Web Services Coordination (WS-Coordination) Version 1.1

OASIS Standard incorporating Approved Errata

26 May 2007

Specification URIs:
This Version:
> http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-errata-pr/wstx-wscoor-1.1-spec-errata-pr.html
> http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-errata-pr.pdf

Previous Version:
> http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os/wstx-wscoor-1.1-spec-os.html
> http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os.pdf

Latest Approved Version:
> http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec/wstx-wscoor-1.1-spec.html
> http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec.pdf

Technical Committee:
> OASIS WS-TX TC

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

Editor(s):
> Max Feingold, Microsoft
> Ram Jeyaraman, Microsoft

Declared XML Namespaces:
> http://docs.oasis-open.org/ws-tx/wscoor/2006/06

Abstract:
> The WS-Coordination specification describes an extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed activities.
>
> The framework defined in this specification enables an application service to create a context needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment.
>
> Additionally this specification describes a definition of the structure of context and the requirements for propagating context between cooperating services.

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

Deleted: 1
Deleted: April
Deleted: os
Field Code Changed
Deleted: os
Field Code Changed
Deleted: os
Deleted: c
Field Code Changed
Deleted: -03
Deleted: c
Deleted: -03
Field Code Changed
Deleted: c
Deleted: -03
Deleted: os
Deleted: 1
Deleted: Apri
Deleted: l

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.

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

Notices

	Deleted: os
	Deleted: 1
	Deleted: Apri
	Deleted: l

Table of Contents

1 Introduction

The current set of Web service specifications (SOAP **[SOAP 1.1] [SOAP 1.2]** and WSDL **[WSDL]**) defines protocols for Web service interoperability. Web services increasingly tie together a large number of participants forming large distributed computational units – we refer to these computation units as activities.

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

This specification defines an extensible framework for coordinating activities using a coordinator and set of coordination protocols. This framework enables participants to reach consistent agreement on the outcome of distributed activities. The coordination protocols that can be defined in this framework can accommodate a wide variety of activities, including protocols for simple short-lived operations and protocols for complex long-lived business activities. For example, WS-AtomicTransaction **[WSAT]** and WS-BusinessActivity **[WSBA]** specifications use and build upon this specification.

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

1.1 Model

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

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

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

A coordination type-specific set of coordination protocols.

This is illustrated below in Figure 1.

Figure 1: A Coordinator
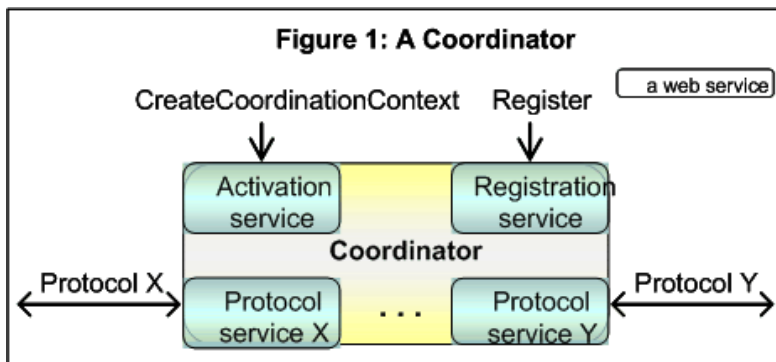
Applications use the Activation service to create the coordination context for an activity. Once a coordination context is acquired by an application, it is then sent by whatever appropriate means to another application.

The context contains the necessary information to register into the activity specifying the coordination behavior that the application will follow.

Additionally, an application that receives a coordination context may use the Registration service of the original application or may use one that is specified by an interposing, trusted coordinator. In this manner an arbitrary collection of Web services may coordinate their joint operation.

Deleted: os	
Deleted: 1	
Deleted: Apri	
Deleted: l	

1.2 Composable Architecture

By using the XML **[XML]**, SOAP **[SOAP 1.1] [SOAP 1.2]** and WSDL **[WSDL]** extensibility model, SOAP-based and WSDL-based specifications are designed to be composed with each other to define a rich Web services environment. As such, WS-Coordination by itself does not define all the features required for a complete solution. WS-Coordination is a building block that is used in conjunction with other specifications and application-specific protocols to accommodate a wide variety of protocols related to the operation of distributed Web services.

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

1.3 Extensibility

The specification provides for extensibility and flexibility along two dimensions. The framework allows for:

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

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

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

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

1.4 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]**.

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

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

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

1.5 Namespace

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

81
```
http://docs.oasis-open.org/ws-tx/wscoor/2006/06
```

1.5.1 Prefix Namespace

83 The following namespaces are used in this document:

Prefix	Namespace
S11	http://schemas.xmlsoap.org/soap/envelope
S12	http://www.w3.org/2003/05/soap-envelope
Wscoor	http://docs.oasis-open.org/ws-tx/wscoor/2006/06
Wsa	http://www.w3.org/2005/08/addressing

1.6 XSD and WSDL Files

85 Dereferencing the XML namespace defined in section 1.5 will produce the Resource Directory
86 Description Language (RDDL) **[RDDL]** document that describes this namespace, including the XML
87 schema **[XML-Schema1] [XML-Schema2]** and WSDL **[WSDL]** declarations associated with this
88 specification.

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

1.7 Coordination Protocol Elements

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

1.8 Normative References

97 **[RDDL]**	Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language	
98	(RDDL) 2.0", http://www.openhealth.org/RDDL/20040118/rddl-20040118.html,	
99	January 2004.	
100 **[RFC2119]**	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels",	
101	http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.	
102 **[SOAP 1.1]**	W3C Note, "SOAP: Simple Object Access Protocol 1.1,"	
103	http://www.w3.org/TR/2000/NOTE-SOAP-20000508, 08 May 2000.	
104 **[SOAP 1.2]**	W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework",	
105	http://www.w3.org/TR/soap12-part1, June 2003.	
106 **[XML]**	W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth	
107	Edition),"http://www.w3.org/TR/2006/REC-xml-20060816, 16 August 2006.	
108 **[XML-ns]**	W3C Recommendation, "Namespaces in XML 1.0 (Second Edition),"	
109	http://www.w3.org/TR/2006/REC-xml-names-20060816, 16 August 2006.	
110 **[XML-Schema1]**	W3C Recommendation, "XML Schema Part 1: Structures Second Edition,"	
111	http://www.w3.org/TR/2004/REC-xmlschema-1-20041028, 28 October 2004.	
112 **[XML-Schema2]**	W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition,"	
113	http://www.w3.org/TR/2004/REC-xmlschema-2-20041028, 28 October 2004.	

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

114	**[WSADDR]**	Web Services Addressing (WS-Addressing) 1.0, W3C Recommendation,
115		http://www.w3.org/2005/08/addressing.
116	**[WSDL]**	Web Services Description Language (WSDL) 1.1
117		http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
118	**[WSPOLICY]**	Web Services Policy 1.2 – Framework (WS-Policy),
119		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/, W3C Member
120		Submission, 25 April 2006.
121	**[WSSec]**	OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message
122		Security 1.0 (WS-Security 2004)", http://docs.oasis-open.org/wss/2004/01/oasis-
123		200401-wss-soap-message-security-1.0.pdf.
124	**[WSSecPolicy]**	Web Services Security Policy Language (WS-SecurityPolicy),
125		http://schemas.xmlsoap.org/ws/2005/07/securitypolicy, Microsoft, VeriSign, IBM,
126		and RSA Security Inc., July 2005.
127	**[WSSecConv]**	Web Services Secure Conversation Language (WS-SecureConversation),
128		http://schemas.xmlsoap.org/ws/2005/02/sc, OpenNetwork, Layer7, Netegrity,
129		Microsoft, Reactivity, IBM, VeriSign, BEA Systems, Oblix, RSA Security, Ping
130		Identity, Westbridge, Computer Associates, February 2005.
131	**[WSTrust]**	Web Services Trust Language (WS-Trust),
132		http://schemas.xmlsoap.org/ws/2005/02/trust, OpenNetwork, Layer7, Netegrity,
133		Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping
134		Identity, Westbridge, Computer Associates, February 2005.

135   1.9 Non-normative References

136

137	**[WSAT]**	Web Services Atomic Transaction (WS-AtomicTransaction)
138		http://docs.oasis-open.org/ws-tx/wsat/2006/06.
139	**[WSBA]**	Web Services Business Activity (WS-BusinessActivity)
140		http://docs.oasis-open.org/ws-tx/wsba/2006/06.

Deleted: Web Services Policy Framework (WS-Policy), http://schemas.xmlsoap.org/ws/2004/09/policy, VeriSign, Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004.

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

141 2 Coordination Context

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

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

```
150  <?xml version="1.0" encoding="utf-8"?>
151  <S11:Envelope xmlns:S11="http://www.w3.org/2003/05/soap-envelope">
152      <S11:Header>
153          . . .
154          <wscoor:CoordinationContext
155              xmlns:wsa="http://www.w3.org/2005/08/addressing"
156              xmlns:wscoor="http://docs.oasis-open.org/ws-tx/wscoor/2006/06"
157              xmlns:myApp="http://www.example.com/myApp"
158              S11:mustUnderstand="true">
159              <wscoor:Identifier>
160                    http://Fabrikam123.com/SS/1234
161              </wscoor:Identifier>
162              <wscoor:Expires>3000</wscoor:Expires>
163              <wscoor:CoordinationType>
164                 http://docs.oasis-open.org/ws-tx/wsat/2006/06
165              </wscoor:CoordinationType>
166              <wscoor:RegistrationService>
167                  <wsa:Address>
168                   http://Business456.com/mycoordinationservice/registration
169                  </wsa:Address>
170                  <wsa:ReferenceParameters>
171                    <myApp:BetaMark> ... </myApp:BetaMark>
172                    <myApp:EBDCode> ... </myApp:EBDCode>
173                  </wsa:ReferenceParameters>
174              </wscoor:RegistrationService>
175              <myApp:IsolationLevel>
176                    RepeatableRead
177              </myApp:IsolationLevel>
178          </wscoor:CoordinationContext>
179          . . .
180      </S11:Header>
181      </S11:Body>
182          . . .
183      </S11:Body >
184  </S11:Envelope>
185
```

186 When an application propagates an activity using a coordination service, applications MUST include a
187 CoordinationContext in the message.

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

3 Coordination Service

190

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

192 • Activation service: Defines a CreateCoordinationContext operation that allows a
193 CoordinationContext to be created. The exact semantics are defined in the specification that
194 defines the coordination type. The Coordination service MAY support the Activation service.

195 • Registration service: Defines a Register operation that allows a Web service to register to
196 participate in a coordination protocol. The Coordination service MUST support the Registration
197 service.

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

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

204 1. App1 sends a CreateCoordinationContext for coordination type Q, getting back a Context Ca that
205 contains the activity identifier A1, the coordination type Q and an Endpoint Reference to
206 CoordinatorA's Registration service RSa.

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

208 3. App2 prefers to use CoordinatorB instead of CoordinatorA, so it uses CreateCoordinationContext
209 with Ca as an input to interpose CoordinatorB. CoordinatorB creates its own CoordinationContext
210 Cb that contains the same activity identifier and coordination type as Ca but with its own
211 Registration service RSb.

212 4. App2 determines the coordination protocols supported by the coordination type Q and then
213 Registers for a coordination protocol Y at CoordinatorB, exchanging Endpoint References for
214 App2 and the protocol service Yb. This forms a logical connection between these Endpoint
215 References that the protocol Y can use.

216 5. This registration causes CoordinatorB to decide to immediately forward the registration onto
217 CoordinatorA's Registration service RSa, exchanging Endpoint References for Yb and the
218 protocol service Ya. This forms a logical connection between these Endpoint References that the
219 protocol Y can use.

220 Figure 2: Two applications with their own coordinators
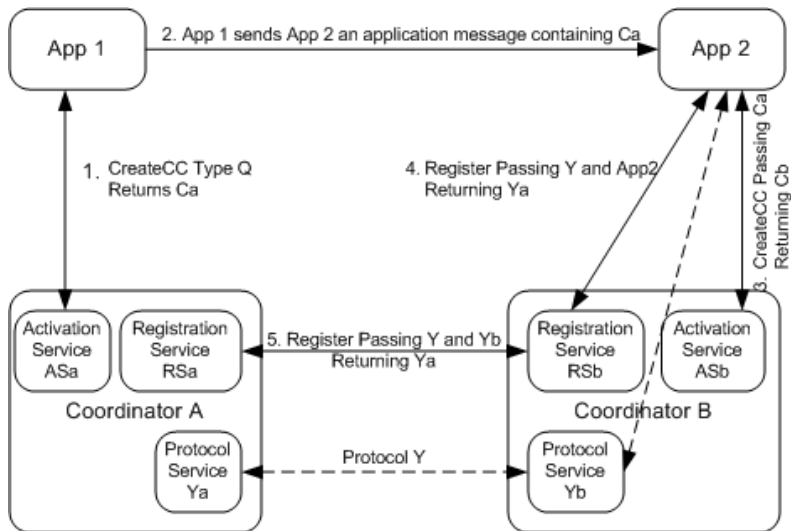
221

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

3.1 Activation Service

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

228 An application sends:

229 CreateCoordinationContext

230   The structure and semantics of this message are defined in Section 3.1.1.

231 The activation service returns:

232 CreateCoordinationContextResponse

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

3.1.1 CreateCoordinationContext

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

240 The following pseudo schema defines this element:

```
<CreateCoordinationContext ...>
    <Expires> ... </Expires>?
    <CurrentContext> ... </CurrentContext>?
    <CoordinationType> ... </CoordinationType>
    ...
</CreateCoordinationContext>
```

247

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

| Deleted: os |
| Deleted: 1 |
| Deleted: Apri |
| Deleted: l |

250   /CreateCoordinationContext/CoordinationType

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

253   /CreateCoordinationContext/Expires

254   Optional. The expiration for the returned CoordinationContext expressed as an unsigned integer
255   in milliseconds.

256   /CreateCoordinationContext/CurrentContext

257   Optional. If absent, the Activation Service creates a coordination context representing a new,
258   independent activity. If present, the Activation Service creates a coordination context representing
259   a new activity which is related to the existing activity identified by the current coordination context
260   contained in this element. Some examples of potential uses of this type of relationship include
261   interposed subordinate coordination, protocol bridging and coordinator replication.

262   /CreateCoordinationContext /{any}

263   Extensibility elements may be used to convey additional information.

264   /CreateCoordinationContext /@{any}

265   Extensibility attributes may be used to convey additional information.

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

```
267    <CreateCoordinationContext>
268        <CoordinationType>
269            http://docs.oasis-open.org/ws-tx/wsat/2006/06
270        </CoordinationType>
271    </CreateCoordinationContext>
```

272   3.1.2 CreateCoordinationContextResponse

273   This returns the CoordinationContext that was created.

274   The following pseudo schema defines this element:

```
275    <CreateCoordinationContextResponse ...>
276        <CoordinationContext> ... </CoordinationContext>
277        ...
278    </CreateCoordinationContextResponse>
```

279   /CreateCoordinationContext/CoordinationContext

280   This is the created coordination context.

281   /CreateCoordinationContext /{any}

282   Extensibility elements may be used to convey additional information.

283   /CreateCoordinationContext /@{any}

284   Extensibility attributes may be used to convey additional information.

285   The following example illustrates a response:

```
286    <CreateCoordinationContextResponse>
287        <CoordinationContext>
288            <Identifier>
289                http://Business456.com/tm/context1234
290            </Identifier>
291            <CoordinationType>
292                http://docs.oasis-open.org/ws-tx/wsat/2006/06
293            </CoordinationType>
294            <RegistrationService>
295                <wsa:Address>
296                    http://Business456.com/tm/registration
```

Formatted: English U.S.

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

```
297                    </wsa:Address>
298                    <wsa:ReferenceParameters>
299                      <myapp:PrivateInstance>
300                         1234
301                      </myapp:PrivateInstance>
302                    </wsa:ReferenceParameters>
303                </RegistrationService>
304           </CoordinationContext>
305      </CreateCoordinationContextResponse>
```

306 3.2 Registration Service

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

310 The requester sends:

311 Register

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

313 The coordinator's registration service responds with:

314 Registration Response

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

316 Figure 3: The usage of Endpoint References during registration
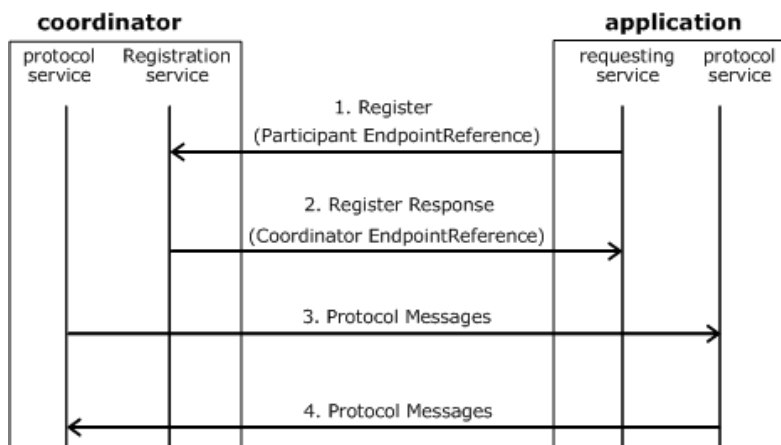
317

318 In Figure 3, the coordinator provides the Registration Endpoint Reference in the CoordinationContext
319 during the CreateCoordinationContext operation. The requesting service receives the Registration service
320 Endpoint Reference in the CoordinationContext in an application message.

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

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

324 3. & 4.) At this point, both sides have the Endpoint References of the other's protocol service, so the
325 protocol messages can target the other side.

326 These Endpoint References may contain (opaque) wsa:ReferenceParameters to fully qualify the target
327 protocol service endpoint. Endpoint References MUST be interpreted according to the rules defined in
328 WS-Addressing 1.0 Core **[WSADDR]**.

Deleted: os	
Deleted: 1	
Deleted: Apri	
Deleted: l	

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

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

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

339 3.2.1 Register Message

340 The Register request is used to do the following:

341 • Participant selection and registration in a particular Coordination protocol under the current
342   coordination type supported by the Coordination Service.

343 • Exchange Endpoint References. Each side of the coordination protocol (participant and
344   coordinator) supplies an Endpoint Reference.

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

347 The following pseudo schema defines this element:

```
348  <Register ...>
349      <ProtocolIdentifier> ... </ProtocolIdentifier>
350      <ParticipantProtocolService> ... </ParticipantProtocolService>
351      ...
352  </Register>
```

353 /Register/ProtocolIdentifier

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

355 /Register/ParticipantProtocolService

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

358 /Register/{any}

359   Extensibility elements may be used to convey additional information.

360 / Register/@{any}

361   Extensibility attributes may be used to convey additional information.

362 The following is an example registration message:

```
363  <Register>
364      <ProtocolIdentifier>
365          http://docs.oasis-open.org/ws-tx/wsat/2006/06/Volatile2PC
366      </ProtocolIdentifier>
367      <ParticipantProtocolService>
368          <wsa:Address>
369              http://Adventure456.com/participant2PCservice
370          </wsa:Address>
371          <wsa:ReferenceParameters>
372              <BetaMark> AlphaBetaGamma </BetaMark>
373          </wsa:ReferenceParameters>
374      </ParticipantProtocolService>
375  </Register>
```

Formatted: Portuguese Brazil

Formatted: English U.S.

Formatted: Portuguese Brazil

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

3.2.2 RegistrationResponse Message

376

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

378 The following pseudo schema defines this element:

```
379     <RegisterResponse ...>
380         <CoordinatorProtocolService> ... </CoordinatorProtocolService>
381         ...
382     </RegisterResponse>
```

383 /RegisterResponse/CoordinatorProtocolService

384 The Endpoint Reference that the Coordination service wants the registered participant to use for
385 the Coordination protocol.

386 /RegisterResponse/{any}

387 Extensibility elements may be used to convey additional information.

388 /RegisterResponse /@{any}

389 Extensibility attributes may be used to convey additional information.

390 The following is an example of a RegisterResponse message:

```
391     <RegisterResponse>
392       <CoordinatorProtocolService>
393         <wsa:Address>
394             http://Business456.com/mycoordinationservice/coordinator
395         </wsa:Address>
396         <wsa:ReferenceParameters>
397           <myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>
398         </wsa:ReferenceParameters>
399       </CoordinatorProtocolService>
400     </RegisterResponse>
```

401 .

4 Coordination Faults

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

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

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

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

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] A human readable explanation of the fault.

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

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

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

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

```
<S12:Envelope>
 <S12:Header>
   <wsa:Action>
       http://docs.oasis-open.org/ws-tx/wscoor/2006/06/fault
   </wsa:Action>
   <!-- Headers elided for clarity.  -->
 </S12:Header>
 <S12:Body>
  <S12:Fault>
   <S12:Code>
     <S12:Value>[Code]</S12:Value>
     <S12:Subcode>
      <S12:Value>[Subcode]</S12:Value>
     </S12:Subcode>
   </S12:Code>
   <S12:Reason>
     <S12:Text xml:lang="en">[Reason]</S12:Text>
   </S12:Reason>
   <S12:Detail>
     [Detail]
     ...
   </S12:Detail>
  </S12:Fault>
 </S12:Body>
</S12:Envelope>
```

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

```
<S11:Envelope>
 <S11:Body>
  <S11:Fault>
   <faultcode>[Subcode]</faultcode>
```

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

```
448        <faultstring xml:lang="en">[Reason]</faultstring>
449      </S11:Fault>
450    </S11:Body>
451  </S11:Envelope>
```

4.1 Invalid State

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

455 Properties:

456 [Code] Sender

457 [Subcode] wscoor:InvalidState

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

459 [Detail] unspecified

4.2 Invalid Protocol

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

464 Properties:

465 [Code] Sender

466 [Subcode] wscoor:InvalidProtocol

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

4.3 Invalid Parameters

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

471 Properties:

472 [Code] Sender

473 [Subcode] wscoor:InvalidParameters

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

4.4 Cannot Create Context

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

478 Properties:

479 [Code] Sender

480 [Subcode] wscoor:CannotCreateContext

481 [Reason] CoordinationContext could not be created.

482 [Detail] unspecified

4.5 Cannot Register Participant

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

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

486   Properties:

487   [Code] Sender

488   [Subcode] wscoor:CannotRegisterParticipant

489   [Reason] Participant could not be registered.

490   [Detail] unspecified

5 Security Model

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

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

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

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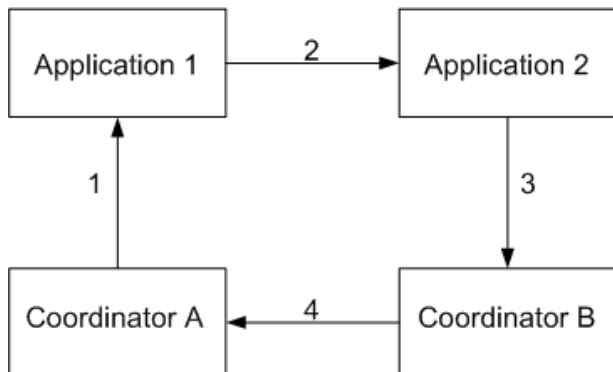
In the figure above, step 1 involves the creation and subsequent communication between the creator of the context and the coordinator A (root). It should be noted that this may be a private or local communication. Step 2 involves the delegation of the right to register with the activity using the information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

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

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

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

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

5.1 CoordinationContext Creation

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

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

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

5.2 Registration Rights Delegation

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

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

The figure below illustrates this simple key delegation model:
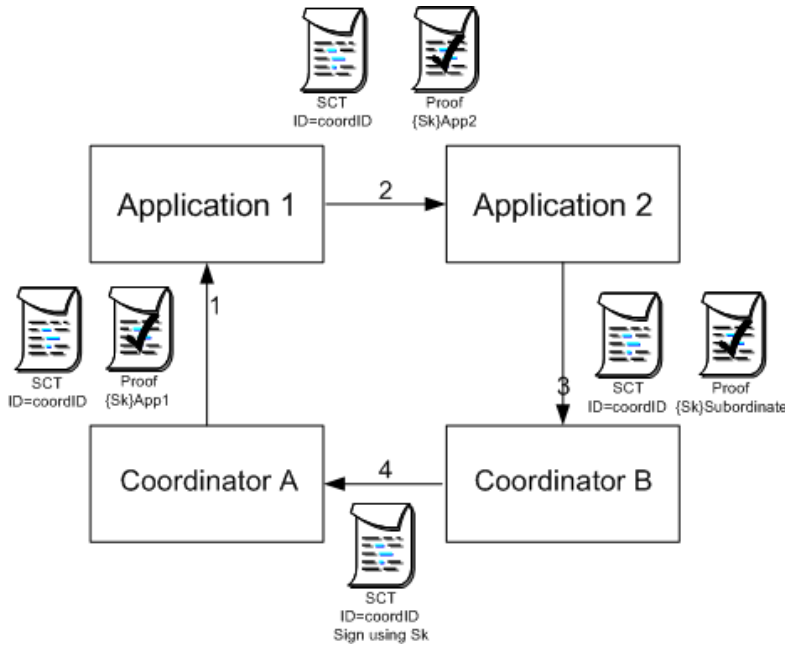
Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

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

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

559
560
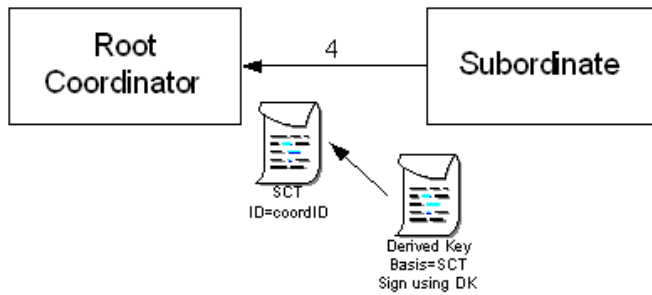
561 6 Security Considerations

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

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

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

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

576 - Using a derived key sequence and switch "generations"

577 - Closing and re-establishing a security context

578 - Exchanging new secrets between the parties

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

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

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

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

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

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

596 - **Authentication** – Authentication is established using the mechanisms described in WS-Security
597   **[WSSec]** and WS-Trust **[WSTrust]**. Each message is authenticated using the mechanisms
598   described in WS-Security **[WSSec]**.

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

602 - **Availability** – Many services are subject to a variety of availability attacks. Replay is a common
603   attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other
604   attacks, such as network-level denial of service attacks are harder to avoid and are outside the

Deleted: os	
Deleted: 1	
Deleted: Apri	
Deleted: l	

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

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

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

7 Use of WS-Addressing Headers

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

- Request messages: **CreateCoordinationContext** and **Register**.
- Reply messages: **CreateCoordinationContextResponse** and **RegisterResponse** and the protocol faults defined in Section 4 of this specification.

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

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

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

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

Deleted: os

Deleted: 1

Deleted: Apri

Deleted: l

8 Glossary

The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

Appendix A. Acknowledgements

650

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

657

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

663

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

665

666 **Participants:**

667

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

693

Deleted: os
Deleted: 1
Deleted: Apri
Deleted: l