Web Services Business Activity (WS-Business Activity)

Working Draft 1.1, January 26, 2006

Document Identifier: wstx-wsba-1.1-spec-wd-02

Location: http://docs.oasis-open.org/wstx/wsba-1.1-spec-wd-02.pdf

Technical Committee: OASIS WS-Tx TC

Chair(s): Eric Newcomer, Iona
Ian Robinson, IBM

Editor(s): Tom Freund, IBM <tfreund@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 modify 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 Composable Architecture ....................................................................................... 5
   1.4 Namespace ............................................................................................................. 5
      1.4.1 Prefix Namespace ............................................................................................... 5
   1.5 Normative References ............................................................................................ 6
   1.6 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

---

Deleted: 1. Introduction 4
1.1 Model 5
1.2 Terminology 5
1.3 Composable Architecture 5
1.4 Namespace 5
1.4.1 Prefix Namespace 5
1.5 Normative References 6
1.6 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

Deleted: 1
Deleted: 5
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.
1.1 Model

Business Activity Coordination protocols provide the following flexibility:

- A business application may be partitioned into business activity scopes. A business activity scope is a business task consisting of a general-purpose computation carried out as a bounded set of operations on a collection of Web services that require a mutually agreed outcome. There can be any number of hierarchical nesting levels. Nested scopes:
  - Allow a business application to select which child tasks are included in the overall outcome processing. For example, a business application might solicit an estimate from a number of suppliers and choose a quote or bid based on lowest-cost.
  - Allow a business application to catch an exception thrown by a child task, apply an exception handler, and continue processing even if something goes wrong. When a child completes its work, it may be associated with a compensation that is registered with the parent activity.

- A participant task within a business activity may specify that it is leaving a business activity. This provides the ability to exit a business activity and allows business programs to delegate processing to other scopes. In contrast to atomic transactions, the participant list is dynamic and a participant may exit the protocol at any time without waiting for the outcome of the protocol.

- It allows a participant task within a business activity to specify its outcome directly without waiting for solicitation. Such a feature is generally useful when a task fails so that the notification can be used by a business activity exception handler to modify the goals and drive processing in a timely manner.

- It allows participants in a coordinated business activity to perform "tentative" operations as a normal part of the activity. The result of such "tentative" operations may become visible before the activity is complete and may require business logic to run in the event that the operation needs to be compensated. Such a feature is critical when the joint work of a business activity requires many operations performed by independent services over a long period of time.

1.2 Terminology

The 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 .

1.3 Composable Architecture

By using the SOAP [SOAP] 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-BusinessActivity by itself does not define all features required for a complete solution. WS-BusinessActivity is a building block used with other specifications of Web services (e.g., WS-Coordination, WS-Security) and application-specific protocols that are able to accommodate a wide variety of coordination protocols related to the coordination actions of distributed applications.

1.4 Namespace

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

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

1.4.1 Prefix Namespace

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td><a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a></td>
</tr>
</tbody>
</table>
If an action URI is used then the action URI MUST consist of the wsba namespace URI concatenated with the "/" character and the element name. For example:

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

### 1.5 Normative References


[ WSCOOR ] Web Services Coordination (WS-Coordination), "http://docs.oasis-open.org/wstx/ws-coordination-1.1-spec-01.pdf"

[ WSDL ] Web Services Description Language (WSDL) 1.1, "http://www.w3.org/TR/2001/NOTE-wsdl-20010315"

[ WSADDR ] Web Services Addressing (WS-Addressing), Microsoft, IBM, Sun, BEA Systems, SAP, Sun, August 2004


[ BPEL ] Web Services Business Process Execution Language, Microsoft, BEA and IBM.


2 Using WS-Coordination

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

2.1 Coordination Context

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


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

A CoordinationContext can have additional elements for extensibility.

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

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

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


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

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

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

3.1 BusinessAgreementWithParticipantCompletion Protocol

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

Participants register for this protocol using the following protocol identifier:

- [http://schemas.xmlsoap.org/ws/2004/10/wsba/ParticipantCompletion](http://schemas.xmlsoap.org/ws/2004/10/wsba/ParticipantCompletion)

The coordinator accepts:

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

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

- **Compensated**: Upon receipt of this notification, the coordinator knows that the participant has recorded a compensation request for a protocol.

- **Closed**: Upon receipt of this notification, the coordinator knows that the participant has finalized successfully.

- **Canceled**
Upon receipt of this notification, the coordinator knows that the participant has finalized
successfully processing the Cancel notification.

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

The participant accepts:

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

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

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

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

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

Both the coordinator and participant accept:

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

Status
Upon receipt of this message the target service returns a QName defined in schema indicating
the current state of the coordinator or participant. For example, if a participant is in the closing
state as indicated by the state table, it would return wsba:Closing.
The coordinator can enter a condition in which it has sent a protocol message and it receives a protocol message from the participant that is consistent with the former state, not the current state. In this case, it is the responsibility of the coordinator to revert to the prior state, accept the notification from the participant, and continue the protocol from that point. If the participant detects this condition, it must discard the inconsistent protocol message from the coordinator.

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

### 3.2 BusinessAgreementWithCoordinatorCompletion Protocol

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

Participants register for this protocol using the following protocol identifier:

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

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

The participant accepts:

**Complete**

Upon receipt of this notification the participant knows that it will receive no new requests for work within the business activity. It should complete application processing and transmit the Completed notification.
Figure 2: BusinessAgreementWithCoordinatorCompletion abstract state diagram
4 WS-BA Policy Assertions

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

4.1 Assertion Models

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

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

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

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

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

4.2 Normative Outlines

The normative outlines for the BA policy assertions are:

<wba:BAAtomicOutcomeAssertion [wsp:Optional="true"]? ... > ... </wba:BAAtomicOutcomeAssertion>

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

/wsba:BAAtomicOutcomeAssertion

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

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

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

<wba:BMixedOutcomeAssertion [wsp:Optional="true"]? ... > ... </wba:BMixedOutcomeAssertion>

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

/wsba:BMixedOutcomeAssertion

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

/wsba:BMixedOutcomeAssertion/@wsp:Optional="true"

Deleted: 5
/wsba: BAMixedOutcomeAssertion/@wsp:Optional="true"

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

4.3 Assertion Attachment

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

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

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

4.4 Assertion Example

An example use of the BA policy assertion follows:

```
(01) <wsdl:definitions
(02)     targetNamespace="hotel.example.com"
(03)     xmlns:tns="hotel.example.com"
(04)     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
(05)     xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/09/policy"
(06)     xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
(07)     xmlns:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
(08)                 wssecurity-utility-1.0.xsd" >
(09)   <wsp:Policy wsu:Id="BAAtomicPolicy" >
(10)     <wsba:BAAtomicOutcomeAssertion/>
(11)   </wsp:Policy>
(12)   </wsdl:definitions>
```

Deleted: 1
Deleted: 5
Lines (9-12) are a policy expression that includes a BA policy assertion (Line 10) to indicate that a coordination context for a business activity with an AtomicOutcome, expressed in WS-Coordination [WS-Coordination], format MUST be used.

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

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

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

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

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

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

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

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

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

Deleted: 1
Deleted: 5
6 Interoperability Considerations

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

Cancel
Back out of a business activity.

Close
Terminate a business activity with a favorable outcome.

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

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

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

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

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

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

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

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

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

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

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

Participants:

[Participant Name, Affiliation | Individual Member]

[Participant Name, Affiliation | Individual Member]
## Appendix B. Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>11/22/2005</td>
<td>Tom Freund</td>
<td>Initial Working Draft</td>
</tr>
<tr>
<td>02</td>
<td>01/28/2006</td>
<td>Tom Freund</td>
<td>WS-TX: Issue #17, Specification Inconsistencies</td>
</tr>
</tbody>
</table>
# Appendix C. State Tables for the Agreement Protocols

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

![Action to take in next state]

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

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

- For the period of time that a protocol message is in transit the sender and recipient states will be different.
- The sender of a protocol message transitions to the "next state" when the message is first sent.
- The recipient of a protocol message transitions to the "next state" when the message is first received.
- As described earlier in this document, if the coordinator receives a protocol message from the participant that is consistent with the former state of the coordinator then the coordinator reverts to its prior state, accepts the notification from the participant, and continues the protocol from that point.
- The GetStatus and Status protocol messages are not included in the tables as these never result in a change of state.

## C.1. Participant view of BusinessAgreementWithParticipantCompletion

<table>
<thead>
<tr>
<th>Participant view of state</th>
<th>Protocol messages received by Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cancel</td>
</tr>
<tr>
<td>Active</td>
<td>Canceling</td>
</tr>
<tr>
<td>Canceling</td>
<td>Ignore</td>
</tr>
<tr>
<td>Completed</td>
<td>Resend Completed</td>
</tr>
<tr>
<td>Closing</td>
<td>Ignore</td>
</tr>
<tr>
<td>Compensating</td>
<td>Ignore</td>
</tr>
<tr>
<td>Faulting (Active, Completed)</td>
<td>Resend Fault Faulting</td>
</tr>
<tr>
<td>Faulting (Compensating)</td>
<td>Ignore</td>
</tr>
<tr>
<td>Exiting</td>
<td>Resend Exit</td>
</tr>
<tr>
<td>Ended</td>
<td>Send Canceled</td>
</tr>
</tbody>
</table>

Deleted: 1
Deleted: 5
### BusinessAgreementWithParticipantCompletion

<table>
<thead>
<tr>
<th>Component</th>
<th>Action</th>
<th>Protocol messages received by Coordinator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>Exiting</td>
<td>Completed Fault Cancelled Closed Compensated</td>
</tr>
<tr>
<td><strong>Cancelling</strong></td>
<td>Invalid State</td>
<td>Cancelled Faulted-Active Cancelled Cancelled Closed Compensated</td>
</tr>
<tr>
<td><strong>Completed</strong></td>
<td>Invalid State</td>
<td>Completed Cancelled Completed Completed Completed Completed</td>
</tr>
<tr>
<td><strong>Closing</strong></td>
<td>Invalid State</td>
<td>Cancelled Cancelled Cancelled Cancelled Closed Closed Closed</td>
</tr>
<tr>
<td><strong>Compensating</strong></td>
<td>Invalid State</td>
<td>Compensated Faulted-Compensated Compensated Compensated Compensated Compensated</td>
</tr>
<tr>
<td><strong>Faulting</strong></td>
<td>Invalid State</td>
<td>Faulting Faulting Faulting Faulting Faulting Faulting</td>
</tr>
<tr>
<td><strong>Exiting</strong></td>
<td>Invalid State</td>
<td>Exiting Exiting Exiting Exiting Exiting Exiting</td>
</tr>
<tr>
<td><strong>Ended</strong></td>
<td>Invalid State</td>
<td>Ended Ended Ended Ended Ended Ended</td>
</tr>
</tbody>
</table>

### C.2. Coordinator view of BusinessAgreementWithParticipantCompletion

<table>
<thead>
<tr>
<th>Component</th>
<th>Action</th>
<th>Protocol messages sent by Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>Exiting</td>
<td>Completed Fault Cancelled Closed Compensated</td>
</tr>
<tr>
<td><strong>Cancelling</strong></td>
<td>Invalid State</td>
<td>Cancelled Faulted-Active Cancelled Cancelled Closed Compensated</td>
</tr>
<tr>
<td><strong>Completed</strong></td>
<td>Invalid State</td>
<td>Completed Cancelled Completed Completed Completed Completed</td>
</tr>
<tr>
<td><strong>Closing</strong></td>
<td>Invalid State</td>
<td>Cancelled Cancelled Cancelled Cancelled Closed Closed Closed</td>
</tr>
<tr>
<td><strong>Compensating</strong></td>
<td>Invalid State</td>
<td>Compensated Faulted-Compensated Compensated Compensated Compensated Compensated</td>
</tr>
<tr>
<td><strong>Faulting</strong></td>
<td>Invalid State</td>
<td>Faulting Faulting Faulting Faulting Faulting Faulting</td>
</tr>
<tr>
<td><strong>Exiting</strong></td>
<td>Invalid State</td>
<td>Exiting Exiting Exiting Exiting Exiting Exiting</td>
</tr>
<tr>
<td><strong>Ended</strong></td>
<td>Invalid State</td>
<td>Ended Ended Ended Ended Ended Ended</td>
</tr>
</tbody>
</table>
### BusinessAgreementWithParticipantCompletion protocol

<table>
<thead>
<tr>
<th>Coordinator view of state</th>
<th>Protocol messages sent by Coordinator</th>
<th>Protocol messages sent by Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cancel</td>
<td>Close</td>
</tr>
<tr>
<td>Active</td>
<td>Canceling-Active</td>
<td>Invalid State Active</td>
</tr>
<tr>
<td>Canceling</td>
<td>Canceling</td>
<td>Invalid State Canceling</td>
</tr>
<tr>
<td>Completed</td>
<td>Invalid State Completed</td>
<td>Closing</td>
</tr>
<tr>
<td>Closing</td>
<td>Invalid State Closing</td>
<td>Closing</td>
</tr>
<tr>
<td>Compensating</td>
<td>Invalid State Compensating</td>
<td>Invalid State Compensating</td>
</tr>
<tr>
<td>Faulting</td>
<td>Invalid State Faulting</td>
<td>Invalid State Faulting</td>
</tr>
<tr>
<td>Exiting</td>
<td>Invalid State Exiting</td>
<td>Invalid State Exiting</td>
</tr>
<tr>
<td>Ended</td>
<td>Invalid State Ended</td>
<td>Invalid State Ended</td>
</tr>
</tbody>
</table>

### BusinessAgreementWithCoordinatorCompletion protocol

<table>
<thead>
<tr>
<th>Participant view of state</th>
<th>Protocol messages received by Participant</th>
<th>Protocol messages received by Coordinator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cancel</td>
<td>Complete</td>
</tr>
<tr>
<td>Active</td>
<td>Canceling</td>
<td>Completing</td>
</tr>
<tr>
<td>Canceling</td>
<td>Ignore Canceling</td>
<td>Ignore Completing</td>
</tr>
<tr>
<td>Completed</td>
<td>Canceling</td>
<td>Ignore Completing</td>
</tr>
<tr>
<td>Completed</td>
<td>Record Completed</td>
<td>Record Completed</td>
</tr>
<tr>
<td>Closing</td>
<td>Ignore Closing</td>
<td>Ignore Closing</td>
</tr>
<tr>
<td>Compensating</td>
<td>Ignore Compensating</td>
<td>Ignore Compensating</td>
</tr>
<tr>
<td>Faulting (Active, Completed)</td>
<td>Record Fault</td>
<td>Record Fault</td>
</tr>
<tr>
<td>Faulting (Compensating)</td>
<td>Ignore Faulting</td>
<td>Ignore Faulting</td>
</tr>
<tr>
<td>Exiting</td>
<td>Record Exit</td>
<td>Record Exit</td>
</tr>
<tr>
<td>Ended</td>
<td>Send Cancel</td>
<td>Ignore</td>
</tr>
</tbody>
</table>
### BusinessAgreementWithCoordinatorCompletion

#### Participant view of state

<table>
<thead>
<tr>
<th>Participant view of state</th>
<th>Protocol messages sent by Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Exit Completed Fault Cancelled Closed Compensated</td>
</tr>
<tr>
<td>Closing</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Completed</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Closing</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Completed</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
</tbody>
</table>

#### Coordinator view of state

<table>
<thead>
<tr>
<th>Coordinator view of state</th>
<th>Protocol messages received by Coordinator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Cancellation</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Cancellation-Completed</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Completing</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Completed</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Closing</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Completed</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Correction</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>Error</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
<tr>
<td>End</td>
<td>Exit Completed Faulting-Active Faulting-Active Cancelled Completed</td>
</tr>
</tbody>
</table>

### C.4. Coordinator view of BusinessAgreementWithCoordinatorCompletion
### BusinessAgreementWithCoordinatorCompletion protocol

<table>
<thead>
<tr>
<th>Coordinator view of state</th>
<th>Protocol messages Sent by Coordinator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cancel</td>
</tr>
<tr>
<td>Active</td>
<td>Canceling-Active</td>
</tr>
<tr>
<td>Canceling</td>
<td>Canceling</td>
</tr>
<tr>
<td>Completing</td>
<td>Canceling-Completing</td>
</tr>
<tr>
<td>Completed</td>
<td>Invalid State Completed</td>
</tr>
<tr>
<td>Closing</td>
<td>Invalid State Closing</td>
</tr>
<tr>
<td>Compensating</td>
<td>Invalid State Compensating</td>
</tr>
<tr>
<td>Faulting</td>
<td>Invalid State Faulting</td>
</tr>
<tr>
<td>Exiting</td>
<td>Invalid State Exiting</td>
</tr>
<tr>
<td>Ended</td>
<td>Invalid State Ended</td>
</tr>
</tbody>
</table>