



Web Services Coordination (WS-Coordination) Version 1.2

OASIS Standard

2 February 2009

Specification URIs:

This Version:

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os/wstx-wscoor-1.2-spec-os.html>
<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.doc> (Authoritative format)
<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf>

Previous Version:

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-cs-01/wstx-wscoor-1.2-spec-cs-01.html>
<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-cs-01.doc> (Authoritative format)
<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-cs-01.pdf>

Latest Approved Version:

<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec.html>
<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec.doc>
<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec.pdf>

Technical Committee:

OASIS Web Services Transaction (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 2008. 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	Conformance	7
1.9	Normative References	7
1.10	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

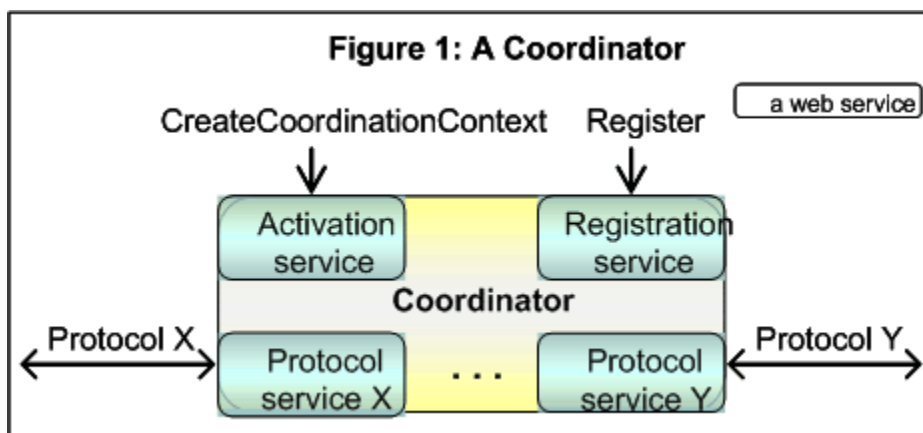
This specification describes a framework for a coordination service (or coordinator) which consists of these component services:

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 There should be no inconsistencies found between any of the normative text within this specification, the
92 normative outlines, the XML Schema definitions, and the WSDL descriptions, and so no general
93 precedence rule is defined. If an inconsistency is observed then it should be reported as a comment on
94 the specification as described in the "Status" section above.

95 1.7 Coordination Protocol Elements

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

100 1.8 Conformance

101 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or
102 REQUIRED level requirements defined herein. A SOAP Node MUST NOT use elements and attributes of
103 the declared XML Namespace (listed on the title page) for this specification within SOAP Envelopes
104 unless it is conformant with this specification.

105 1.9 Normative References

- 106 [RDDL] Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language
107 (RDDL) 2.0", <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>,
108 January 2004.
- 109 [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels",
110 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 111 [SOAP 1.1] W3C Note, "SOAP: Simple Object Access Protocol 1.1,"
112 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>, 08 May 2000.

113	[SOAP 1.2]	W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", http://www.w3.org/TR/2007/REC-soap12-part1-20070427/ , April 2007.
114		
115		
116	[XML]	W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", " http://www.w3.org/TR/2006/REC-xml-20060816 , 16 August 2006.
117		
118	[XML-ns]	W3C Recommendation, "Namespaces in XML 1.0 (Second Edition)," http://www.w3.org/TR/2006/REC-xml-names-20060816 , 16 August 2006.
119		
120	[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second Edition," http://www.w3.org/TR/2004/REC-xmlschema-1-20041028 , 28 October 2004.
121		
122	[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition," http://www.w3.org/TR/2004/REC-xmlschema-2-20041028 , 28 October 2004.
123		
124	[WSADDR]	Web Services Addressing (WS-Addressing) 1.0, W3C Recommendation, http://www.w3.org/2005/08/addressing .
125		
126	[WSDL]	Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/2001/NOTE-wsdl-20010315 .
127		
128	[WSPOLICY]	W3C Recommendation, Web Services Policy 1.5 – Framework (WS-Policy), http://www.w3.org/TR/2007/REC-ws-policy-20070904/ , September 2007.
129		
130	[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 .
131		
132		
133		OASIS Standard, February 2006, Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf .
134		
135		
136		
137	[WSSecPolicy]	OASIS Standard, WS-SecurityPolicy 1.3, February 2009. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.doc .
138		
139	[WSSecConv]	OASIS Standard, WS-SecureConversation 1.4, February 2009. http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc .
140		
141		
142	[WSTrust]	OASIS Standard, WS-Trust 1.4, February 2009. http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc .
143		

144 1.10 Non-normative References

145		
146	[WSAT]	OASIS Standard, Web Services Atomic Transaction (WS-AtomicTransaction) 1.2, February 2009. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os.doc
147		
148	[WSBA]	OASIS Standard, Web Services Business Activity (WS-BusinessActivity) 1.2, February 2009. http://docs.oasis-open.org/ws-tx/wstx-wsba-1.2-spec-os.doc .
149		

2 Coordination Context

150

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

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

```
159 <?xml version="1.0" encoding="utf-8"?>
160 <S11:Envelope xmlns:S11="http://www.w3.org/2003/05/soap-envelope">
161   <S11:Header>
162     . . .
163     <wscoor:CoordinationContext
164       xmlns:wsa="http://www.w3.org/2005/08/addressing"
165       xmlns:wscoor="http://docs.oasis-open.org/ws-tx/wscoor/2006/06"
166       xmlns:myApp="http://www.example.com/myApp"
167       S11:mustUnderstand="true">
168       <wscoor:Identifier>
169         http://Fabrikam123.com/SS/1234
170       </wscoor:Identifier>
171       <wscoor:Expires>3000</wscoor:Expires>
172       <wscoor:CoordinationType>
173         http://docs.oasis-open.org/ws-tx/wsac/2006/06
174       </wscoor:CoordinationType>
175       <wscoor:RegistrationService>
176         <wsa:Address>
177           http://Business456.com/mycoordinationservice/registration
178         </wsa:Address>
179         <wsa:ReferenceParameters>
180           <myApp:BetaMark> ... </myApp:BetaMark>
181           <myApp:EBDCode> ... </myApp:EBDCode>
182         </wsa:ReferenceParameters>
183       </wscoor:RegistrationService>
184       <myApp:IsolationLevel>
185         RepeatableRead
186       </myApp:IsolationLevel>
187     </wscoor:CoordinationContext>
188     . . .
189   </S11:Header>
190   </S11:Body>
191   . . .
192 </S11:Body >
193 </S11:Envelope>
194
```

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

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

199

3 Coordination Service

200

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

201

- 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.

202

203

204

- 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.

205

206

207

- A set of coordination protocol services for each supported coordination type. These are defined in the specification that defines the coordination type.

208

209

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.

210

211

212

213

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.

214

215

216

2. App1 then sends an application message to App2 containing the Context Ca.

217

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.

218

219

220

221

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.

222

223

224

225

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.

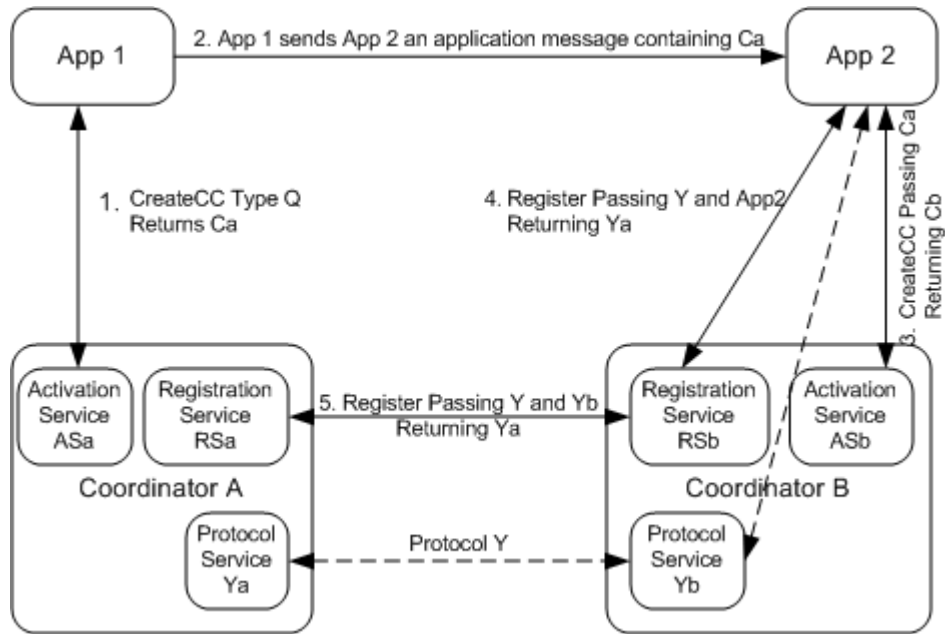
226

227

228

229

Figure 2: Two applications with their own coordinators



230

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

235 3.1 Activation Service

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

237 An application sends:

238 CreateCoordinationContext

239 The structure and semantics of this message are defined in [Section 3.1.1](#).

240 The activation service returns:

241 CreateCoordinationContextResponse

242 The structure and semantics of this message is defined in [Section 3.1.2](#)

243 3.1.1 CreateCoordinationContext

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

249 The following pseudo schema defines this element:

```

250 <CreateCoordinationContext ...>
251   <Expires> ... </Expires>?
252   <CurrentContext> ... </CurrentContext>?
253   <CoordinationType> ... </CoordinationType>
254   ...
255 </CreateCoordinationContext>
256
  
```

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

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

262 /CreateCoordinationContext/Expires
263 Optional. The expiration for the returned CoordinationContext expressed as an unsigned integer
264 in milliseconds.

265 /CreateCoordinationContext/CurrentContext
266 Optional. If absent, the Activation Service creates a coordination context representing a new,
267 independent activity. If present, the Activation Service creates a coordination context representing
268 a new activity which is related to the existing activity identified by the current coordination context
269 contained in this element. Some examples of potential uses of this type of relationship include
270 interposed subordinate coordination, protocol bridging and coordinator replication.

271 /CreateCoordinationContext /{any}
272 Extensibility elements may be used to convey additional information.

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

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

```
276 <CreateCoordinationContext>  
277   <CoordinationType>  
278     http://docs.oasis-open.org/ws-tx/wsat/2006/06  
279   </CoordinationType>  
280 </CreateCoordinationContext>
```

281 **3.1.2 CreateCoordinationContextResponse**

282 This returns the CoordinationContext that was created.

283 The following pseudo schema defines this element:

```
284 <CreateCoordinationContextResponse ...>  
285   <CoordinationContext> ... </CoordinationContext>  
286   ...  
287 </CreateCoordinationContextResponse>
```

288 /CreateCoordinationContext/CoordinationContext

289 This is the created coordination context.

290 /CreateCoordinationContext /{any}
291 Extensibility elements may be used to convey additional information.

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

294 The following example illustrates a response:

```
295 <CreateCoordinationContextResponse>  
296   <CoordinationContext>  
297     <Identifier>  
298       http://Business456.com/tm/context1234  
299     </Identifier>  
300     <CoordinationType>  
301       http://docs.oasis-open.org/ws-tx/wsat/2006/06  
302     </CoordinationType>  
303     <RegistrationService>  
304       <wsa:Address>  
305         http://Business456.com/tm/registration
```

```

306         </wsa:Address>
307         <wsa:ReferenceParameters>
308             <myapp:PrivateInstance>
309                 1234
310             </myapp:PrivateInstance>
311         </wsa:ReferenceParameters>
312     </RegistrationService>
313 </CoordinationContext>
314 </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

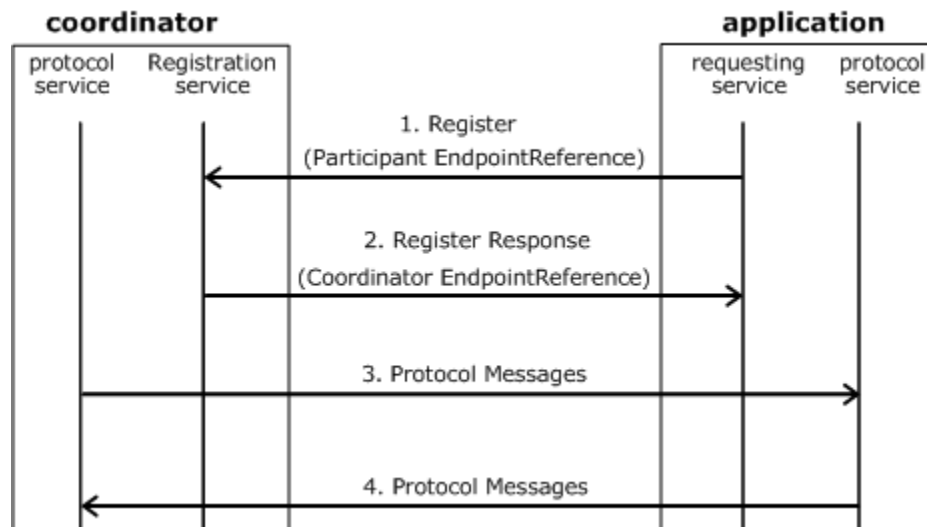
The syntax and semantics of this message are defined in [Section 3.2.1](#).

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:ReferenceParameters to fully qualify the target protocol service endpoint. Endpoint References MUST be interpreted according to the rules defined in WS-Addressing 1.0 Core **[WSADDR]**.

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

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

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

348 3.2.1 Register Message

349 The Register request is used to do the following:

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

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

356 The following pseudo schema defines this element:

```
357 <Register ...>  
358   <ProtocolIdentifier> ... </ProtocolIdentifier>  
359   <ParticipantProtocolService> ... </ParticipantProtocolService>  
360   ...  
361 </Register>
```

362 /Register/ProtocolIdentifier

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

364 /Register/ParticipantProtocolService

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

367 /Register/{any}

368 Extensibility elements may be used to convey additional information.

369 / Register/@{any}

370 Extensibility attributes may be used to convey additional information.

371 The following is an example registration message:

```
372 <Register>  
373   <ProtocolIdentifier>  
374     http://docs.oasis-open.org/ws-tx/wsat/2006/06/Volatile2PC  
375   </ProtocolIdentifier>  
376   <ParticipantProtocolService>  
377     <wsa:Address>  
378       http://Adventure456.com/participant2PCservice  
379     </wsa:Address>  
380     <wsa:ReferenceParameters>  
381       <BetaMark> AlphaBetaGamma </BetaMark>  
382     </wsa:ReferenceParameters>  
383   </ParticipantProtocolService>  
384 </Register>
```

385 3.2.2 RegistrationResponse Message

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

387 The following pseudo schema defines this element:

```
388 <RegisterResponse ...>  
389   <CoordinatorProtocolService> ... </CoordinatorProtocolService>  
390   ...  
391 </RegisterResponse>
```

392 /RegisterResponse/CoordinatorProtocolService

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

395 /RegisterResponse/{any}

396 Extensibility elements may be used to convey additional information.

397 /RegisterResponse /@{any}

398 Extensibility attributes may be used to convey additional information.

399 The following is an example of a RegisterResponse message:

```
400 <RegisterResponse>  
401   <CoordinatorProtocolService>  
402     <wsa:Address>  
403       http://Business456.com/mycoordinationservice/coordinator  
404     </wsa:Address>  
405     <wsa:ReferenceParameters>  
406       <myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>  
407     </wsa:ReferenceParameters>  
408   </CoordinatorProtocolService>  
409 </RegisterResponse>
```

410 .

411 4 Coordination Faults

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

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

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

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

418 [Code] The fault code.

419 [Subcode] The fault subcode.

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

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

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

424

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

425

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

```
427 <S12:Envelope>
428 <S12:Header>
429   <wsa:Action>
430     http://docs.oasis-open.org/ws-tx/wscoor/2006/06/fault
431   </wsa:Action>
432   <!-- Headers elided for clarity. -->
433 </S12:Header>
434 <S12:Body>
435   <S12:Fault>
436     <S12:Code>
437       <S12:Value> [Code] </S12:Value>
438     <S12:Subcode>
439       <S12:Value> [Subcode] </S12:Value>
440     </S12:Subcode>
441   </S12:Code>
442   <S12:Reason>
443     <S12:Text xml:lang="en"> [Reason] </S12:Text>
444   </S12:Reason>
445   <S12:Detail>
446     [Detail]
447     ...
448   </S12:Detail>
449 </S12:Fault>
450 </S12:Body>
451 </S12:Envelope>
```

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

```
453 <S11:Envelope>
454 <S11:Body>
455   <S11:Fault>
456     <faultcode> [Subcode] </faultcode>
```

```
457     <faultstring xml:lang="en">[Reason]</faultstring>
458     </S11:Fault>
459     </S11:Body>
460 </S11:Envelope>
```

461 **4.1 Invalid State**

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

464 Properties:

465 [Code] Sender

466 [Subcode] wscoor:InvalidState

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

468 [Detail] unspecified

469 **4.2 Invalid Protocol**

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

473 Properties:

474 [Code] Sender

475 [Subcode] wscoor:InvalidProtocol

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

477 **4.3 Invalid Parameters**

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

480 Properties:

481 [Code] Sender

482 [Subcode] wscoor:InvalidParameters

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

484 **4.4 Cannot Create Context**

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

487 Properties:

488 [Code] Sender

489 [Subcode] wscoor:CannotCreateContext

490 [Reason] CoordinationContext could not be created.

491 [Detail] unspecified

492 **4.5 Cannot Register Participant**

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

495 Properties:
496 [Code] Sender
497 [Subcode] wscoor:CannotRegisterParticipant
498 [Reason] Participant could not be registered.
499 [Detail] unspecified

5 Security Model

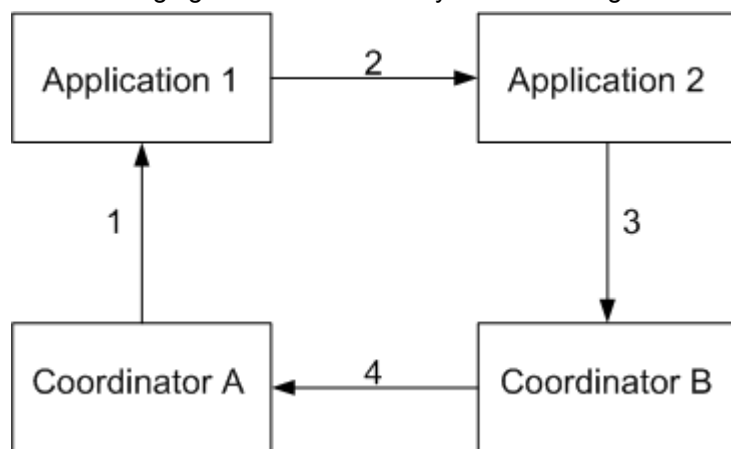
500

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

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

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

510 The following figure illustrates a fairly common usage scenario:



511

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

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

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

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

531 **5.1 CoordinationContext Creation**

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

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

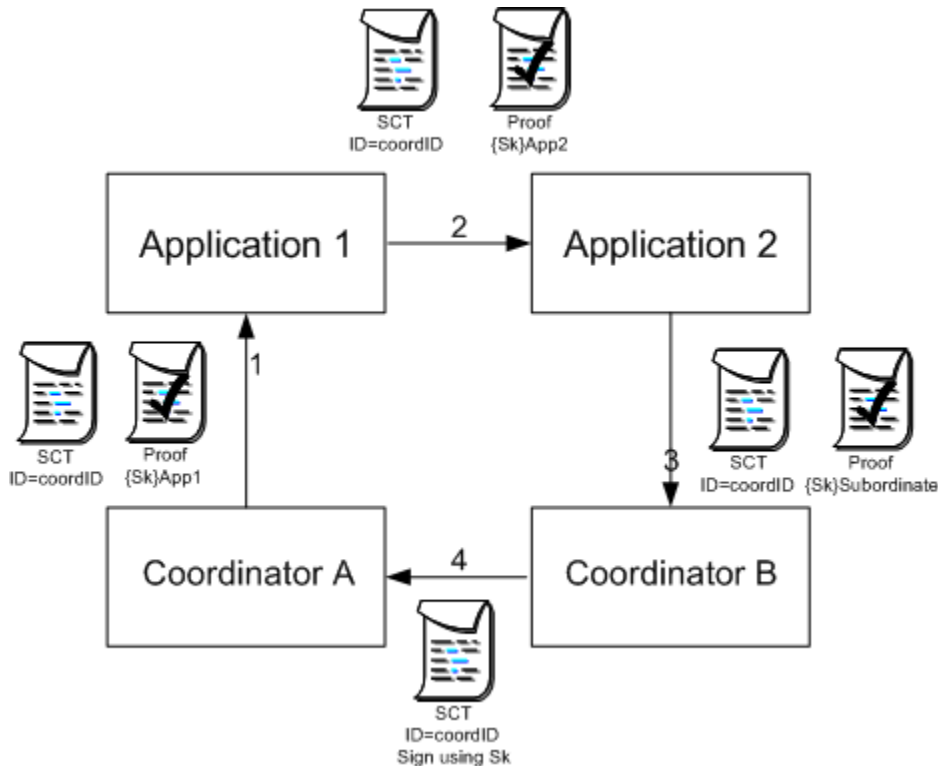
539 Additionally, the coordinator provides a shared secret which is used to indicate authorization to register
540 with the coordination context by other parties. The secret is communicated using a security token and a
541 <wst:RequestSecurityTokenResponse> element inside a <wst:IssuedTokens> header.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

546 **5.2 Registration Rights Delegation**

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

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

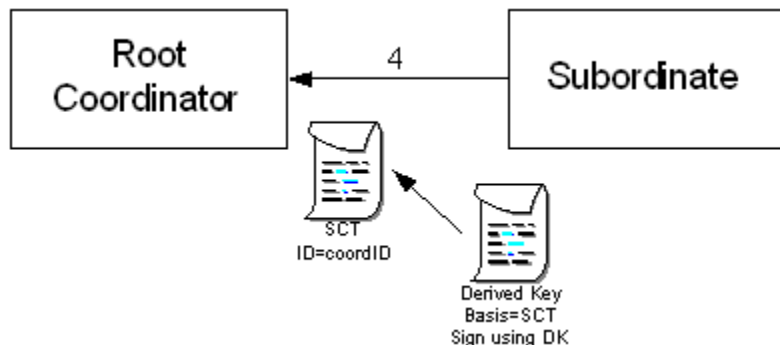
554 The figure below illustrates this simple key delegation model:



555

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

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



568
569

570 6 Security Considerations

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

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

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

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

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

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

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

- 598 • **Message alteration** – Alteration is prevented by including signatures of the message information
599 using WS-Security [WSSec].
- 600 • **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-
601 Security [WSSec].
- 602 • **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by
603 comparing secured policies – see WS-Policy [WSPOLICY] and WS-SecurityPolicy
604 [WSSecPolicy]).
- 605 • **Authentication** – Authentication is established using the mechanisms described in WS-Security
606 [WSSec] and WS-Trust [WSTrust]. Each message is authenticated using the mechanisms
607 described in WS-Security [WSSec].
- 608 • **Accountability** – Accountability is a function of the type of and string of the key and algorithms
609 being used. In many cases, a strong symmetric key provides sufficient accountability. However, in
610 some environments, strong PKI signatures are required.
- 611 • **Availability** – Many services are subject to a variety of availability attacks. Replay is a common
612 attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other
613 attacks, such as network-level denial of service attacks are harder to avoid and are outside the

614 scope of this specification. That said, care should be taken to ensure that minimal processing be
615 performed prior to any authenticating sequences.

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

620 7 Use of WS-Addressing Headers

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

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

626 Request messages used in WS-Coordination protocols MUST be constructed in accordance with section
627 3.3 of WS-Addressing 1.0 Core [**WSADDR**].

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

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

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

634

8 Glossary

635 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

659 Appendix A. Acknowledgements

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

666
667 The following individuals have provided invaluable input into the initial contribution: Francisco Curbera
668 (IBM), Sanjay Dalal (BEA Systems), Doug Davis (IBM), Don Ferguson (IBM), Kirill Gavrylyuk (Microsoft),
669 Dan House (IBM), Oisin Hurley (IONA), Frank Leymann (IBM), Thomas Mikalsen (IBM), Jagan Peri
670 (Microsoft), Alex Somogyi (BEA Systems), Stefan Tai (IBM), Satish Thatte (Microsoft), Gary Tully (IONA),
671 Sanjiva Weerawarana (IBM).

672

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

674

675 Participants:

676

677 Charlton Barreto, Adobe Systems, Inc.
678 Martin Chapman, Oracle Corporation
679 Kevin Conner, JBoss Inc.
680 Paul Cotton, Microsoft Corporation
681 Doug Davis, IBM
682 Colleen Evans, Microsoft Corporation
683 Max Feingold, Microsoft Corporation
684 Thomas Freund, IBM
685 Robert Freund, Hitachi, Ltd.
686 Peter Furniss, Choreology Ltd.
687 Marc Goodner, Microsoft Corporation
688 Alastair Green, Choreology Ltd.
689 Daniel House, IBM
690 Ram Jeyaraman, Microsoft Corporation
691 Paul Knight, Nortel Networks Limited
692 Mark Little, JBoss Inc.
693 Jonathan Marsh, Microsoft Corporation
694 Monica Martin, Sun Microsystems
695 Joseph Fialli, Sun Microsystems
696 Eric Newcomer, IONA Technologies
697 Eisaku Nishiyama, Hitachi, Ltd.
698 Alain Regnier, Ricoh Company, Ltd.
699 Ian Robinson, IBM
700 Tom Rutt, Fujitsu Limited
701 Andrew Wilkinson, IBM

702