



Web Services Business Activity 1.1 (WS-Business Activity)

Working Draft, November 22, 2005

Document Identifier:

wstx-wsba-1.1-spec-wd-01

Location:

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

Technical Committee:

OASIS WS-Tx TC

Chair(s):

Eric Newcomer, Iona
Ian Robinson, IBM

Editor(s):

Tom Freund, IBM <tjfreund@us.ibm.com>
Alastair Green, Choreology Ltd. <alastair.green@choreology.com>
John Harby, Independent Consultant <jharby@gmail.com>
Mark Little, Arjuna Technologies Ltd. <mark.little@arjuna.com>

Abstract:

This specification provides the definition of the business activity coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification. The specification defines two specific agreement coordination protocols for the business activity coordination type: BusinessAgreementWithParticipantCompletion, and BusinessAgreementWithCoordinatorCompletion. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of long-running distributed activities.

Status:

This document is published by the WS-Tx TX as 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.....	4
1.1	Model.....	5
1.2	Terminology.....	5
1.3	Namespace.....	5
1.3.1	Prefix Namespace.....	5
1.4	Normative References.....	6
1.5	Non-Normative References.....	6
2	Using WS-Coordination.....	7
2.1	Coordination Context.....	7
3	Coordination Types and Protocols.....	8
3.1	BusinessAgreementWithParticipantCompletion Protocol.....	8
3.2	BusinessAgreementWithCoordinatorCompletion Protocol.....	10
4	WS-BA Policy Assertions.....	12
4.1	Assertion Models.....	12
4.2	Normative Outlines.....	12
4.3	Assertion Attachment.....	13
4.4	Assertion Example.....	13
5	Security Considerations.....	15
6	Interoperability Considerations.....	16
7	Glossary.....	17
	Appendix A. Acknowledgements.....	18
	Appendix B. Revision History.....	19
	Appendix C. State Tables for the Agreement Protocols.....	20
	C.1. Participant view of BusinessAgreementWithParticipantCompletion.....	20
	C.2. Coordinator view of BusinessAgreementWithParticipantCompletion.....	21
	C.3. Participant view of BusinessAgreementWithCoordinatorCompletion.....	22
	C.4. Coordinator view of BusinessAgreementWithCoordinatorCompletion.....	23

1 Introduction

The current set of Web service specifications [WSDL] [SOAP] defines protocols for Web service interoperability. Web services increasingly tie together a number of participants forming large distributed applications. The resulting activities may have complex structure and relationships.

The WS-Coordination specification defines an extensible framework for defining coordination types. A coordination type can have multiple coordination protocols, each intended to coordinate a different role that a Web service plays in the activity.

To establish the necessary relationships between participants, messages exchanged between participants carry a CoordinationContext. The CoordinationContext includes a Registration service Endpoint Reference of a Coordination service. Participants use that Registration service to register for one or more of the protocols supported by that activity.

To understand the protocol described in this specification, the following assumptions are made:

- The reader is familiar with the WS-Coordination [WSCOOR] specification that defines the framework for the WS-BusinessActivity coordination protocols.
- The reader is familiar with WS-Addressing [WSADDR] and WS-Policy [WSPOLICY].

This specification provides the definition of a business activity coordination type used to coordinate activities that apply business logic to handle exceptions that occur during the execution of activities of a business process. Actions are applied immediately and are permanent. Compensating actions may be invoked in the event of an error. The Business Activity specification defines protocols that enable existing business process and work flow systems to wrap their proprietary mechanisms and interoperate across trust boundaries and different vendor implementations.

Business Activities have the following characteristics:

- A business activity may consume many resources over a long duration.
- There may be a significant number of atomic transactions involved.
- Individual tasks within a business activity can be seen prior to the completion of the business activity, their results may have an impact outside of the computer system.
- Responding to a request may take a very long time. Human approval, assembly, manufacturing, or delivery may have to take place before a response can be sent.
- In the case where a business exception requires an Activity to be logically undone, abort is typically not sufficient. Exception handling mechanisms may require business logic, for example in the form of a compensation task, to reverse the effects of a previously completed task.
- Participants in a business activity may be in different domains of trust where all trust relationships are established explicitly.

These characteristics lead to a design point, with the following assumptions:

- All state transitions are reliably recorded, including application state and coordination metadata.
- All notifications are acknowledged in the protocol to ensure a consistent view of state between the coordinator and participant.
- Each notification is defined as an individual message. Transport level request/response retry and time out are not sufficient mechanisms to achieve end-to-end agreement coordination for long-running activities.

This specification leverages WS-Coordination by extending it to support business activities. It does this by adding constraints to the protocols defined in WS-Coordination and by defining its own Coordination protocols.

The constraints that Business Activity puts on WS-Coordination protocols are described in Section 2. The Business Activity Coordination protocols are defined in Section 3.

46 Terms introduced in this specification are explained in the body of the specification and summarized in
47 the Glossary.

48 1.1 Model

49 Business Activity Coordination protocols provide the following flexibility:

- 50 • A business application may be partitioned into business activity scopes. A business activity scope is
51 a business task consisting of a general-purpose computation carried out as a bounded set of
52 operations on a collection of Web services that require a mutually agreed outcome. There can be
53 any number of hierarchical nesting levels. Nested scopes:
 - 54 – Allow a business application to select which child tasks are included in the overall outcome
55 processing. For example, a business application might solicit an estimate from a number of
56 suppliers and choose a quote or bid based on lowest-cost.
 - 57 – Allow a business application to catch an exception thrown by a child task, apply an exception
58 handler, and continue processing even if something goes wrong. When a child completes its
59 work, it may be associated with a compensation that is registered with the parent activity.
- 60 • A participant task within a business activity may specify that it is leaving a business activity. This
61 provides the ability to exit a business activity and allows business programs to delegate processing to
62 other scopes. In contrast to atomic transactions, the participant list is dynamic and a participant may
63 exit the protocol at any time without waiting for the outcome of the protocol.
- 64 • It allows a participant task within a business activity to specify its outcome directly without waiting for
65 solicitation. Such a feature is generally useful when a task fails so that the notification can be used
66 by a business activity exception handler to modify the goals and drive processing in a timely manner.
- 67 • It allows participants in a coordinated business activity to perform "tentative" operations as a normal
68 part of the activity. The result of such "tentative" operations may become visible before the activity is
69 complete and may require business logic to run in the event that the operation needs to be
70 compensated. Such a feature is critical when the joint work of a business activity requires many
71 operations performed by independent services over a long period of time.

72 1.2 Terminology

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

76 1.3 Namespace

77 The XML namespace URI that MUST be used by implementations of this specification is:

```
78 http://schemas.xmlsoap.org/ws/2004/10/wsba
```

79 1.3.1 Prefix Namespace

Prefix	Namespace
S	http://www.w3.org/2003/05/soap-envelope
wscor	http://schemas.xmlsoap.org/ws/2004/10/wscor
wsba	http://schemas.xmlsoap.org/ws/2004/10/wsba

80

81 If an action URI is used then the action URI MUST consist of the wsba namespace URI concatenated
82 with the "/" character and the element name. For example:

```
83 http://schemas.xmlsoap.org/ws/2004/10/wsba/Complete
```

84 1.4 Normative References

- 85 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
86 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 87 [SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
- 88 [URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
89 Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August
90 1998.
- 91 [XML-ns] W3C Recommendation, "Namespaces in XML," 14 January 1999.
- 92 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001.
- 93 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.
- 94 [WSCOOR] Web Services Coordination (WS-Coordination), "[http://docs.oasis-
95 open.org/wstx/ws-coordination-1.1-spec-wd-01.pdf](http://docs.oasis-open.org/wstx/ws-coordination-1.1-spec-wd-01.pdf)"
- 96 [WSDL] Web Services Description Language (WSDL) 1.1
97 "<http://www.w3.org/TR/2001/NOTE-wsdl-20010315>"
- 98 [WSADDR] Web Services Addressing (WS-Addressing), Microsoft, IBM, Sun, BEA Systems,
99 SAP, Sun, August 2004
- 100 [WSPOLICY] Web Services Policy Framework (WS-Policy), VeriSign, Microsoft, Sonic
101 Software, IBM, BEA Systems, SAP, September 2004
- 102 [WSPOLICYATTACH] Web Services Policy Attachment (WS-PolicyAttachment), VeriSign,
103 Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004
104

105 1.5 Non-Normative References

- 106
- 107 [BPEL] Web Services Business Process Execution Language, Microsoft, BEA and IBM.
- 108 [WSSec] OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message
109 Security 1.0 (WS-Security 2004)"
- 110 [WSSecPolicy] Web Services Security Policy Language (WS-SecurityPolicy), Microsoft,
111 VeriSign, IBM, RSA Security, December 2002
- 112 [WSSecConv] Web Services Secure Conversation Language (WS-SecureConversation),
113 OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, IBM, VeriSign, BEA
114 Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates,
115 February 2005
- 116 [WSTrust] Web Services Trust Language (WS-Trust), OpenNetwork, Layer7, Netegrity,
117 Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping
118 Identity, Westbridge, Computer Associates, February 2005
119

120 2 Using WS-Coordination

121 This section describes the Business Activity usage of WS-Coordination protocols.

122 2.1 Coordination Context

123 A business activity uses the WS-Coordination CoordinationContext with the CoordinationType set to one
124 of the following URIs:

125 `http://schemas.xmlsoap.org/ws/2004/10/wsba/AtomicOutcome`
126 `http://schemas.xmlsoap.org/ws/2004/10/wsba/MixedOutcome`

127 A coordination context may have an Expires attribute. This attribute specifies the earliest point in time at
128 which a long-running activity may be terminated solely due to its length of operation. A participant could
129 terminate its participation in the long running activity using the Exit protocol message.

130 A CoordinationContext can have additional elements for extensibility.

131 Due to the extensibility of WS-Coordination it is also possible to define a coordination protocol type that,
132 in addition to specifying the agreement protocol between a coordinator and a participant, also specifies
133 the behavior of the coordination logic. For example, it may specify that the coordinator will act in an all-or-
134 nothing manner to determine its outcome based on the outcomes communicated by its participants, or
135 that it will use a specific majority rule when determining its final outcome based on the outcomes of its
136 participants.

137 3 Coordination Types and Protocols

138 Business activities support two coordination types and two protocol types. Either protocol type may be
139 used with either coordination type.

140 The coordination types are atomic and mixed as identified by the following URIs:

141 `http://schemas.xmlsoap.org/ws/2004/10/wsba/AtomicOutcome`

142 `http://schemas.xmlsoap.org/ws/2004/10/wsba/MixedOutcome`

143 A coordinator for an AtomicOutcome coordination type must direct all participants to close or all
144 participants to compensate. A coordinator for a MixedOutcome coordination type may direct each
145 individual participant to close or compensate. All coordinators MUST implement the AtomicOutcome
146 coordination type. Any coordinator MAY implement the MixedOutcome coordination type.

147 The Coordination protocols for business activities are summarized below with names relative to the wsba
148 base name:

- 149 • **BusinessAgreementWithParticipantCompletion:** A participant registers for this protocol with its
150 coordinator, so that its coordinator can manage it. A participant must know when it has completed all
151 work for a business activity.
- 152 • **BusinessAgreementWithCoordinatorCompletion:** A participant registers for this protocol with its
153 coordinator, so that its coordinator can manage it. A participant relies on its coordinator to tell it when
154 it has received all requests to perform work within the business activity.

155 3.1 BusinessAgreementWithParticipantCompletion Protocol

156 The state diagram in Figure 1 specifies the behavior of the protocol between a coordinator and a
157 participant. The agreement coordination state reflects what each participant knows of their relationship at
158 a given point in time. As messages take time to be delivered, the views of the coordinator and a
159 participant may temporarily differ. Omitted are details such as resending of messages or the exchange of
160 error messages due to protocol error.

161 Participants register for this protocol using the following protocol identifier:

162 `http://schemas.xmlsoap.org/ws/2004/10/wsba/ParticipantCompletion`

163 The coordinator accepts:

164 **Completed**

165 Upon receipt of this notification, the coordinator knows that the participant has completed all
166 processing related to the protocol instance. For the next protocol message the coordinator
167 should send a Close or Compensate notification to indicate the final outcome of the protocol
168 instance.

169 **Fault**

170 Upon receipt of this notification, the coordinator knows that the participant has failed from the
171 active or compensating state. For the next protocol message the coordinator should send a
172 Faulted notification. This notification carries a QName defined in schema indicating the cause of
173 the fault.

174 **Compensated**

175 Upon receipt of this notification, the coordinator knows that the participant has recorded a
176 compensation request for a protocol.

177 **Closed**

178 Upon receipt of this notification, the coordinator knows that the participant has finalized
179 successfully.

180 **Canceled**

181 Upon receipt of this notification, the coordinator knows that the participant has finalized
182 successfully processing the Cancel notification.

183 **Exit**

184 Upon receipt of this notification, the coordinator knows that the participant will no longer
185 participate in the business activity. For the next protocol message the coordinator should send
186 an Exited notification.

187 The participant accepts:

188 **Close**

189 Upon receipt of this notification, the participant knows the protocol instance is to complete
190 successfully. For the next protocol message the participant should send a Closed notification to
191 end the protocol instance.

192 **Cancel**

193 Upon receipt of this notification, the participant knows that the work being done has to be
194 canceled. For the next protocol message the participant should send a Canceled notification to
195 end the protocol instance.

196 **Compensate**

197 Upon receipt of this notification, the participant knows that the work being done should be
198 compensated. For the next protocol message the participant should send a Compensated
199 notification to end the protocol instance.

200 **Faulted**

201 Upon receipt of this notification, the participant knows that the coordinator is aware of a fault and
202 no further actions are required of the participant.

203 **Exited**

204 Upon receipt of this notification, the participant knows that the coordinator is aware the participant
205 will no longer participate in the activity.

206 Both the coordinator and participant accept:

207 **GetStatus**

208 This message requests the current state of a coordinator or participant. In response the
209 coordinator or participant returns a Status message containing a QName indicating which row of
210 the state table [Appendix A: State Tables for the Agreement Protocols] the coordinator or
211 participant is currently in. GetStatus never provokes a state change.

212 **Status**

213 Upon receipt of this message the target service returns a QName defined in schema indicating
214 the current state of the coordinator or participant. For example, if a participant is in the closing
215 state as indicated by the state table, it would return wsba:Closing.

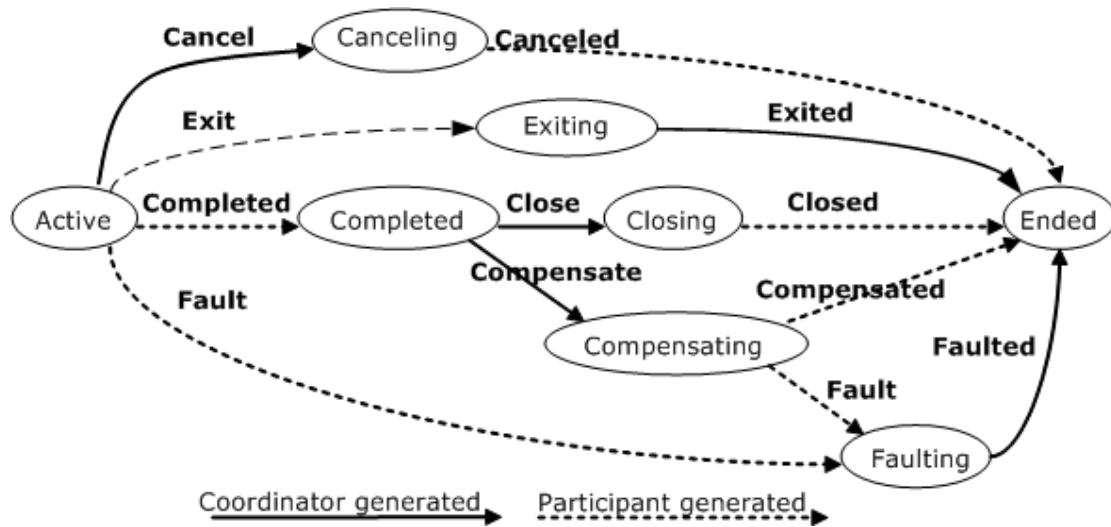


Figure 1: BusinessAgreementWithParticipantCompletion abstract state diagram

216

217

218 The coordinator can enter a condition in which it has sent a protocol message and it receives a protocol
 219 message from the participant that is consistent with the former state, not the current state. In this case, it
 220 is the responsibility of the coordinator to revert to the prior state, accept the notification from the
 221 participant, and continue the protocol from that point. If the participant detects this condition, it must
 222 discard the inconsistent protocol message from the coordinator.

223 A party should be prepared to receive duplicate notifications. If a duplicate message is received it should
 224 be treated as specified in the state tables described in this document.

225 **3.2 BusinessAgreementWithCoordinatorCompletion Protocol**

226 The BusinessAgreementWithCoordinatorCompletion protocol is the same as the
 227 BusinessAgreementWithParticipantCompletion protocol, except that a participant relies on its coordinator
 228 to tell it when it has received all requests to do work within the business activity.

229 Participants register for this protocol using the following protocol identifier:

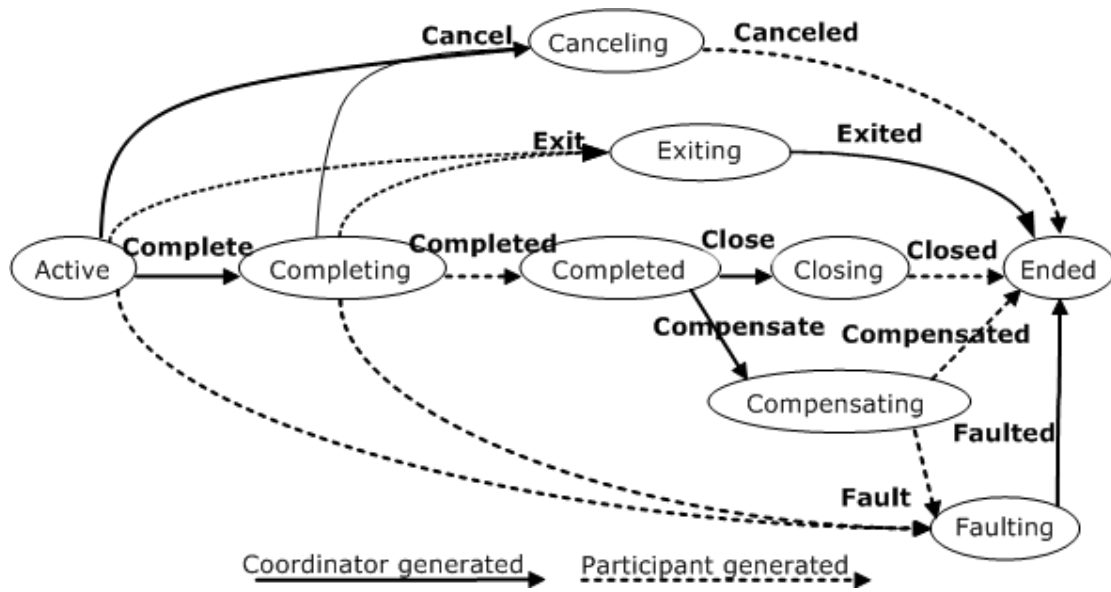
230 <http://schemas.xmlsoap.org/ws/2004/10/wsba/CoordinatorCompletion>

231 In addition to the notifications in Section 3.1, Business agreement with coordinator completion supports
 232 the following:

233 The participant accepts:

234 **Complete**

235 Upon receipt of this notification the participant knows that it will receive no new requests for work
 236 within the business activity. It should complete application processing and transmit the
 237 Completed notification.



238

239

Figure 2: BusinessAgreementWithCoordinatorCompletion abstract state diagram

240 4 WS-BA Policy Assertions

241 WS-Policy Framework [WS-Policy] and WS-Policy Attachment [WSPOLICYATTACH] collectively define a
242 framework, model and grammar for expressing the capabilities, requirements, and general characteristics
243 of entities in an XML Web services-based system. To enable a web service to describe business activity-
244 related capabilities and requirements of a service and its operations, this specification defines a pair of
245 Business Agreement policy assertions that leverage the WS-Policy framework

246 4.1 Assertion Models

247 The BA policy assertions are provided by a web service to qualify the business activity-related processing
248 of messages associated with the particular operation to which the assertions are scoped. The BA policy
249 assertions indicate:

250 whether the sender of an input message MAY, MUST or SHOULD NOT include an AtomicOutcome
251 coordination context flowed with the message. The coordination type of such a context MUST be the
252 following:

253 `http://schemas.xmlsoap.org/ws/2004/10/wsba/AtomicOutcome`

254 whether the sender of an input message MAY, MUST or SHOULD NOT include a MixedOutcome
255 coordination context flowed with the message. The coordination type of such a context MUST be the
256 following:

257 `http://schemas.xmlsoap.org/ws/2004/10/wsba/MixedOutcome`

258 4.2 Normative Outlines

259 The normative outlines for the BA policy assertions are:

```
260 <wsba:BAAtomicOutcomeAssertion [wsp:Optional="true"]? ... >  
261 ...  
262 </wsba:BAAtomicOutcomeAssertion>
```

263 The following describes additional, normative constraints on the outline listed above:

264 **/wsba:BAAtomicOutcomeAssertion**

265 A policy assertion that specifies that the sender of an input message MUST include a
266 coordination context for a business activity with AtomicOutcome coordination type flowed with the
267 message.

268 **/wsba:BAAtomicOutcomeAssertion/@wsp:Optional="true"**

269 Per WS-Policy [WS-Policy], this is compact notation for two policy alternatives, one with and one
270 without the assertion. Presence of both policy alternatives indicates that the behavior indicated by
271 the assertion is optional, such that an AtomicOutcome coordination context MAY be flowed inside
272 an input message. The absence of the assertion is interpreted to mean that an AtomicOutcome
273 coordination context SHOULD NOT be flowed inside an input message.

```
274 <wsba:BAMixedOutcomeAssertion [wsp:Optional="true"]? ... >  
275 ...  
276 </wsba:BAMixedOutcomeAssertion>
```

277 The following describes additional, normative constraints on the outline listed above:

278 **/wsba:BAMixedOutcomeAssertion**

279 A policy assertion that specifies that the sender of an input message MUST include a
280 coordination context for a business activity with MixedOutcome coordination type flowed with the
281 message.

282 **/wsba: BAMixedOutcomeAssertion/@wsp:Optional="true"**

283 Per WS-Policy [WS-Policy], this is compact notation for two policy alternatives, one with and one
284 without the assertion. Presence of both policy alternatives indicates that the behavior indicated by
285 the assertion is optional, such that a MixedOutcome coordination context MAY be flowed inside
286 an input message. The absence of the assertion is interpreted to mean that a MixedOutcome
287 coordination context SHOULD NOT be flowed inside an input message.

288 4.3 Assertion Attachment

289 Because the BA policy assertions indicate business activity-related behavior for a single operation, the
290 assertions have Operation Policy Subject.

291 WS-PolicyAttachment [WSPOLICYATTACH] defines two [WSDL] policy attachment points with Operation
292 Policy Subject:

- 293 • wsdl:portType/wsdl:operation – A policy expression containing a BA policy assertion MUST NOT be
294 attached to a wsdl:portType; the BA policy assertions specify a concrete behavior whereas the
295 wsdl:portType is an abstract construct.
- 296 • wsdl:binding/wsdl:operation – A policy expression containing a BA policy assertion SHOULD be
297 attached to a wsdl:binding.

298 4.4 Assertion Example

299 An example use of the BA policy assertion follows:

```
300 (01) <wsdl:definitions
301 (02)     targetNamespace="hotel.example.com"
302 (03)     xmlns:tns="hotel.example.com"
303 (04)     xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
304 (05)     xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
305 (06)     xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsba"
306 (07)     xmlns:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
307 (08)         wssecurity-utility-1.0.xsd" >
308 (09)     <wsp:Policy wsu:Id="BAAtomicPolicy" >
309 (10)         <wsba:BAAtomicOutcomeAssertion/>
310 (11)         <!-- omitted assertions -->
311 (12)     </wsp:Policy>
312 (13)     <!-- omitted elements -->
313 (14)     <wsdl:binding name="HotelBinding" type="tns:HotelPortType" >
314 (15)         <!-- omitted elements -->
315 (16)         <wsdl:operation name="ReserveRoom" >
316 (17)             <wsp:PolicyReference URI="#BAAtomicPolicy"
317 (18)                 wsdl:required="true" />
318 (19)             <!-- omitted elements -->
319 (20)         </wsdl:operation>
320 (21)     </wsdl:binding>
321 (22) </wsdl:definitions>
```

322 Lines (9-12) are a policy expression that includes a BA policy assertion (Line 10) to indicate that a
323 coordination context for a business activity with an AtomicOutcome, expressed in WS-Coordination [WS-
324 Coordination], format MUST be used.

325 Lines (16-20) are a WSDL [WSDL 1.1] binding. Line (17) indicates that the policy in Lines (9-12) applies
326 to this binding, specifically indicating that a coordination context for a business activity with an
327 AtomicOutcome MUST flow inside "ReserveRoom" messages.

328 5 Security Considerations

329 It is strongly RECOMMENDED that the communication between services be secured using the
330 mechanisms described in WS-Security [WSSec]. In order to properly secure messages, the body and all
331 relevant headers need to be included in the signature. Specifically, the <wscoor:CoordinationContext>
332 header needs to be signed with the body and other key message headers in order to "bind" the two
333 together.

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

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

- 341 • Attaching a nonce to each message and using it in a derived key function with the shared secret
- 342 • Using a derived key sequence and switch "generations"
- 343 • Closing and re-establishing a security context (not possible for delegated keys)
- 344 • Exchanging new secrets between the parties (not possible for delegated keys)

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

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

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

- 355 • **Message alteration** – Alteration is prevented by including signatures of the message information
356 using WS-Security [WSSec].
- 357 • **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security.
- 358 • **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing
359 secured policies – see WS-Policy [WSPOLICY] and WS-SecurityPolicy [WSSecPolicy]).
- 360 • **Authentication** – Authentication is established using the mechanisms described in WS-Security and
361 WS-Trust [WSTrust]. Each message is authenticated using the mechanisms described in WS-
362 Security [WSSec].
- 363 • **Accountability** – Accountability is a function of the type of and string of the key and algorithms being
364 used. In many cases, a strong symmetric key provides sufficient accountability. However, in some
365 environments, strong PKI signatures are required.
- 366 • **Availability** – Many services are subject to a variety of availability attacks. Replay is a common
367 attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other
368 attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope
369 of this specification. That said, care should be taken to ensure that minimal processing be performed
370 prior to any authenticating sequences.
- 371 • **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack,
372 mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in
373 WS-Security [WSSec]. Alternatively, and optionally, other technologies, such as sequencing, can
374 also be used to prevent replay of application messages.

375 **6 Interoperability Considerations**

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

378 7 Glossary

379 **Cancel**

380 Back out of a business activity.

381 **Close**

382 Terminate a business activity with a favorable outcome.

383 **Compensate**

384 A message to a Completed participant from a coordinator to execute its compensation. This
385 message is part of both the BusinessAgreementWithParticipantCompletion and
386 BusinessAgreementWithCoordinatorCompletion protocols.

387 **Complete**

388 A message to a participant from a coordinator telling it that it has been given all of the work for
389 that business activity. This message is part of the
390 BusinessAgreementWithCoordinatorCompletion protocol.

391 **Completed**

392 A message from a participant telling a coordinator that the participant has successfully executed
393 everything asked of it and needs to continue participating in the protocol. This message is part of
394 both the BusinessAgreementWithParticipantCompletion and
395 BusinessAgreementWithCoordinatorCompletion protocols.

396 **Exit**

397 A message from a participant telling a coordinator that the participant does not need to continue
398 participating in the protocol. This message is part of both the
399 BusinessAgreementWithParticipantCompletion and
400 BusinessAgreementWithCoordinatorCompletion protocols.

401 **Fault**

402 A message from a participant telling a coordinator that the participant could not execute
403 successfully.

404 **BusinessAgreementWithParticipantCompletion protocol**

405 A business activity coordination protocol that supports long-lived business processes and allows
406 business logic to handle business logic exceptions. A participant in this protocol must know when
407 it has completed with its tasks in a business activity.

408 **BusinessAgreementWithCoordinatorCompletion protocol**

409 A business activity coordination protocol that supports long-lived business processes and allows
410 business logic to handle business logic exceptions. A participant in this protocol relies on its
411 coordinator to tell it when it has received all requests to do work within a business activity.

412 **Scope**

413 A business activity instance. A scope integrates coordinator and application logic. A web
414 services application can be partitioned into a hierarchy of scopes, where the application
415 understands the relationship between the parent scope and its child scopes.

416 **Appendix A. Acknowledgements**

417 The following individuals have participated in the creation of this specification and are gratefully
418 acknowledged:

419 **Participants:**

420 [Participant Name, Affiliation | Individual Member]

421 [Participant Name, Affiliation | Individual Member]

422

423

Appendix B. Revision History

424

Revision	Date	Editor	Changes Made
01	11/22/2005	Tom Freund	Initial Working Draft

425

426 **Appendix C. State Tables for the Agreement Protocols**

427 The following state tables show state transitions that occur in the receiver when a protocol message is
 428 received or in the sender when a protocol message is sent. Each table uses the following convention:



429
 430 where the next state refers to the next agreement protocol state. An Action of Invalid State means the
 431 sent or received protocol message cannot occur in the current state.

432 The following rules need to be applied when reading the state tables in this document:

- 433 • For the period of time that a protocol message is in transit the sender and recipient states will be
 434 different.
- 435 The sender of a protocol message transitions to the "next state" when the message is first sent.
- 436 The recipient of a protocol message transitions to the "next state" when the message is first received.
- 437 • As described earlier in this document, if the coordinator receives a protocol message from the
 438 participant that is consistent with the former state of the coordinator then the coordinator reverts to its
 439 prior state, accepts the notification from the participant, and continues the protocol from that point.

440 The GetStatus and Status protocol messages are not included in the tables as these never result in a
 441 change of state.

442 **C.1. Participant view of**
 443 **BusinessAgreementWithParticipantCompletion**

BusinessAgreementWithParticipantCompletion protocol					
Participant view of state	Protocol messages received by Participant				
	Cancel	Close	Compensate	Faulted	Exited
Active	Canceling	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	<i>Ignore</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completed	<i>Resend Completed</i> Completed	Closing	Compensating	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Ignore</i> Closing	<i>Ignore</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing
Compensating	<i>Ignore</i> Compensating	<i>Invalid State</i> Compensating	<i>Ignore</i> Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating
Faulting (Active, Completed)	<i>Resend Fault</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Ended	<i>Invalid State</i> Faulting
Faulting (Compensating)	<i>Ignore</i> Faulting	<i>Invalid State</i> Faulting	<i>Resend Fault</i> Faulting	Ended	<i>Invalid State</i> Faulting
Exiting	<i>Resend Exit</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	Ended
Ended	<i>Send Canceled</i> Ended	<i>Send Closed</i> Ended	<i>Send Compensated</i> Ended	<i>Ignore</i> Ended	<i>Ignore</i> Ended

444

BusinessAgreementWithParticipantCompletion						
Participant view of state	Protocol messages sent by Participant					
	Exit	Completed	Fault	Canceled	Closed	Compensated
Active	Exiting	Completed	Faulting-Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	Ended	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completed	<i>Invalid State</i> Completed	Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	Ended	<i>Invalid State</i> Closing
Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Faulting-Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Ended
Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting
Exiting	Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting
Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	Ended	Ended	Ended

445

C.2. Coordinator view of BusinessAgreementWithParticipantCompletion

446

447

BusinessAgreementWithParticipantCompletion						
Coordinator view of state	Protocol messages received by Coordinator					
	Exit	Completed	Fault	Canceled	Closed	Compensated
Active	Exiting	Completed	Faulting-Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	Exiting	Completed	Faulting-Active	Ended	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completed	<i>Invalid State</i> Completed	<i>Ignore</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Invalid State</i> Closing	<i>Resend Close</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	Ended	<i>Invalid State</i> Closing
Compensating	<i>Invalid State</i> Compensating	<i>Resend Compensate</i> Compensating	Faulting-Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Ended
Faulting (Compensating)	<i>Invalid State</i> Faulting	<i>Ignore</i> Faulting	<i>Ignore</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting
Faulting (Active)	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Ignore</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting
Exiting	<i>Ignore</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting
Ended	<i>Resend Exited</i> Ended	<i>Ignore</i> Ended	<i>Resend Faulted</i> Ended	<i>Ignore</i> Ended	<i>Ignore</i> Ended	<i>Ignore</i> Ended

448

BusinessAgreementWithParticipantCompletion protocol					
Coordinator view of state	Protocol messages sent by Coordinator				
	Cancel	Close	Compensate	Faulted	Exited
Active	Canceling-Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completed	<i>Invalid State</i> Completed	Closing	Compensating	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Invalid State</i> Closing	Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing
Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating
Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Ended	<i>Invalid State</i> Faulting
Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	Ended
Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	Ended	Ended

449

450

C.3. Participant view of BusinessAgreementWithCoordinatorCompletion

451

BusinessAgreementWithCoordinatorCompletion protocol						
Participant view of state	Protocol messages received by Participant					
	Cancel	Complete	Close	Compensate	Faulted	Exited
Active	Canceling	Completing	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	<i>Ignore</i> Canceling	<i>Ignore</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completing	Canceling	<i>Ignore</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing
Completed	<i>Resend Completed</i> Completed	<i>Resend Completed</i> Completed	Closing	Compensating	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Ignore</i> Closing	<i>Ignore</i> Closing	<i>Ignore</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing
Compensating	<i>Ignore</i> Compensating	<i>Ignore</i> Compensating	<i>Invalid State</i> Compensating	<i>Ignore</i> Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating
Faulting (Active, Completed)	<i>Resend Fault</i> Faulting	<i>Resend Fault</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Ended	<i>Invalid State</i> Faulting
Faulting (Compensating)	<i>Ignore</i> Faulting	<i>Ignore</i> Faulting	<i>Invalid State</i> Faulting	<i>Resend Fault</i> Faulting	Ended	<i>Invalid State</i> Faulting
Exiting	<i>Resend Exit</i> Exiting	<i>Resend Exit</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	Ended
Ended	<i>Send Canceled</i> Ended	<i>Ignore</i> Ended	<i>Send Closed</i> Ended	<i>Send Compensated</i> Ended	<i>Ignore</i> Ended	<i>Ignore</i> Ended

452

BusinessAgreementWithCoordinatorCompletion						
Participant view of state	Protocol messages sent by Participant					
	Exit	Completed	Fault	Canceled	Closed	Compensated
Active	Exiting	<i>Invalid State</i> Active	Faulting-Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	Ended	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completing	Exiting	Completed	Faulting-Active	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing
Completed	<i>Invalid State</i> Completed	Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	Ended	<i>Invalid State</i> Closing
Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Faulting-Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Ended
Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting
Exiting	Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting
Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	Ended	Ended	Ended

453

C.4. Coordinator view of BusinessAgreementWithCoordinatorCompletion

454

455

BusinessAgreementWithCoordinatorCompletion						
Coordinator view of state	Protocol messages received by Coordinator					
	Exit	Completed	Fault	Canceled	Closed	Compensated
Active	Exiting	<i>Invalid State</i> Active	Faulting-Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling-Active	Exiting	<i>Invalid State</i> Canceling	Faulting-Active	Ended	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Canceling-Completing	Exiting	Completed	Faulting-Active	Ended	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completing	Exiting	Completed	Faulting-Active	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing
Completed	<i>Invalid State</i> Completed	Ignore Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Invalid State</i> Closing	Resend Close Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	Ended	<i>Invalid State</i> Closing
Compensating	<i>Invalid State</i> Compensating	Resend Compensate Compensating	Faulting-Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Ended
Faulting (Compensating)	<i>Invalid State</i> Faulting	Ignore Faulting	Ignore Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting
Faulting (Active, Completing)	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Ignore Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting
Exiting	Ignore Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting
Ended	Resend Exited Ended	Ignore Ended	Resend Faulted Ended	Ignore Ended	Ignore Ended	Ignore Ended

456

BusinessAgreementWithCoordinatorCompletion protocol						
Coordinator view of state	Protocol messages Sent by Coordinator					
	Cancel	Complete	Close	Compensate	Faulted	Exited
Active	Canceling-Active	Completing	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active	<i>Invalid State</i> Active
Canceling	Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling	<i>Invalid State</i> Canceling
Completing	Canceling-Completing	Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing	<i>Invalid State</i> Completing
Completed	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed	Closing	Compensating	<i>Invalid State</i> Completed	<i>Invalid State</i> Completed
Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing	<i>Invalid State</i> Closing
Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating	Compensating	<i>Invalid State</i> Compensating	<i>Invalid State</i> Compensating
Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	<i>Invalid State</i> Faulting	Ended	<i>Invalid State</i> Faulting
Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	<i>Invalid State</i> Exiting	Ended
Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	<i>Invalid State</i> Ended	Ended	Ended

457

458