12:Text xml:lang="en"> [Reason] </S12:Text>
      </S12:Reason>
      <S12:Detail>
        [Detail]
        ...
      </S12:Detail>
    </S12:Fault>
  </S12:Body>
</S12:Envelope>
```

443 The properties bind to a SOAP 1.1 [**SOAP 1.1**] fault as follows:

444

445

446

447

```
<S11:Envelope>
  <S11:Body>
    <S11:Fault>
      <faultcode> [Subcode] </faultcode>
```



```
448     <faultstring xml:lang="en">[Reason]</faultstring>
449     </S11:Fault>
450     </S11:Body>
451 </S11:Envelope>
```

452 **4.1 Invalid State**

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

455 Properties:

456 [Code] Sender

457 [Subcode] wscoor:InvalidState

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

459 [Detail] unspecified

460 **4.2 Invalid Protocol**

461 This fault is sent by either the coordinator or a participant to indicate that the endpoint that generated the
462 fault received a message which is invalid for the protocols supported by the endpoint. This is an
463 unrecoverable condition.

464 Properties:

465 [Code] Sender

466 [Subcode] wscoor:InvalidProtocol

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

468 **4.3 Invalid Parameters**

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

471 Properties:

472 [Code] Sender

473 [Subcode] wscoor:InvalidParameters

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

475 **4.4 Cannot Create Context**

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

478 Properties:

479 [Code] Sender

480 [Subcode] wscoor:CannotCreateContext

481 [Reason] CoordinationContext could not be created.

482 [Detail] unspecified

483 **4.5 Cannot Register Participant**

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

486 Properties:
487 [Code] Sender
488 [Subcode] wscor:CannotRegisterParticipant
489 [Reason] Participant could not be registered.
490 [Detail] unspecified

5 Security Model

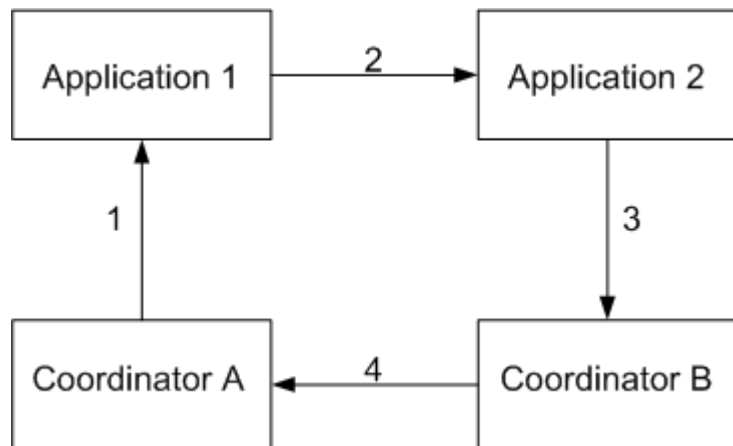
491

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

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

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

501 The following figure illustrates a fairly common usage scenario:



502

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

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

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

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

522 **5.1 CoordinationContext Creation**

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

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

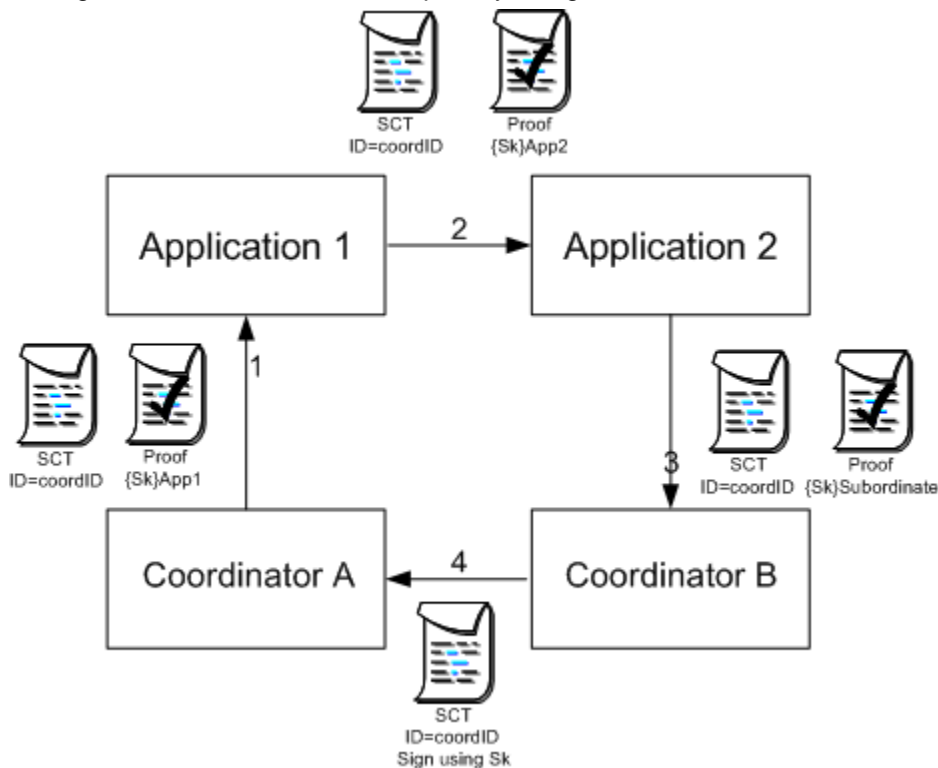
530 Additionally, the coordinator provides a shared secret which is used to indicate authorization to register
 531 with the coordination context by other parties. The secret is communicated using a security token and a
 532 <wst:RequestSecurityTokenResponse> element inside a <wsp:IssuedTokens> header.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

537 **5.2 Registration Rights Delegation**

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

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

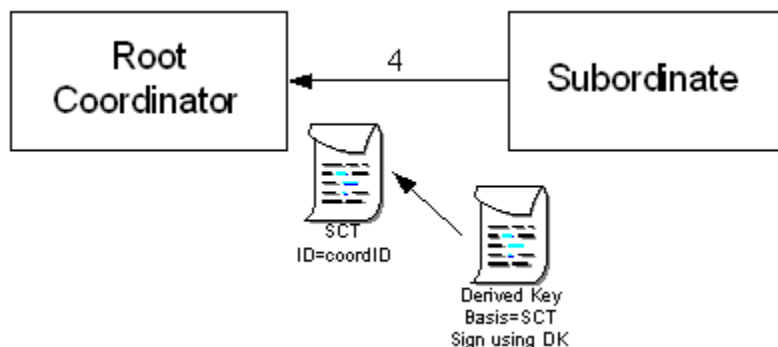
545 The figure below illustrates this simple key delegation model:



546

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

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



559

560

561 6 Security Considerations

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

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

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

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

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

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

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

- 590 • **Message alteration** – Alteration is prevented by including signatures of the message information
591 using WS-Security [**WSec**].
- 592 • **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-
593 Security [**WSec**].
- 594 • **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by
595 comparing secured policies – see WS-Policy [**WSPOLICY**] and WS-SecurityPolicy
596 [**WSecPolicy**]).
- 597 • **Authentication** – Authentication is established using the mechanisms described in WS-Security
598 [**WSec**] and WS-Trust [**WSTrust**]. Each message is authenticated using the mechanisms
599 described in WS-Security [**WSec**].
- 600 • **Accountability** – Accountability is a function of the type of and string of the key and algorithms
601 being used. In many cases, a strong symmetric key provides sufficient accountability. However, in
602 some environments, strong PKI signatures are required.
- 603 • **Availability** – Many services are subject to a variety of availability attacks. Replay is a common
604 attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other

605 attacks, such as network-level denial of service attacks are harder to avoid and are outside the
606 scope of this specification. That said, care should be taken to ensure that minimal processing be
607 performed prior to any authenticating sequences.

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

612 7 Use of WS-Addressing Headers

613 The protocols defined in WS-Coordination use a “request-response” message exchange pattern. The
614 messages used in these protocols can be classified into two types:

- 615 • Request messages: **CreateCoordinationContext** and **Register**.
- 616 • Reply messages: **CreateCoordinationContextResponse** and **RegisterResponse** and the
617 protocol faults defined in [Section 4](#) of this specification.

618 Request messages used in WS-Coordination protocols MUST be constructed in accordance with section
619 3.3 of WS-Addressing 1.0 Core [**WSADDR**].

620 Reply and fault messages used in WS-Coordination protocols MUST be constructed in accordance with
621 section 3.4 of WS-Addressing 1.0 Core [**WSADDR**].

622 Request and reply messages MUST include as the [action] property an action URI that consists of the
623 wscor namespace URI concatenated with the "/" character and the element name of the message. For
624 example:

625 `http://docs.oasis-open.org/ws-tx/wscor/2006/06/Register`

626

8 Glossary

627 The following definitions are used throughout this specification:

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

630 **CoordinationContext:** Contains the activity identifier, its coordination type that represents the collection
631 of behaviors supported by the activity and a Registration service Endpoint Reference that participants can
632 use to register for one or more of the protocols supported by that activity's coordination type.

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

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

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

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

644 **Registration service:** This supports a Register operation that is used by participants to register for any of
645 the coordination protocols supported by a coordination type, such as WS-AtomicTransaction [**WSAT**]
646 Two-Phase Commit (2PC) or WS-BusinessActivity [**WSBA**]
647 BusinessAgreementWithCoordinatorCompletion.

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

651 Appendix A. Acknowledgements

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

658
659 The following individuals have provided invaluable input into the initial contribution: Francisco Curbera
660 (IBM), Sanjay Dalal (BEA Systems), Doug Davis (IBM), Don Ferguson (IBM), Kirill Gavrylyuk (Microsoft),
661 Dan House (IBM), Oisin Hurley (IONA), Frank Leymann (IBM), Thomas Mikalsen (IBM), Jagan Peri
662 (Microsoft), Alex Somogyi (BEA Systems), Stefan Tai (IBM), Satish Thatte (Microsoft), Gary Tully (IONA),
663 Sanjiva Weerawarana (IBM).

664

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

666

667 Participants:

668

669 Charlton Barreto, Adobe Systems, Inc.

670 Martin Chapman, Oracle Corporation

671 Kevin Conner, JBoss Inc.

672 Paul Cotton, Microsoft Corporation

673 Doug Davis, IBM

674 Colleen Evans, Microsoft Corporation

675 Max Feingold, Microsoft Corporation

676 Thomas Freund, IBM

677 Robert Freund, Hitachi, Ltd.

678 Peter Furniss, Choreology Ltd.

679 Marc Goodner, Microsoft Corporation

680 Alastair Green, Choreology Ltd.

681 Daniel House, IBM

682 Ram Jeyaraman, Microsoft Corporation

683 Paul Knight, Nortel Networks Limited

684 Mark Little, JBoss Inc.

685 Jonathan Marsh, Microsoft Corporation

686 Monica Martin, Sun Microsystems

687 Joseph Fialli, Sun Microsystems

688 Eric Newcomer, IONA Technologies

689 Eisaku Nishiyama, Hitachi, Ltd.

690 Alain Regnier, Ricoh Company, Ltd.

691 Ian Robinson, IBM

692 Tom Rutt, Fujitsu Limited

693 Andrew Wilkinson, IBM

694