



Web Services Coordination (WS-Coordination) 1.1

Working Draft, November 22, 2005

Document Identifier:

wstx-wscoor-1.1-spec-wd-01

Location:

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

Technical Committee:

OASIS WS-TX TC

Chair(s):

Eric Newcomer, Iona
Ian Robinson, IBM

Editor(s):

Max Feingold, Microsoft

Abstract:

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

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

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

Status:

This document is a working draft.

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

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

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

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

Notices

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's procedures with respect to rights in OASIS specifications can be found at 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 implementors or users of this specification, can be obtained from the OASIS President.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS President.

Copyright © OASIS Open 2005. *All Rights Reserved.*

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 paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document 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 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1	Introduction.....	6
1.1	Model	6
1.2	Composable Architecture	7
1.3	Extensibility	7
1.4	Terminology	8
1.5	Namespace.....	9
1.6	XSD and WSDL Files	9
1.7	Coordination Protocol Elements	10
1.8	References.....	10
2	Coordination Context.....	12
3	Coordination Service	14
3.1	Activation Service	15
3.1.1	CreateCoordinationContext.....	15
3.1.2	CreateCoordinationContextResponse	16
3.2	Registration Service.....	17
3.2.1	Register Message	19
3.2.2	RegistrationResponse Message	20
4	Coordination Faults	21
4.1	Invalid State	22
4.2	Invalid Protocol	22
4.3	Invalid Parameters	23
4.4	No Activity	23
4.5	Context Refused	23
4.6	Already Registered	24
5	Security Model.....	25
5.1	CoordinationContext Creation	26
5.2	Registration Rights Delegation	26
6	Security Considerations	29
7	Glossary	31
	Appendix A. Acknowledgements	32
	Appendix B. Revision History.....	33
	Appendix C. Non-normative Text.....	34

1 Introduction

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

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

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

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

1.1 Model

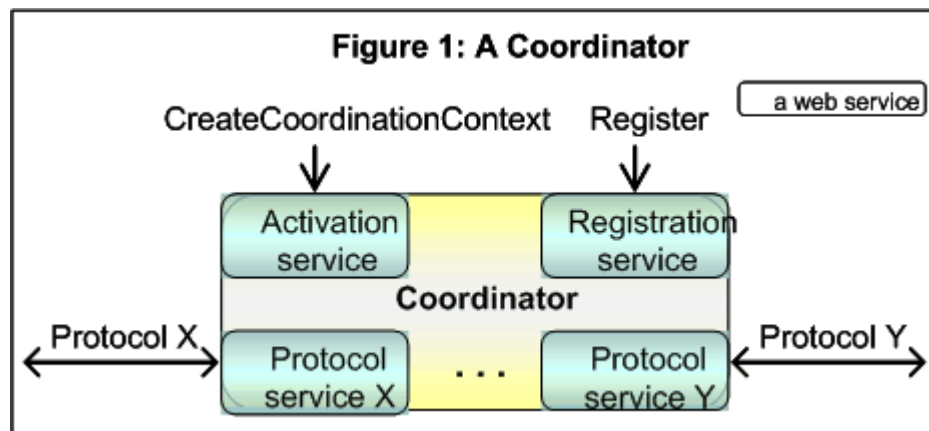
This specification describes a framework for a coordination service (or coordinator) which consists of these component services:

An Activation service with an operation that enables an application to create a coordination instance or context.

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

A coordination type-specific set of coordination protocols.

This is illustrated below in Figure 1.



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.

33 1.2 Composable Architecture

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

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

43 1.3 Extensibility

44 The specification provides for extensibility and flexibility along two dimensions. The framework allows for:
45 The publication of new coordination protocols.

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

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

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

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

57 1.4 Terminology

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

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

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 XSD).
- 71 • Characters are appended to elements, attributes, and <!-- descriptions --> as follows: "?" (0 or 1), "*" (0 or more), "+" (1 or more). The characters "[" and "]" are used to indicate that contained items are to
72 be treated as a group with respect to the "?", "*", or "+" characters.
- 73 • The XML namespace prefixes (defined below) are used to indicate the namespace of the element
74 being defined.
- 75 • Examples starting with <?xml contain enough information to conform to this specification; others
76 examples are fragments and require additional information to be specified in order to conform.
77

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

80 1.5 Namespace

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

82 `http://schemas.xmlsoap.org/ws/2004/10/wscoor`

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

84 The following namespaces are used in this document:

Prefix	Namespace
S	http://www.w3.org/2003/05/soap-envelope
wscoor	http://schemas.xmlsoap.org/ws/2004/10/wscoor
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing

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

87 `http://schemas.xmlsoap.org/ws/2004/10/wscoor/Register`

88

89 1.6 XSD and WSDL Files

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

92 <http://schemas.xmlsoap.org/ws/2004/10/wscoor/wscoor.xsd>

93 <http://schemas.xmlsoap.org/ws/2004/10/wscoor/wscoor.wsdl>

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

96 1.7 Coordination Protocol Elements

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

101

102 1.8 References

- 103 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
104 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 105 **[SOAP]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
- 106 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
107 Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August
108 1998.
- 109 **[XML-ns]** W3C Recommendation, "Namespaces in XML," 14 January 1999.
- 110 **[XML-Schema1]** W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001.
- 111 **[XML-Schema2]** W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.
- 112 **[WSADDR]** Web Services Addressing (WS-Addressing), Microsoft, IBM, Sun, BEA Systems,
113 SAP, Sun, August 2004

114	[WSAT]	Web Services Atomic Transaction (WS-AtomicTransaction), Arjuna Technologies
115		Ltd., BEA Systems, Hitachi Ltd., IBM, IONA Technologies and Microsoft, August
116		2005
117	[WSDL]	Web Services Description Language (WSDL) 1.1
118		"http://www.w3.org/TR/2001/NOTE-wsdl-20010315"
119	[WSPOLICY]	Web Services Policy Framework (WS-Policy), VeriSign, Microsoft, Sonic
120		Software, IBM, BEA Systems, SAP, September 2004
121	[WSSec]	OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message
122		Security 1.0 (WS-Security 2004)"
123	[WSSecPolicy]	Web Services Security Policy Language (WS-SecurityPolicy), Microsoft,
124		VeriSign, IBM, and RSA Security Inc., July 2005
125	[WSSecConv]	Web Services Secure Conversation Language (WS-SecureConversation),
126		OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, IBM, VeriSign, BEA
127		Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates,
128		February 2005
129	[WSTrust]	Web Services Trust Language (WS-Trust), OpenNetwork, Layer7, Netegrity,
130		Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping
131		Identity, Westbridge, Computer Associates, February 2005
132		
133		

134 2 Coordination Context

135 The CoordinationContext is a Context type that is used to pass Coordination information to parties
136 involved in a coordination service. CoordinationContext elements are placed within application
137 messages. Conveying a context on an application message is commonly referred to as flowing the
138 context. A CoordinationContext provides access to a coordination registration service, a coordination
139 type, and relevant extensions.

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

```
141 <?xml version="1.0" encoding="utf-8"?>
142 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
143   <S:Header>
144     . . .
145     <wscoor:CoordinationContext
146       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
147       xmlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wscoor"
148       xmlns:myApp="http://fabrikaml23.com/myApp"
149       S:mustUnderstand="true">
150       <wscoor:Identifier>
151         http://Fabrikaml23.com/SS/1234
152       </wscoor:Identifier>
153       <wscoor:Expires>3000</wscoor:Expires>
154       <wscoor:CoordinationType>
155         http://schemas.xmlsoap.org/ws/2004/10/wsat
156       </wscoor:CoordinationType>
157       <wscoor:RegistrationService>
158         <wsa:Address>
159           http://Business456.com/mycoordination-service/registration
160         </wsa:Address>
161         <wsa:ReferenceProperties>
162           <myApp:BetaMark> ... </myApp:BetaMark>
163           <myApp:EBDCCode> ... </myApp:EBDCCode>
164         </wsa:ReferenceProperties>
165       </wscoor:RegistrationService>
166       <myApp:IsolationLevel>
167         RepeatableRead
168       </myApp:IsolationLevel>
169     </wscoor:CoordinationContext>
170     . . .
171   </S:Header>
172   </S:Body>
173   . . .
174 </S:Body >
175 </S:Envelope>
176
```

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

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

181 3 Coordination Service

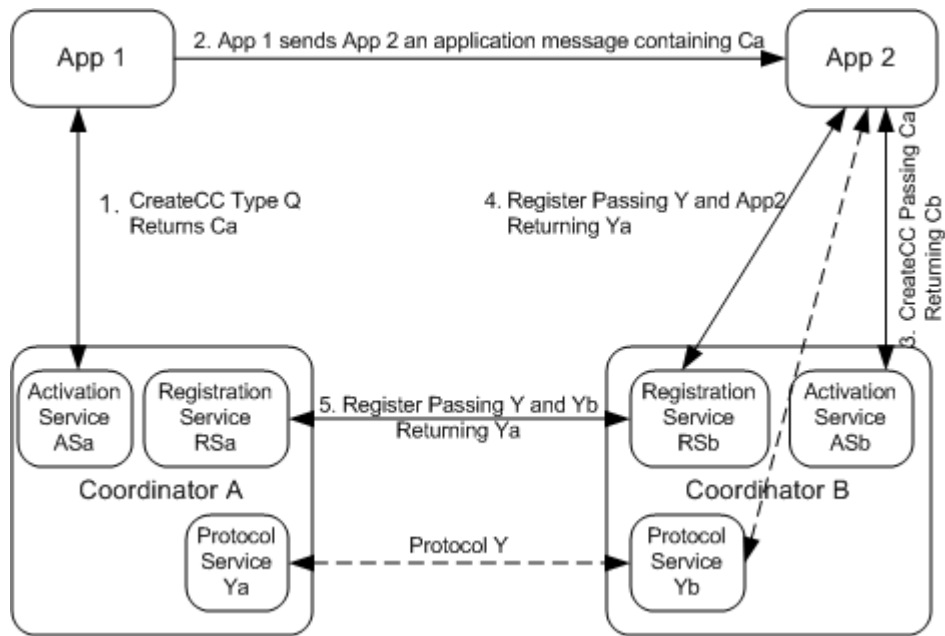
182 The Coordination service (or coordinator) is an aggregation of the following services:

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

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

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

207 Figure 2: Two applications with their own coordinators



208
209

210 3.1 Activation Service

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

212 An application sends:

213 CreateCoordinationContext

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

215 The activation service returns:

216 CreateCoordinationContextResponse

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

218 3.1.1 CreateCoordinationContext

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

224 The following pseudo schema defines this element:

```

225 <CreateCoordinationContext ... >
226   <Expires> ... </Expires>?
227   <CurrentContext> ... </CurrentContext>?
228   <CoordinationType> ... </CoordinationType>
229   ...
230 </CreateCoordinationContext>
231

```

232 /CreateCoordinationContext/CoordinationType

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

235 /CreateCoordinationContext/Expires

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

238 /CreateCoordinationContext/CurrentContext

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

241 /CreateCoordinationContext /{any}

242 Extensibility elements may be used to convey additional information.

243 /CreateCoordinationContext /@{any}

244 Extensibility attributes may be used to convey additional information.

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

```
246 <CreateCoordinationContext>  
247   <CoordinationType>  
248     http://schemas.xmlsoap.org/ws/2004/10/wsat  
249   </CoordinationType>  
250 </CreateCoordinationContext>
```

251 **3.1.2 CreateCoordinationContextResponse**

252 This returns the CoordinationContext that was created.

253 The following pseudo schema defines this element:

```
254 <CreateCoordinationContextResponse ...>  
255   <CoordinationContext> ... </CoordinationContext>  
256   ...  
257 </CreateCoordinationContextResponse>
```

258 /CreateCoordinationContext/CoordinationContext

259 This is the created coordination context.

260 /CreateCoordinationContext /{any}

261 Extensibility elements may be used to convey additional information.

262 /CreateCoordinationContext /@{any}

263 Extensibility attributes may be used to convey additional information.

264 The following example illustrates a response:

```
265 <CreateCoordinationContextResponse>  
266   <CoordinationContext>  
267     <Identifier>  
268       http://Business456.com/tm/context1234  
269     </Identifier>  
270     <CoordinationType>  
271       http://schemas.xmlsoap.org/ws/2004/10/wsat  
272     </CoordinationType>  
273     <RegistrationService>  
274       <wsa:Address>  
275         http://Business456.com/tm/registration  
276       </wsa:Address>  
277       <wsa:ReferenceProperties>  
278         <myapp:PrivateInstance>  
279           1234  
280         </myapp:PrivateInstance>  
281       </wsa:ReferenceProperties>  
282     </RegistrationService>  
283   </CoordinationContext>  
284 </CreateCoordinationContextResponse>
```

285 **3.2 Registration Service**

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

289 The requester sends:

290 Register

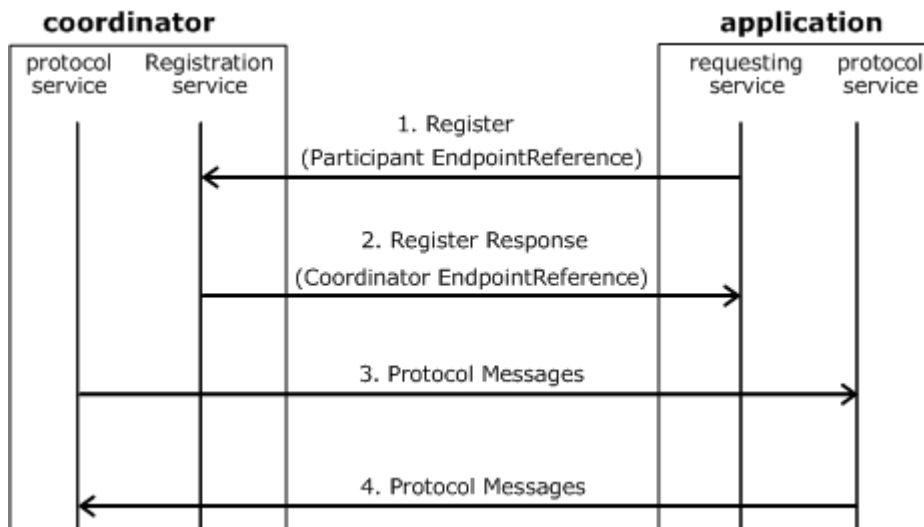
291 The syntax and semantics of this message are defined in Section 3.2.1.

292 The coordinator's registration service responds with:

293 Registration Response

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

295 Figure 3: The usage of Endpoint References during registration



296

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

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

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

303 3. & 4.) At this point, both sides have the Endpoint References of the other's protocol service, so the
304 protocol messages can target the other side.

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

308 **3.2.1 Register Message**

309 The Register request is used to do the following:

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

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

316 The following pseudo schema defines this element:

```
317 <Register ...>  
318   <ProtocolIdentifier> ... </ProtocolIdentifier>  
319   <ParticipantProtocolService> ... </ParticipantProtocolService>  
320   ...  
321 </Register>
```

322 /Register/ProtocolIdentifier

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

324 /Register/ParticipantProtocolService

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

327 /Register/{any}

328 Extensibility elements may be used to convey additional information.

329 / Register/@{any}

330 Extensibility attributes may be used to convey additional information.

331 The following is an example registration message:

```
332 <Register>  
333   <ProtocolIdentifier>  
334     http://schemas.xmlsoap.org/ws/2004/10/wsat/Volatile2PC  
335   </ProtocolIdentifier>  
336   <ParticipantProtocolService>  
337     <wsa:Address>  
338       http://Adventure456.com/participant2PCservice  
339     </wsa:Address>  
340     <wsa:ReferenceProperties>  
341       <BetaMark> AlphaBetaGamma </BetaMark>  
342     </wsa:ReferenceProperties>  
343   </ParticipantProtocolService>  
344 </Register>
```

345 3.2.2 RegistrationResponse Message

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

347 The following pseudo schema defines this element:

```
348 <RegisterResponse ...>  
349   <CoordinatorProtocolService> ... </CoordinatorProtocolService>  
350   ...  
351 </RegisterResponse>
```

352 /RegisterResponse/CoordinatorProtocolService

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

355 /RegisterResponse/{any}

356 Extensibility elements may be used to convey additional information.

357 /RegisterResponse /@{any}

358 Extensibility attributes may be used to convey additional information.

359 The following is an example of a RegisterResponse message:

```
360 <RegisterResponse>
361   <CoordinatorProtocolService>
362     <wsa:Address>
363       http://Business456.com/mycoordinationservice/coordinator
364     </wsa:Address>
365     <wsa:ReferenceProperties>
366       <myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>
367     </wsa:ReferenceProperties>
368   </CoordinatorProtocolService>
369 </RegisterResponse>
```

370 .

371 4 Coordination Faults

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

373 `http://schemas.xmlsoap.org/ws/2004/10/wscor/fault`

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

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

377 [Code] The fault code.

378 [Subcode] The fault subcode.

379 [Reason] The English language reason element.

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

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

383

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

384

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

```
386 <S:Envelope>  
387 <S:Header>  
388 <wsa:Action>  
389 http://schemas.xmlsoap.org/ws/2004/10/wscor/fault  
390 </wsa:Action>  
391 <!-- Headers elided for clarity. -->  
392 </S:Header>  
393 <S:Body>  
394 <S:Fault>  
395 <S:Code>  
396 <S:Value>[Code]</S:Value>  
397 <S:Subcode>  
398 <S:Value>[Subcode]</S:Value>  
399 </S:Subcode>  
400 </S:Code>  
401 <S:Reason>  
402 <S:Text xml:lang="en">[Reason]</S:Text>  
403 </S:Reason>  
404 <S:Detail>  
405 [Detail]  
406 ...  
407 </S:Detail>  
408 </S:Fault>  
409 </S:Body>  
410 </S:Envelope>
```

411 The properties bind to a SOAP 1.1 fault as follows:

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

```
417 </S11:Fault>
418 </S11:Body>
419 </S11:Envelope>
```

420 **4.1 Invalid State**

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

423 Properties:

424 [Code] Sender

425 [Subcode] wscoor:InvalidState

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

427 [Detail] unspecified

428 .

429 **4.2 Invalid Protocol**

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

432 Properties:

433 [Code] Sender

434 [Subcode] wscoor:InvalidProtocol

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

436 **4.3 Invalid Parameters**

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

439 Properties:

440 [Code] Sender

441 [Subcode] wscoor:InvalidParameters

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

443 **4.4 No Activity**

444 This fault is sent by the coordinator if the participant has been quiet for too long and is presumed to have
445 ended.

446 Properties:

447 [Code] Sender

448 [Subcode] wscoor:NoActivity

449 [Reason] The participant is not responding and is presumed to have ended.

450 [Detail] unspecified

451 **4.5 Context Refused**

452 This fault is sent to a coordinator to indicate that the endpoint cannot accept a context which it was
453 passed:

454 Properties:

455 [Code] Sender
456 [Subcode] wscor:ContextRefused
457 [Reason] The coordination context that was provided could not be accepted.
458 [Detail] unspecified
459

460 **4.6 Already Registered**

461 This fault is sent to a participant if the coordinator detects that the participant attempted to
462 register for the same protocol of the same activity more than once.

463 Properties:

464 [Code] Sender
465 [Subcode] wscor:AlreadyRegistered
466 [Reason] The participant has already registered for the same protocol.
467 [Detail] unspecified
468

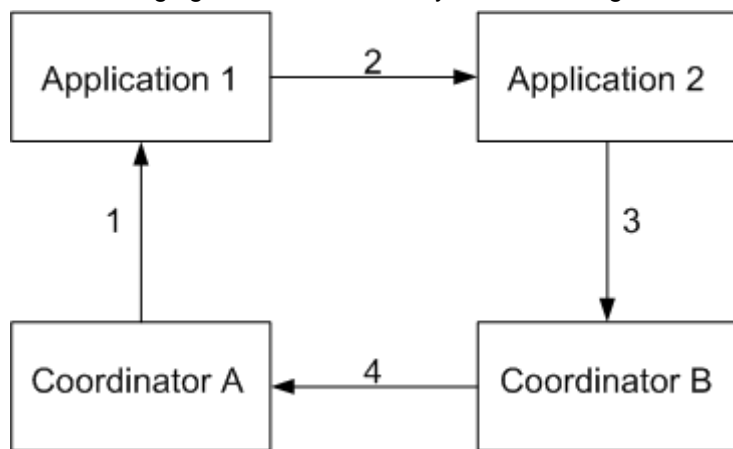
469 5 Security Model

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

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

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

479 The following figure illustrates a fairly common usage scenario:



480
481 In the figure above, step 1 involves the creation and subsequent communication between
482 the creator of the context and the coordinator A (root). It should be noted that this may be
483 a private or local communication. Step 2 involves the delegation of the right to register
484 with the activity using the information from the coordination context and subsequent
485 application messages between two applications (and may include middleware involvement)
486 which are participants in the activity. Step 3 involves delegation of the right to register with
487 the activity to coordinator B (subordinate) that manages all access to the activity on behalf
488 of the second, and possibly other parties. Again note that this may also be a private or
489 local communication. Step 4 involves registration with the coordinator A by the coordinator
490 B and proof that registration rights were delegated.

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

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

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

502 **5.1 CoordinationContext Creation**

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

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

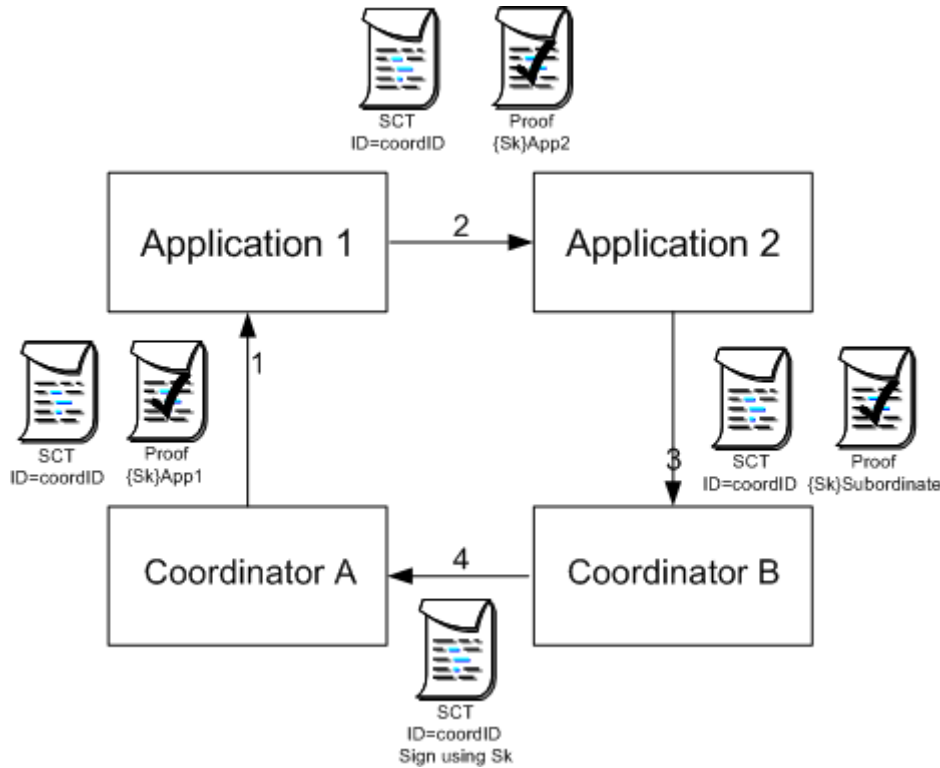
510 Additionally, the coordinator provides a shared secret which is used to indicate authorization to register
511 with the coordination context by other parties. The secret is communicated using a security token and a
512 <wst:RequestSecurityTokenResponse> element inside a <wst:IssuedTokens> header.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

517 **5.2 Registration Rights Delegation**

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

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

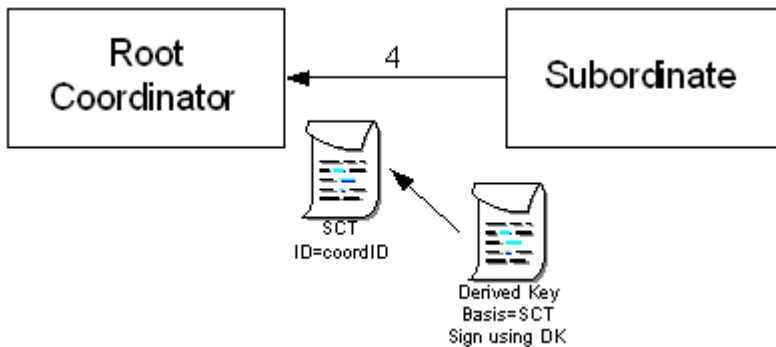
525 The figure below illustrates this simple key delegation model:



526

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

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



539

540

541 6 Security Considerations

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

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

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

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

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

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

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

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

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

590 7 Glossary

591 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

614

615 **Appendix A. Acknowledgements**

616 This document is based on initial contribution to OASIS WS-TX Technical Committee by the
617 following authors: Luis Felipe Cabrera, Microsoft, George Copeland, Microsoft, Max Feingold,
618 Microsoft,(Editor), Robert W Freund, Hitachi, Tom Freund, IBM, Jim Johnson, Microsoft, Sean Joyce,
619 IONA, Chris Kaler, Microsoft, Johannes Klein, Microsoft, David Langworthy, Microsoft, Mark Little, Arjuna
620 Technologies, Anthony Nadalin, IBM, Eric Newcomer, IONA, David Orchard, BEA Systems, Ian
621 Robinson, IBM, John Shewchuk, Microsoft, Tony Storey, IBM/
622

623 The following individuals have provided invaluable input into the initial contribution: Francisco Curbera,
624 IBM, Sanjay Dalal, BEA Systems, Doug Davis, IBM, Don Ferguson, IBM, Kirill Gavrylyuk, Microsoft, Dan
625 House, IBM, Oisin Hurley, IONA, Frank Leymann, IBM, Thomas Mikalsen, IBM, Jagan Peri, Microsoft,
626 Alex Somogyi, BEA Systems, Stefan Tai, IBM, Satish Thatte, Microsoft, Gary Tully, IONA, Sanjiva
627 Weerawarana, IBM/
628

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

631
632 TBD

633
634 [Participant Name, Affiliation | Individual Member]
635 [Participant Name, Affiliation | Individual Member]
636

637

Appendix B. Revision History

638

[optional; should not be included in OASIS Standards]

639

Revision	Date	Editor	Changes Made
01	05-11-22	Max Feingold	Initial Working Draft

640

641 **Appendix C. Non-normative Text**

642