



# Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2

## OASIS Standard

2 February 2009

### Specification URIs:

#### This Version:

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

#### Previous Version:

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

#### Latest Approved Version:

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

### Technical Committee:

OASIS Web Services Transaction (WS-TX) TC

### Chair(s):

Eric Newcomer, Iona  
Ian Robinson, IBM

### Editor(s):

Mark Little, JBoss Inc. <mark.little@jboss.com>  
Andrew Wilkinson, IBM <awilkinson@uk.ibm.com>

### Declared XML Namespaces:

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

### Abstract:

The WS-AtomicTransaction specification provides the definition of the Atomic Transaction coordination type that is to be used with the extensible coordination framework described in WS-Coordination. This specification defines three specific agreement coordination protocols for the Atomic Transaction coordination type: completion, volatile two-phase commit, and durable two-phase commit. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of short-lived distributed activities that have the all-or-nothing property.

### Status:

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

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the

“Send A Comment” button on the Technical Committee’s web page at [www.oasis-open.org/committees/ws-tx](http://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](http://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](http://www.oasis-open.org/committees/ws-tx) .

---

# Notices

Copyright © OASIS Open 2008. All Rights Reserved.

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

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

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

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

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

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

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

---

# Table of contents

1	Introduction .....	5
1.1	Composable Architecture .....	5
1.2	Terminology .....	5
1.3	Namespace .....	6
1.3.1	Prefix Namespace.....	6
1.4	XSD and WSDL Files.....	6
1.5	Protocol Elements .....	7
1.6	Conformance.....	7
1.7	Normative References .....	7
2	Atomic Transaction Context .....	9
3	Atomic Transaction Protocols.....	10
3.1	Preconditions .....	10
3.2	Completion Protocol.....	10
3.3	Two-Phase Commit Protocol.....	11
3.3.1	Volatile Two-Phase Commit Protocol.....	11
3.3.2	Durable Two-Phase Commit Protocol.....	12
3.3.3	2PC Diagram and Notifications .....	12
4	Policy Assertion.....	14
4.1	Assertion Model .....	14
4.2	Normative Outline .....	14
4.3	Assertion Attachment .....	14
4.4	Assertion Example .....	14
5	Transaction Faults.....	16
5.1	Inconsistent Internal State.....	17
5.2	Unknown Transaction .....	17
6	Security Model .....	18
7	Security Considerations .....	20
8	Use of WS-Addressing Headers .....	22
9	State Tables.....	23
9.1	Completion Protocol.....	23
9.2	2PC Protocol.....	24
A.	Acknowledgements.....	28

---

# 1 Introduction

The current set of Web service specifications [WSDL][SOAP11][SOAP12] 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.

WS-Coordination [WSCOOR] defines an extensible framework for defining coordination types. This specification provides the definition of an Atomic Transaction coordination type used to coordinate activities having an "all or nothing" property. Atomic transactions commonly require a high level of trust between participants and are short in duration. WS-AtomicTransaction defines protocols that enable existing transaction processing systems to wrap their proprietary protocols and interoperate across different hardware and software vendors.

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

The reader is familiar with existing standards for two-phase commit protocols and with commercially available implementations of such protocols. Therefore this section includes only those details that are essential to understanding the protocols described.

The reader is familiar with WS-Coordination [WSCOOR] which defines the framework for the Atomic Transaction coordination protocols.

The reader is familiar with WS-Addressing [WSADDR] and WS-Policy [WSPOLICY].

Atomic transactions have an all-or-nothing property. The actions taken by a transaction participant prior to commit are only tentative; typically they are neither persistent nor made visible outside the transaction. When an application finishes working on a transaction, it requests the coordinator to determine the outcome for the transaction. The coordinator determines if there were any processing failures by asking the participants to vote. If the participants all vote that they were able to execute successfully, the coordinator commits all actions taken. If a participant votes that it needs to abort or a participant does not respond at all, the coordinator aborts all actions taken. Commit directs the participants to make the tentative actions final so they may, for example, be made persistent and be made visible outside the transaction. Abort directs the participants to make the tentative actions appear as if they never happened. Atomic transactions have proven to be extremely valuable for many applications. They provide consistent failure and recovery semantics, so the applications no longer need to deal with the mechanics of determining a mutually agreed outcome decision or to figure out how to recover from a large number of possible inconsistent states.

This specification defines protocols that govern the outcome of Atomic Transactions. It is expected that existing transaction processing systems will use WS-AtomicTransaction to wrap their proprietary mechanisms and interoperate across different vendor implementations.

## 1.1 Composable Architecture

By using the XML [XML], SOAP [SOAP11][SOAP12] and WSDL [WSDL] extensibility model, SOAP-based and WSDL-based specifications are designed to work together to define a rich Web services environment. As such, WS-AtomicTransaction by itself does not define all features required for a complete solution. WS-AtomicTransaction is a building block used with other specifications of Web services (e.g., WS-Coordination [WSCOOR], WS-Security [WSSec]) and application-specific protocols that are able to accommodate a wide variety of coordination protocols related to the coordination actions of distributed applications.

## 1.2 Terminology

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

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

- 47 • The syntax appears as an XML instance, but the values indicate the data types instead of values.
- 48 • Element names ending in "..." (such as <element.../> or <element...>) indicate that
- 49 elements/attributes irrelevant to the context are being omitted.
- 50 • Attributed names ending in "..." (such as name=...) indicate that the values are specified below.
- 51 • Grammar in bold has not been introduced earlier in the document, or is of particular interest in an
- 52 example.
- 53 • <!-- description --> is a placeholder for elements from some "other" namespace (like ##other in
- 54 XSD).
- 55 • Characters are appended to elements, attributes, and <!-- descriptions --> as follows: "?" (0 or 1),
- 56 "\*" (0 or more), "+" (1 or more). The characters "[" and "]" are used to indicate that contained
- 57 items are to be treated as a group with respect to the "?", "\*", or "+" characters.
- 58 • The XML namespace prefixes (defined below) are used to indicate the namespace of the element
- 59 being defined.
- 60 • Examples starting with <?xml contain enough information to conform to this specification; others
- 61 examples are fragments and require additional information to be specified in order to conform.

## 62 1.3 Namespace

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

64 `http://docs.oasis-open.org/ws-tx/wsat/2006/06`

65 This MUST also be used as the CoordinationContext type for Atomic Transactions.

### 66 1.3.1 Prefix Namespace

67 The following namespaces are used in this document:

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

## 68 1.4 XSD and WSDL Files

69 Dereferencing the XML namespace defined in section 1.3 will produce the Resource Directory  
 70 Description Language (RDDL) [RDDL] document that describes this namespace, including the XML  
 71 schema [XML-Schema1] [XML-Schema2] and WSDL [WSDL] declarations associated with this  
 72 specification.

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

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

## 79 1.5 Protocol Elements

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

## 84 1.6 Conformance

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

89

## 90 1.7 Normative References

- 91 **[RDDL]** Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language  
92 (RDDL) 2.0", <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>,  
93 January 2004
- 94 **[RFC2119]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels",  
95 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC2119, March 1997
- 96 **[SOAP11]** W3C Note, "SOAP: Simple Object Access Protocol 1.1",  
97 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>, 08 May 2000
- 98 **[SOAP12]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework  
99 (Second Edition)", <http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>,  
100 April 2007.
- 101 **[WSADDR]** Web Services Addressing (WS-Addressing) 1.0,  
102 <http://www.w3.org/2005/08/addressing>, W3C Recommendation, May 2006
- 103 **[WSCOOR]** OASIS Standard, Web Services Coordination (WS-Coordination) 1.2,  
104 <http://docs.oasis-open.org/wscor/wstx-wscor-1.2-spec-os.doc>,  
105 February 2009
- 106 **[WSDL]** Web Services Description Language (WSDL) 1.1,  
107 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 108 **[WSPOLICY]** W3C Recommendation, Web Services Policy 1.5 – Framework (WS-Policy),  
109 <http://www.w3.org/TR/2007/REC-ws-policy-20070904/>, September 2007.
- 110 **[WSPOLICYATTACH]** W3C Recommendation, Web Services Policy 1.5 – Attachment (WS-  
111 PolicyAttachment), [http://www.w3.org/TR/2007/REC-ws-policy-attach-  
112 20070904/](http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/), September 2007.
- 113 **[WSSec]** OASIS Standard, March 2004, Web Services Security: SOAP Message  
114 Security 1.0 (WS-Security 2004), [http://docs.oasis-  
115 open.org/wss/2004/01/oasis-200401-wss-soap-message-security-  
116 1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf).  
117 OASIS Standard, February 2006, Web Services Security: SOAP Message  
118 Security 1.1 (WS-Security 2004), [http://www.oasis-  
119 open.org/committees/download.php/16790/wss-v1.1-spec-os-  
120 SOAPMessageSecurity.pdf](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf).

121 **[WSConv]** OASIS Standard, WS-SecureConversation 1.4, February 2009  
122 [http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-](http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc)  
123 [secureconversation-1.4-spec-os.doc](http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc)

124 **[WSPolicy]** OASIS Standard, WS-SecurityPolicy 1.3, February 2009.  
125 [http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.doc)  
126 [securitypolicy-1.3-spec-os.doc](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.doc).

127 **[WSTrust]** OASIS Standard, WS-Trust 1.4, February 2009. [http://docs.oasis-](http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc)  
128 [open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc](http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc)

129 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth  
130 Edition)", <http://www.w3.org/TR/2006/REC-xml-20060816>, 16 August 2006

131 **[XML-ns]** W3C Recommendation, "Namespaces in XML (Second Edition)",  
132 <http://www.w3.org/TR/2006/REC-xml-names-20060816>, 16 August 2006

133 **[XML-Schema1]** W3C Recommendation, "XML Schema Part 1: Structures Second Edition",  
134 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>, 28 October 2004

135 **[XML-Schema2]** W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition",  
136 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>, 28 October 2004



137

---

## 2 Atomic Transaction Context

138 WS-AtomicTransaction builds on WS-Coordination [WSCOOR], which defines an Activation service, a  
139 Registration service, and a CoordinationContext type. Example message flows and a complete  
140 description of creating and registering for coordinated activities is found in WS-Coordination [WSCOOR].

141 The Atomic Transaction coordination context is a CoordinationContext type with the coordination type  
142 defined in this section. Atomic Transaction application messages that propagate a coordination context  
143 MUST use an Atomic Transaction coordination context. If these application messages use a SOAP  
144 binding, the Atomic Transaction coordination context MUST flow as a SOAP header in the message.

145 WS-AtomicTransaction adds the following semantics to the CreateCoordinationContext operation on the  
146 Activation service:

147 If the request includes the CurrentContext element, the target coordinator is interposed as a subordinate  
148 to the coordinator stipulated inside the CurrentContext element.

149 If the request does not include a CurrentContext element, the target coordinator creates a new  
150 transaction and acts as the root.

151 A coordination context MAY have an Expires element. This element specifies the period, measured from  
152 the point in time at which the context was first created or received, after which a transaction MAY be  
153 terminated solely due to its length of operation. From that point forward, the coordinator MAY elect to  
154 unilaterally roll back the transaction, so long as it has not made a commit decision. Similarly a 2PC  
155 participant MAY elect to abort its work in the transaction so long as it has not already decided to prepare.

156 The Atomic Transaction protocol is identified by the following coordination type:

157

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

---

## 3 Atomic Transaction Protocols

158

159 This specification defines the following protocols for Atomic Transactions:

160 **Completion:** The completion protocol initiates commit processing. Based on each protocol's registered  
161 participants, the coordinator begins with Volatile 2PC and then proceeds through Durable 2PC. The final  
162 result is signaled to the initiator.

163 **Two-Phase Commit (2PC):** The 2PC protocol coordinates registered participants to reach a commit  
164 or abort decision, and ensures that all participants are informed of the final result. The 2PC protocol has  
165 two variants:

166 **Volatile 2PC:** Participants managing volatile resources such as a cache register for this protocol.

167 **Durable 2PC:** Participants managing durable resources such as a database register for this protocol.

168 A participant MAY register for more than one of these protocols.

### 3.1 Preconditions

169

170 The correct operation of the protocols requires that a number of preconditions must be established prior  
171 to the processing:

172 The source SHOULD have knowledge of the destination's policies, if any, and the source SHOULD be  
173 capable of formulating messages that adhere to this policy.

174 If a secure exchange of messages is required, then the source and destination MUST have appropriate  
175 security credentials (such as transport-level security credentials or security tokens) in order to protect the  
176 messages.

### 3.2 Completion Protocol

177

178 The Completion protocol is used by an application to tell the coordinator to either try to commit or abort an  
179 Atomic Transaction. After the transaction has completed, a status is returned to the application.

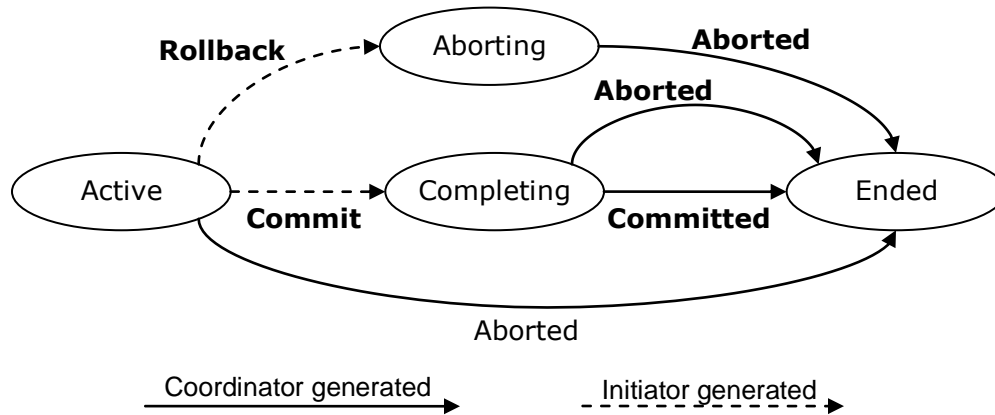
180 An initiator that registers for this protocol MUST use the following protocol identifier:

181

```
http://docs.oasis-open.org/ws-tx/wsat/2006/06/Completion
```

182 A Completion protocol coordinator MUST be the root coordinator of an Atomic Transaction. The  
183 Registration service for a subordinate coordinator MUST respond to an attempt to register for this  
184 coordination protocol with the WS-Coordination fault Cannot Register Participant.

185 The diagram below illustrates the protocol abstractly. Refer to section 9 State Tables for a detailed  
186 description of this protocol.



187  
 188 The coordinator accepts:  
 189 Commit  
 190 Upon receipt of this notification, the coordinator knows that the initiator has completed application  
 191 processing. A coordinator that is Active SHOULD attempt to commit the transaction.

192 Rollback  
 193 Upon receipt of this notification, the coordinator knows that the initiator has terminated application  
 194 processing. A coordinator that is Active MUST abort the transaction.

195 The initiator accepts:  
 196 Committed  
 197 Upon receipt of this notification, the initiator knows that the coordinator reached a decision to  
 198 commit.

199 Aborted  
 200 Upon receipt of this notification, the initiator knows that the coordinator reached a decision to  
 201 abort.

202 A coordination service that supports an Activation service MUST support the Completion protocol.

### 203 3.3 Two-Phase Commit Protocol

204 The Two-Phase Commit (2PC) protocol is a Coordination protocol that defines how multiple participants  
 205 reach agreement on the outcome of an Atomic Transaction. The 2PC protocol has two variants: Volatile  
 206 2PC and Durable 2PC.

#### 207 3.3.1 Volatile Two-Phase Commit Protocol

208 Upon receiving a Commit notification in the Completion protocol, the root coordinator begins the prepare  
 209 phase of all participants registered for the Volatile 2PC protocol. All participants registered for this  
 210 protocol MUST respond before a Prepare is issued to a participant registered for Durable 2PC. Further  
 211 participants MAY register with the coordinator until the coordinator issues a Prepare to any durable  
 212 participant. Once this has happened the Registration Service for the coordinator MUST respond to any  
 213 further Register requests with a Cannot Register Participant fault message. A volatile recipient is not  
 214 guaranteed to receive a notification of the transaction's outcome.

215 Participants that register for this protocol MUST use the following protocol identifier:

216 <http://docs.oasis-open.org/ws-tx/wsac/2006/06/Volatile2PC>

217 **3.3.2 Durable Two-Phase Commit Protocol**

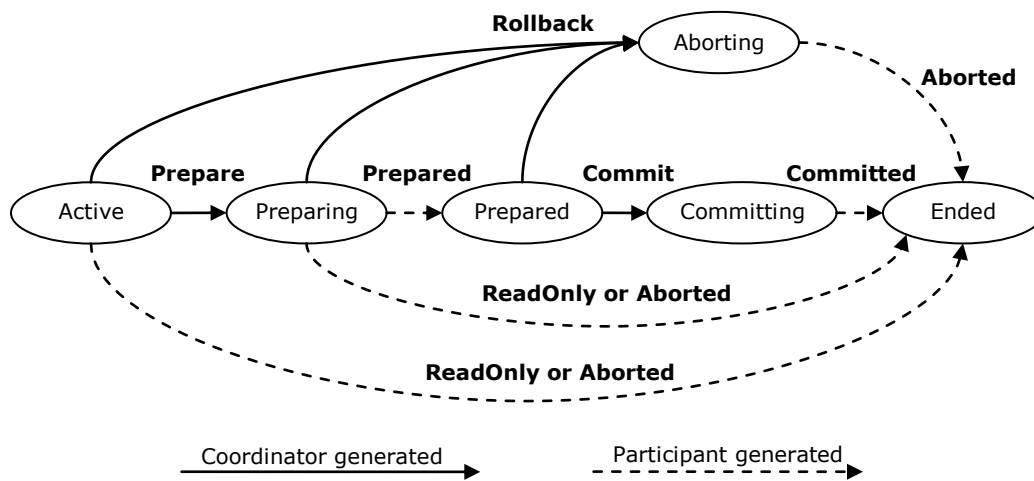
218 Upon successfully completing the prepare phase for Volatile 2PC participants, the root coordinator begins  
219 the prepare phase for Durable 2PC participants. All participants registered for this protocol **MUST**  
220 respond Prepared or ReadOnly before a Commit notification is issued to a participant registered for either  
221 protocol.

222 Participants that register for this protocol **MUST** use the following protocol identifier:

223 `http://docs.oasis-open.org/ws-tx/wsac/2006/06/Durable2PC`

224 **3.3.3 2PC Diagram and Notifications**

225 The diagram below illustrates the protocol abstractly. Refer to section 9 State Tables for a detailed  
226 description of this protocol.



227  
228 The participant accepts:

229 **Prepare**

230 Upon receipt of this notification, the participant knows to enter phase one and vote on the  
231 outcome of the transaction. A participant that is Active **MUST** respond by sending Aborted,  
232 Prepared, or ReadOnly notification as its vote. If the participant does not know of the transaction,  
233 it **MUST** send an Aborted notification. If the participant knows that it has already voted, it **MUST**  
234 resend the same vote.

235 **Rollback**

236 Upon receipt of this notification, the participant knows to abort and forget the transaction. A  
237 participant that is not Committing **MUST** respond by sending an Aborted notification and  
238 **SHOULD** then forget all knowledge of this transaction. If the participant does not know of the  
239 transaction, it **MUST** send an Aborted notification to the coordinator.

240 **Commit**

241 Upon receipt of this notification, the participant knows to commit the transaction. This notification  
242 **MUST** only be sent after phase one and if the participant voted to commit. If the participant does  
243 not know of the transaction, it **MUST** send a Committed notification to the coordinator.

244 The coordinator accepts:

245 **Prepared**

246            Upon receipt of this notification, the coordinator knows the participant is Prepared and votes to  
247            commit the transaction.

248    ReadOnly

249            Upon receipt of this notification, the coordinator knows the participant votes to commit the  
250            transaction, and has forgotten the transaction. The participant does not wish to participate in  
251            phase two.

252    Aborted

253            Upon receipt of this notification, the coordinator knows the participant has aborted and forgotten  
254            the transaction.

255    Committed

256            Upon receipt of this notification, the coordinator knows the participant has committed and  
257            forgotten the transaction.

258    Conforming implementations MUST implement the 2PC protocol.

---

## 259 4 Policy Assertion

260 WS-Policy Framework [WSPOLICY] and WS-Policy Attachment [WSPOLICYATTACH] collectively define  
261 a framework, model and grammar for expressing the capabilities, requirements, and general  
262 characteristics of entities in an XML Web services-based system. To enable a Web service to describe  
263 transactional capabilities and requirements of a service and its operations, this specification defines an  
264 Atomic Transaction policy assertion that leverages the WS-Policy [WSPOLICY] framework.

### 265 4.1 Assertion Model

266 The Atomic Transaction policy assertion is provided by a Web service to qualify the transactional  
267 processing of messages associated with the particular operation to which the assertion is scoped. It  
268 indicates whether a requester MAY or MUST include an Atomic Transaction coordination context flowed  
269 with the message.

### 270 4.2 Normative Outline

271 The normative outline for the Atomic Transaction policy assertion is:

```
272 <wsat:ATAssertion [wsp:Optional="true"]? ... >  
273 ...  
274 </wsat:ATAssertion>
```

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

276 /wsat:ATAssertion

277 A policy assertion that specifies that an Atomic Transaction coordination context MUST be flowed inside a  
278 requester's message. From the perspective of the requester, the target service that processes the  
279 transaction MUST behave as if it had participated in the transaction. For application messages that use a  
280 SOAP binding, the Atomic Transaction coordination context MUST flow as a SOAP header in the  
281 message.

282 /wsat:ATAssertion/@wsp:Optional="true"

283 Per WS-Policy [WSPOLICY], this is compact notation for two policy alternatives, one with and one without  
284 the assertion.

285 The Atomic Transaction policy assertion MUST NOT include a wsp:Ignorable attribute with a value of  
286 "true".

### 287 4.3 Assertion Attachment

288 Because the Atomic Transaction policy assertion indicates Atomic Transaction behavior for a single  
289 operation, the assertion has an Operation Policy Subject [WSPOLICYATTACH].

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

292 wsdl:portType/wsdl:operation – A policy expression containing the Atomic Transaction policy assertion  
293 MUST NOT be attached to a wsdl:portType; the Atomic Transaction policy assertion specifies a concrete  
294 behavior whereas the wsdl:portType is an abstract construct.

295 wsdl:binding/wsdl:operation – A policy expression containing the Atomic Transaction policy assertion  
296 SHOULD be attached to a wsdl:binding.

### 297 4.4 Assertion Example

298 An example use of the Atomic Transaction policy assertion follows:

```

299 (01) <wsdl:definitions
300 (02)     targetNamespace="bank.example.com"
301 (03)     xmlns:tns="bank.example.com"
302 (04)     xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
303 (05)     xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
304 (06)     xmlns:wsat="http://docs.oasis-open.org/ws-tx/wsat/2006/06"
305 (07)     xmlns:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
306 wssecurity-utility-1.0.xsd" >
307 (08)     <wsp:Policy wsu:Id="TransactedPolicy" >
308 (09)         <wsat:ATAssertion wsp:optional="true" />
309 (10)         <!-- omitted assertions -->
310 (11)     </wsp:Policy>
311 (12)     <!-- omitted elements -->
312 (13)     <wsdl:binding name="BankBinding" type="tns:BankPortType" >
313 (14)         <!-- omitted elements -->
314 (15)         <wsdl:operation name="TransferFunds" >
315 (16)             <wsp:PolicyReference URI="#TransactedPolicy" wSDL:required="true"
316 />
317 (17)             <!-- omitted elements -->
318 (18)         </wsdl:operation>
319 (19)     </wsdl:binding>
320 (20) </wsdl:definitions>

```

322 Lines 8-11 are a policy expression that includes an Atomic Transaction policy assertion (line 9) to indicate  
323 that an Atomic Transaction in WS-Coordination [WSCOOR] format MAY be used.

324 Lines 13-19 are a WSDL [WSDL] binding. Line 16 indicates that the policy in lines 8-11 applies to this  
325 binding, specifically indicating that an Atomic Transaction MAY flow inside messages.

---

## 326 5 Transaction Faults

327 Atomic Transaction faults MUST include, as the [action] property, the following fault action URI:

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

329 The protocol faults defined in this section are generated if the condition stated in the preamble is met.  
330 These faults are targeted at a destination endpoint according to the protocol fault handling rules defined  
331 for that protocol.

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

333 [Code] The fault code.

334 [Subcode] The fault subcode.

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

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

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

339

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

340

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

```
342 <S12:Envelope>
343 <S12:Header>
344   <wsa:Action>
345     http://docs.oasis-open.org/ws-tx/wsat/2006/06/fault
346   </wsa:Action>
347   <!-- Headers elided for clarity. -->
348 </S12:Header>
349 <S12:Body>
350 <S12:Fault>
351 <S12:Code>
352   <S12:Value>[Code]</S12:Value>
353   <S12:Subcode>
354     <S12:Value>[Subcode]</S12:Value>
355   </S12:Subcode>
356 </S12:Code>
357 <S12:Reason>
358   <S12:Text xml:lang="en">[Reason]</S12:Text>
359 </S12:Reason>
360 <S12:Detail>
361   [Detail]
362   ...
363 </S12:Detail>
364 </S12:Fault>
365 </S12:Body>
366 </S12:Envelope>
```

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

```
368 <S11:Envelope>
369 <S11:Body>
370 <S11:Fault>
371 <faultcode> [Subcode]</faultcode>
```



```
372     <faultstring xml:lang="en">[Reason]</faultstring>
373     </S11:Fault>
374     </S11:Body>
375 </S11:Envelope>
```

## 376 **5.1 Inconsistent Internal State**

377 This fault is sent by a participant or coordinator to indicate that a protocol violation has been detected  
378 after it is no longer possible to change the outcome of the transaction. This is indicative of a global  
379 consistency failure and is an unrecoverable condition.

380 Properties:

381 **[Code]** Sender

382 **[Subcode]** wsat:InconsistentInternalState

383 **[Reason]** A global consistency failure has occurred. This is an unrecoverable condition.

384 **[Detail]** Unspecified

## 385 **5.2 Unknown Transaction**

386 This fault is sent by a coordinator to indicate that it has no knowledge of the transaction and consequently  
387 cannot convey the outcome.

388 Properties:

389 [Code] Sender

390 **[Subcode]** wsat:UnknownTransaction

391 **[Reason]** The coordinator has no knowledge of the transaction. This is an unrecoverable condition.

392 **[Detail]** Unspecified

393

## 6 Security Model

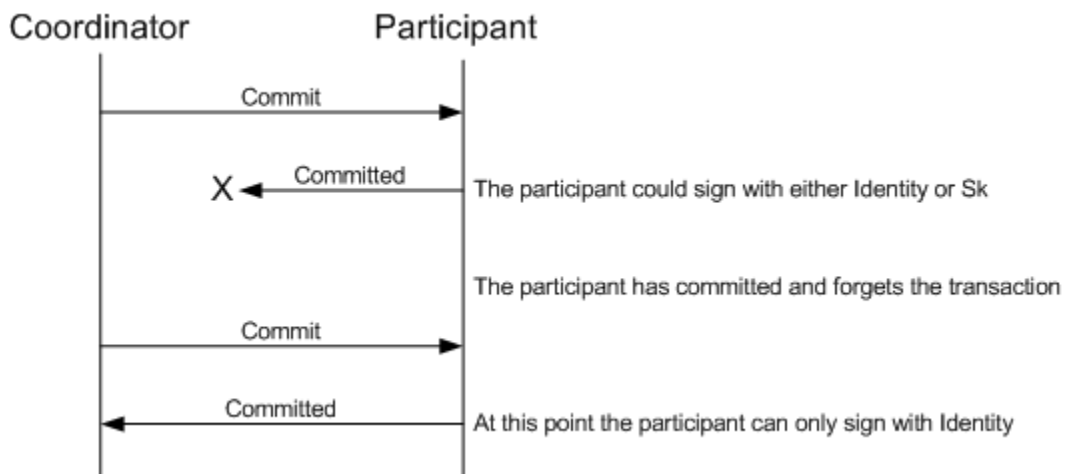
394 The security model for Atomic Transactions builds on the model defined in WS-Coordination [WSCOOR].  
395 That is, services have policies specifying their requirements and requestors provide claims (either implicit  
396 or explicit) and the requisite proof of those claims. Coordination context creation establishes a base  
397 secret which can be delegated by the creator as appropriate.

398 Because Atomic Transactions represent a specific use case rather than the general nature of  
399 coordination contexts, additional aspects of the security model can be specified.

400 All access to Atomic Transaction protocol instances is on the basis of identity. The nature of transactions,  
401 specifically the uncertainty of systems means that the security context established to register for the  
402 protocol instance may not be available for the entire duration of the protocol.

403 Consider, for example, the scenarios where a participant has committed its part of the transaction, but for  
404 some reason the coordinator never receives acknowledgement of the commit. The result is that when  
405 communication is re-established in the future, the coordinator will attempt to confirm the commit status of  
406 the participant, but the participant, having committed the transaction and forgotten all information  
407 associated with it, no longer has access to the special keys associated with the token.

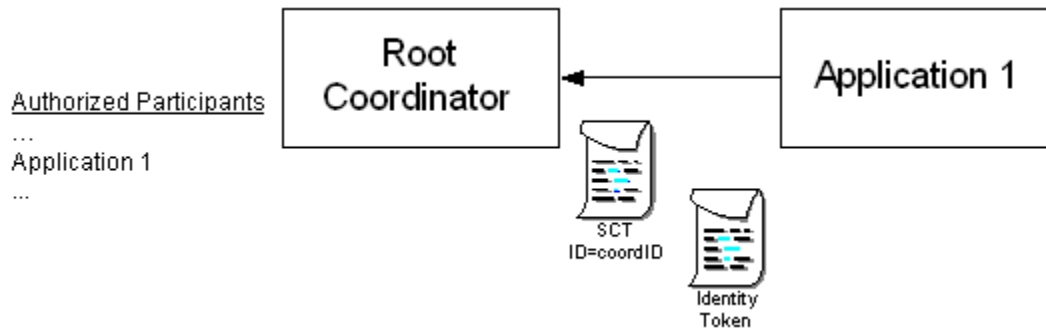
408 The participant can only prove its identity to the coordinator when it indicates that the specified  
409 transaction is not in its log and assumed committed. This is illustrated in the figure below:



410

411 There are, of course, techniques to mitigate this situation but such options will not always be successful.  
412 Consequently, when dealing with Atomic Transactions, it is critical that identity claims always be proven to  
413 ensure that correct access control is maintained by coordinators.

414 There is still value in coordination context-specific tokens because they offer a bootstrap mechanism so  
415 that all participants need not be pre-authorized. As well, it provides additional security because only those  
416 instances of an identity with access to the token will be able to securely interact with the coordinator  
417 (limiting privileges strategy). This is illustrated in the figure below:



418

419

420

421

The "list" of authorized participants ensures that application messages having a coordination context are properly authorized since altering the coordination context ID will not provide additional access unless (1) the bootstrap key is provided, or (2) the requestor is on the authorized participant "list" of identities.

---

## 422 7 Security Considerations

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

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

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

435 Attaching a nonce to each message and using it in a derived key function with the shared secret

436 Using a derived key sequence and switch "generations"

437 Closing and re-establishing a security context (not possible for delegated keys)

438 Exchanging new secrets between the parties (not possible for delegated keys)

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

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

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

449 **Message alteration** – Alteration is prevented by including signatures of the message information using  
450 WS-Security [WSSec].

451 **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security  
452 [WSSec].

453 **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing  
454 secured policies – see WS-Policy [WSPOLICY] and WS-SecurityPolicy [WSSecPolicy]).

455 **Authentication** – Authentication is established using the mechanisms described in WS-Security and WS-  
456 Trust [WSTrust]. Each message is authenticated using the mechanisms described in WS-Security  
457 [WSSec].

458 **Accountability** – Accountability is a function of the type of and string of the key and algorithms being  
459 used. In many cases, a strong symmetric key provides sufficient accountability. However, in some  
460 environments, strong PKI signatures are required.

461 **Availability** – Many services are subject to a variety of availability attacks. Replay is a common attack  
462 and it is RECOMMENDED that this be addressed as described in the next bullet. Other attacks, such as  
463 network-level denial of service attacks are harder to avoid and are outside the scope of this specification.  
464 That said, care should be taken to ensure that minimal processing be performed prior to any  
465 authenticating sequences.

466 **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack,  
467 mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-

468 Security [[WSec](#)]. Alternatively, and optionally, other technologies, such as sequencing, can also be used  
469 to prevent replay of application messages.

---

## 8 Use of WS-Addressing Headers

470

471 The protocols defined in WS-AtomicTransaction use a "one way" message exchange pattern consisting of  
472 a sequence of notification messages between a Coordinator and a Participant. There are two types of  
473 notification messages used in these protocols:

474 A notification message is a terminal message when it indicates the end of a coordinator/participant  
475 relationship. **Committed**, **Aborted** and **ReadOnly** are terminal messages, as are the protocol faults  
476 defined in this specification and in WS-Coordination [WSCOOR].

477 A notification message is a non-terminal message when it does not indicate the end of a  
478 coordinator/participant relationship. **Commit**, **Rollback**, **Prepare** and **Prepared** are non-terminal  
479 messages.

480 The following statements define addressing interoperability requirements for the Atomic Transaction  
481 message types:

482 Non-terminal notification messages:

- 483 • MUST include a [source endpoint] property whose [address] property is not set to  
484 'http://www.w3.org/2005/08/addressing/anonymous' or  
485 'http://www.w3.org/2005/08/addressing/none'.

486 Both terminal and non-terminal notification messages:

- 487 • MUST include a [reply endpoint] property whose [address] property is set to  
488 'http://www.w3.org/2005/08/addressing/none'.

489 Notification messages used in WS-AtomicTransaction protocols MUST include as the [action] property an  
490 action URI that consists of the wsat namespace URI concatenated with the "/" character and the element  
491 name of the message. For example:

492 `http://docs.oasis-open.org/ws-tx/wsat/2006/06/Commit`

493 Notification messages are normally addressed according to section 3.3 of WS-Addressing 1.0 – Core  
494 [WSADDR] by both coordinators and participants using the Endpoint References initially obtained during  
495 the Register-RegisterResponse exchange. If a [source endpoint] property is present in a notification  
496 message, it MAY be used by the recipient. Cases exist where a Coordinator or Participant has forgotten a  
497 transaction that is completed and needs to respond to a resent protocol message. In such cases, the  
498 [source endpoint] property SHOULD be used as described in section 3.3 of WS-Addressing 1.0 – Core  
499 [WSADDR]. Permanent loss of connectivity between a coordinator and a participant in an in-doubt state  
500 can result in data corruption.

501 Protocol faults raised by a Coordinator or Participant during the processing of a notification message are  
502 terminal notifications and MUST be composed using the same mechanisms as other terminal notification  
503 messages.

504 All messages are delivered using connections initiated by the sender.

## 505 9 State Tables

506 The following state tables specify the behavior of coordinators and participants when presented with  
 507 protocol messages or internal events.

508 Each cell in the tables uses the following convention:

509

Legend
Action to take
Next state

510

511 Each state supports a number of possible events. Expected events are processed by taking the  
 512 prescribed action and transitioning to the next state. Unexpected protocol messages MUST result in a  
 513 fault message as defined in the state tables. These faults use standard fault codes as defined in either  
 514 WS-Coordination [WSCOOR] or in section 5 Transaction Faults. Events that may not occur in a given  
 515 state are labeled as N/A.

516 Notes:

517 Transitions with a "N/A" as their action are inexpressible. A TM should view these transitions as serious  
 518 internal consistency issues that are likely fatal conditions.

519 The "Internal events" shown are those events, created either within a TM itself or on its local system, that  
 520 cause state changes and/or trigger the sending of a protocol message.

### 521 9.1 Completion Protocol

522

Completion Protocol (Coordinator View)			
Inbound Events	States		
	None	Active	Completing
Commit	Unknown Transaction None	Initiate user commit Completing	Ignore Completing
Rollback	Unknown Transaction None	Initiate user rollback, send aborted None	Invalid State Completing
Internal Events			
Commit Decision	N/A	N/A	Send committed None

Abort Decision	N/A	Send aborted None	Send aborted None
----------------	-----	----------------------	----------------------

523

## 524 9.2 2PC Protocol

525 These tables present the view of a coordinator or participant with respect to a single partner. A  
 526 coordinator with multiple participants can be understood as a collection of independent coordinator state  
 527 machines, each with its own state.

528

Atomic Transaction 2PC Protocol (Coordinator View)							
Inbound Events	States						
	None	Active	Preparing	Prepared	PreparedSuccess	Committing	Aborting
Prepared	Durable : Send Rollback Volatile: Unknown Transaction None	Invalid State Aborting	Record Vote Prepared	Ignore Prepared	Ignore PreparedSuccess	Resend Commit Committing	Resend Rollback Aborting
ReadOnly	Ignore None	Forget None	Forget None	Inconsistent Internal State Prepared	Inconsistent Internal State PreparedSuccess	Inconsistent Internal State Committing	Forget None
Aborted	Ignore None	Forget None	Forget None	Inconsistent Internal State Prepared	Inconsistent Internal State PreparedSuccess	Inconsistent Internal State Committing	Forget None
Committed	Ignore None	Invalid State Aborting	Invalid State Aborting	Inconsistent Internal State Prepared	Inconsistent Internal State PreparedSuccess	Forget None	Inconsistent Internal State Aborting
Internal Events							



User Commit	N/A	Send Prepare Preparing	N/A	N/A	N/A	N/A	N/A
User Rollback	N/A	Send Rollback Aborting	N/A	N/A	N/A	N/A	N/A
Expires Times Out	N/A	Send Rollback Aborting	Send Rollback Aborting	Send Rollback Aborting	Ignore PreparedSuccess	Ignore Committing	Ignore Aborting
Comms Times Out	N/A	N/A	Resend Prepare Preparing	N/A	N/A	Resend Committing	N/A
Commit Decision	N/A	N/A	N/A	Record Outcome PreparedSuccess	N/A	N/A	N/A
Rollback Decision	N/A	Send Rollback Aborting	Send Rollback Aborting	Send Rollback Aborting	N/A	N/A	N/A
Write Done	N/A	N/A	N/A	N/A	Send Commit Committing	N/A	N/A
Write Failed	N/A	N/A	N/A	N/A	Send Rollback Aborting	N/A	N/A
Participant Abandoned	N/A	N/A	N/A	N/A	N/A	Durable: N/A Volatile: None	None

529

530 "Forget" implies that the subordinate's participation is removed from the coordinator (if necessary), and  
531 otherwise the message is ignored

Atomic Transaction 2PC Protocol (Participant View)	
Inbound	States

Events	None	Active	Preparing	Prepared	PreparedSuccess	Committing
Prepare	Send Aborted None	Gather Vote Decision Preparing	Ignore Preparing	Ignore Prepared	Resend Prepared PreparedSuccess	Ignore Committing
Commit	Send Committed None	Invalid State None	Invalid State None	Invalid State None	Initiate Commit Decision Committing	Ignore Committing
Rollback	Send Aborted None	Initiate Rollback and Send Aborted None	Initiate Rollback and Send Aborted None	Initiate Rollback and Send Aborted None	Initiate Rollback and Send Aborted None	Inconsistent Internal State Committing
Internal Events						
Expires Times Out	N/A	Initiate Rollback and Send Aborted None	Initiate Rollback and Send Aborted None	Ignore Prepared	Ignore PreparedSuccess	Ignore Committing
Comms Times Out	N/A	N/A	N/A	N/A	Resend Prepared PreparedSuccess	N/A

Commit Decision	N/A	N/A	Record Commit Prepared	N/A	N/A	Send Committed None
Rollback Decision	N/A	Send Aborted None	Send Aborted None	N/A	N/A	N/A
Write Done	N/A	N/A	N/A	Send Prepared Prepared Success	N/A	N/A
Write Failed	N/A	N/A	N/A	Initiate Rollback and Send Aborted None	N/A	N/A
ReadOnly Decision	N/A	Send ReadOnly None	Send ReadOnly None	N/A	N/A	N/A

532

---

533

## A. Acknowledgements

534 This document is based on initial contributions to the OASIS WS-TX Technical Committee by the  
535 following authors: Luis Felipe Cabrera (Microsoft), George Copeland (Microsoft), Max Feingold  
536 (Microsoft), Robert W Freund (Hitachi), Tom Freund (IBM), Jim Johnson (Microsoft), Sean Joyce (IONA),  
537 Chris Kaler (Microsoft), Johannes Klein (Microsoft), David Langworthy (Microsoft), Mark Little (Arjuna  
538 Technologies), Frank Leymann (IBM), Eric Newcomer (IONA), David Orchard (BEA Systems), Ian  
539 Robinson (IBM), Tony Storey (IBM), Satish Thatte (Microsoft).

540

541 The following individuals have provided invaluable input into the initial contribution: Francisco Curbera  
542 (IBM), Doug Davis (IBM), Gert Drapers (Microsoft), Don Ferguson (IBM), Kirill Gavrylyuk (Microsoft), Dan  
543 House (IBM), Oisin Hurley (IONA), Thomas Mikalsen (IBM), Jagan Peri (Microsoft), John Shewchuk  
544 (Microsoft), Stefan Tai (IBM).

545

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

### 547 **Participants:**

548       Charlton Barreto, Adobe Systems, Inc.  
549       Martin Chapman, Oracle  
550       Kevin Conner, JBoss Inc.  
551       Paul Cotton, Microsoft Corporation  
552       Doug Davis, IBM  
553       Colleen Evans, Microsoft Corporation  
554       Max Feingold, Microsoft Corporation  
555       Thomas Freund, IBM  
556       Robert Freund, Hitachi, Ltd.  
557       Peter Furniss, Choreology Ltd.  
558       Marc Goodner, Microsoft Corporation  
559       Alastair Green, Choreology Ltd.  
560       Daniel House, IBM  
561       Ram Jeyaraman, Microsoft Corporation  
562       Paul Knight, Nortel Networks Limited  
563       Mark Little, JBoss Inc.  
564       Jonathan Marsh, Microsoft Corporation  
565       Monica Martin, Sun Microsystems  
566       Joseph Fialli, Sun Microsystems  
567       Eric Newcomer, IONA Technologies  
568       Eisaku Nishiyama, Hitachi, Ltd.  
569       Alain Regnier, Ricoh Company, Ltd.  
570       Ian Robinson, IBM  
571       Tom Rutt, Fujitsu Limited  
572       Andrew Wilkinson, IBM