Web Services Coordination Framework Specification (WS-CF)

Committee Draft 0.2

Version created 1 August 2005

Deleted: 1 August 200524 May 2005

Editors

Mark Little (mark.little@arjuna.com)

Eric Newcomer (eric.newcomer@iona.com)

Greg Pavlik (greg.pavlik@oracle.com)

Abstract

OASIS Web Services Composite Application Framework (WS-CAF) provides a set of modular and composable service definitions to facilitate the construction of applications that combine multiple services together in composite applications. The fundamental capability offered by the WS-Coordination Framework specification is the ability to register a web service as a participant in some kind of domain specific function. An example scenario may be to register with a publication-subscription topic to receive a stream of messages asynchronously. While it is expected that the vast majority of protocols will involve some form of signaling to registered services via SOAP messages, this signaling is not a part of the model itself. Monitoring protocols, for example, may express interest in participation in some interaction semantic without any subsequent signaling to registered services; messaging protocols may use an optimized channel based on a native MOM protocol for message distribution.

WS-Context provides a late binding session model for the web services environment. SOAP messages that are to be processed within the scope of an activity contain Context headers, uniquely identifying a single activity. WS-Coordination Framework extends the session model for protocols that require group membership paradigms by defining a Registration Context Type. The Registration Context Type extends the basic context type and provides a Web service reference to a Registration Service. Registration in the context of an activity adds the registered service to an activity group. Membership in the group may be used to drive some group specific protocol (e.g. data replication) over the lifetime of the activity group or may be used to coordinate signals associated with a termination protocol (e.g., two phase commit). The purpose and semantics of activity group membership are protocol specific.

Coordination is a requirement present in a variety of different aspects of distributed applications. For instance, workflow, atomic transactions, caching and replication, security, auctioning, and business-to-business activities all require some level of what may be collectively referred to as “coordination.” For example, coordination of multiple Web services in choreography may be required to ensure the correct result of a series of operations comprising a single business transaction. Coordination protocols may be layered on WS-Coordination Framework.

Deleted: Web Services Coordination Framework Specification (WS-CF)¶ Editors draft version 0.4¶ 3 May 2005¶

Deleted: -

Deleted: is

Deleted: ¶ ¶

Table of contents

Deleted: 5

Deleted: 5

Deleted: 9

Deleted: 10

Deleted: 11

Deleted: 11

Deleted: 12

Deleted: 14

Deleted: 15

Deleted: 15

Deleted: 15

Deleted: 16

Deleted: 16

Deleted: 16

Deleted: 16

Deleted: 18

Deleted: 19

Deleted: 20

Deleted: 21

Deleted: 1 . Note on terminology . 4¶
1.1 Namespace . 4¶
1.1.1 Prefix Namespace . 4¶
1.2 Referencing Specifications . 4¶
2 . Introduction . 5¶
3 . WS-CF architecture . 6¶
3.1 Overview . 6¶
3.2 Invocation of Service Operations . 6¶
3.3 Relationship to WSDL . 7¶
3.4 Referencing and addressing conventions . 8¶
4 . WS-CF components . 10¶
4.1 Participant Service . 10¶
4.2 Registration Service . 11¶
4.2.1 Service-to-Registration interactions . 11¶
addParticipant . 11¶
removeParticipant . 12¶
recoverParticipant . 12¶
recoverRegistration . 13¶
getStatus . 13¶
4.2.2 Registration Context . 15¶
4.3 Interposition . 16¶
5 . References . 17¶

1 Note on terminology

The keywords "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 [2].

Namespace URIs of the general form http://example.org and http://example.com represents some application-dependent or context-dependent URI as defined in RFC 2396 [3].

1.1 Namespace

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

```
http://docs.oasis-open.org/wscaf/2005/07/wscf
```

1.1.1 Prefix Namespace

Prefix	Namespace
wscf	**http://docs.oasis-open.org/wscaf/2005/07/wscf**
wsctx	**http://docs.oasis-open.org/wscaf/2005/06/wsctx**
ref	**http://docs.oasisopen.org/wsrm/2004/06/reference-1.1**
wsdl	**http://schemas.xmlsoap.org/wsdl/**
xs	**http://www.w3.org/2001/XMLSchema**
wsu	**http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd**
tns	**http://docs.oasis-open.org/wscaf/2005/07/wscf**

1.2 Referencing Specifications

One or more other specifications, such as (but not limited to) WS-ACID may reference the WS-CF specification. The usage of optional items in WS-CF is typically determined by the requirements of such as referencing specification.

A referencing specification generally defines the protocol types based on WS-CF. Any application that uses WS-CF must also decide what optional features are required. For the purpose of this document, the term *referencing specification* covers both formal specifications and more general applications that use WS-CF.

1.3 Precedence of schema and WSDL

Throughout this specification, WSDL and schema elements may be used for illustrative or convenience purposes. However, in a situation where those elements within this document differ from the separate WS-CF WSDL or schema files, it is those files that have precedence and not this specification.

Deleted: Namespace URIs of the general form "some-URI" represents some application-dependent or context-dependent URI as defined in RFC 2396 [3].¶

Deleted: 2

Deleted: 2

Deleted: 4

Deleted: 9

Deleted: d

Deleted: targetNamespace

Deleted: TXM

Formatted: Bullets and Numbering

2 Architecture

Many protocols in distributed systems require software agents to perform a registration function to participate in the protocol. Examples of protocols that require explicit registration functions include notifications, transactions, virtually synchronous replica models based on group membership paradigms, and security. WS-Coordination Framework provides a WSDL interface for registering Web services as participants in arbitrary protocols. This is supported through the Registration Service.

Context information can flow implicitly (transparently to the application) within normal messages sent to the participants, or it may be an explicit action on behalf of the client/service. This information is specific to the type of activity being performed and may identify registration endpoints, the other participants in an Activity, recovery information in the event of a failure, etc. Furthermore, it may be required that additional application specific context information flow to these participants or the services which use them. WS-Coordination Framework introduces a **wscf:RegistrationContextType** that builds on the context type defined in WS-Context to provide additional information required to enlist as a participant in an activity. Applications may use the registration context type by extension to define collections of services called “activity groups”. WS-Coordination Framework provides support for protocols that depend on group membership paradigms, such as coordination and security.

Coordination is an integral part of any distributed system, but there is no single type of coordination protocol that can suffice for all composite applications. This specification defines a common Web Services Coordination Framework (WS-CF) that allows users and services to tie into it and customize it for each service or application. A suitably designed coordination framework should provide enough flexibility and extensibility to its users that allow it to be tailored, statically or dynamically, to fit any requirement.

This framework builds upon WS-Context and supports WS-ACID, WS-LRA and WS-BP, as well as other Web Service standards in the area of choreography, workflow and transactions. In the case of transactions, for example, unlike other attempts that are solutions to one specific problem area and are therefore not applicable to others, different extended transaction models can be relatively easily developed to suit specific domains, and interoperability across transaction protocols supported.

The following sections outline the architecture of WS-CF, describing the components that implementations provide and those that are required from users.

2.1 Overview

WS-CF builds upon the activity concept defined in the WS-Context specification [ref] by narrowing the notion of an activity to that of an *activity group*: such a group contains members (participants) that will be driven through the same protocol. WS-CF says nothing about specifics of such coordination protocols and when or where participants may join and leave: this is left up to referencing specifications to define.

The group membership facilities are used to build and manage relationships between services. For example, an activity group can be used as the basic definition of a participant set in a coordination protocol. The group paradigm is central to coordination, whether it is coordinating the outcome of distributed transactions, security domains, replica consistency, cache coherency etc. Because WS-CF is meant to support a range of coordination protocols, each possessing different protocol messages and potentially different coordinator interfaces, WS-CF does not define how or when coordination occurs. This is left to referencing specifications.

The activity group is tied to an underlying WS-Context activity such that their lifetimes coincide. Web Services that wish to join or leave the group make use of the Registration Service; the membership of the group may also be obtained from the Registration Service.

Deleted: Introduction

Deleted: TXM

Deleted: <#>WS-CF architecture¶

Formatted: Bullets and Numbering

- Specific implementations of the Registration Service MAY impose restrictions on how and when group membership changes may occur; these are outside the scope of the WS-CF specification. In addition, some uses of group membership MAY place constraints on consistent views of group membership, particularly in the presence of member failures. Ensuring this kind of view membership consistency is left to referencing specifications.

Deleted: may

Deleted: may

The main components involved in using and defining the WS-CF are:

- A Registration service, which provides an interface for the registration of participants within a specific protocol.
- A Participant service, which defines the operation or operations that are performed as part of the protocol. It is possible to register participants that have no protocol specific callback operations.
- A Registration Context Type, which allows participants to join an activity group.

This specification allows group membership to be managed with reference to a specific context; the relationship between different contexts is defined by the WS-Context specification; specific protocols based on activity groups may support subgroups and interposed activities. Activity groups are particularly useful for structuring relationships in the kinds of coordination protocols found in transaction systems and data replication/consistency protocols for clustered services.

WS-CF supports the notion of *interposition*: where a Participant Service that is enlisted with a Registration Service also behaves as a Registration Service to other Participant Services. In this way, WS-CF supports the building of graphs and trees by the addition of participants to an activity structure that are themselves registration endpoints.

The technique of interposition uses proxies (or subordinates). Each domain that imports a WS-CF context MAY create a subordinate registration service that enrolls with the imported registration service as though it were a participant. This specification does not prescribe how and when this may occur. Interposition then requires the importing domain to use a different context when communicating with services and participants that are required to register with the subordinate registration service, as shown in Figure 33.

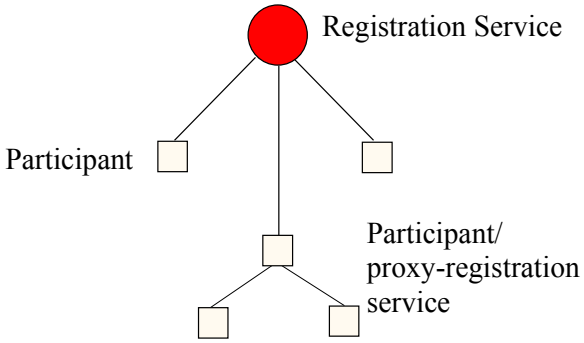

Figure 1, Participant coordinator.

Deleted: *1*

This specification does not define what are allowable forms of graphs that may be created using interposition. Such definitions are the responsibility of referencing specifications.

Deleted: ¶

2.2 Invocation of Service Operations

Formatted: Bullets and Numbering

How application services are invoked is outside the scope of this specification: they MAY use synchronous or asynchronous message passing.

Irrespective of how remote invocations occur, context information related to the sender's activity needs to be referenced or propagated. This specification determines the format of the context, how it is referenced, and how a context may be created.

In order to support both synchronous and asynchronous interactions, the components are described in terms of the behavior and the interactions that occur between them. All interactions are described in terms of correlated messages, which a referencing specification MAY abstract at a higher level into request/response pairs.

Faults and errors that may occur when a service is invoked are communicated back to other Web services in the activity via SOAP messages that are part of the standard protocol. To achieve this, the fault mechanism of the underlying SOAP-based transport is used. For example, if an operation fails because no activity is present when one is required, then the callback interface will receive a SOAP fault including type of the fault and additional implementation specific information items supported the SOAP fault definition. WS-Coordination Framework specific fault types are described for each operation. A fault type is communicated as an XML QName; the prefix consists of the WS-Coordination Framework namespace and the local part is the fault name listed in the operation description.

> Note, a transientFault message is produced when the implementation finds it cannot successfully execute the requested operation at that time from some *temporary* reason. This reason may be implementation or referencing specification specific. A receiver of a transientFault is free to retry the operation which originally generated it on the assumption that eventually a different response will be produced. Sub-types of transientFault MAY be further defined using the fault model described which can allow for the communication of more specific information on the type of fault.

As long as implementations ensure that the on-the-wire message formats are compliant with those defined in this specification, how the end-points are implemented and how they expose the various operations (e.g., via WSDL [1]) is not mandated by this specification. However, a normative WSDL binding is provided by default in this specification.

> Note, this specification does not assume that a reliable message delivery mechanism has to be used for message interactions. As such, it MAY be implementation dependant as to what action is taken if a message is not delivered or no response is received.

2.3 Relationship to WSDL

Where WSDL is used in this specification it uses one-way messages with callbacks. This is the normative style. Other binding styles are possible (perhaps defined by referencing specifications), although they may have different acknowledgment styles and delivery mechanisms. It is beyond the scope of WS-Coordination Framework to define these styles.

> Note, conformant implementations MUST support the normative WSDL defined in the specification where those respective interfaces are required. WSDL for optional components in the specification is REQUIRED only in the cases where the respective components are supported.

For clarity WSDL is shown in an abbreviated form in the main body of the document: only portTypes are illustrated; a default binding to SOAP 1.1-over-HTTP is also assumed as per [1].

Deleted: ntext

Deleted: ntext

Deleted: How application services are invoked is outside the scope of this specification; however, context information related to the sender's activity needs to be referenced and/or propagated. ¶
Irrespective of how remote invocations occur, context information related to the sender's activity needs to be referenced or propagated. This specification determines the format of the context, how it is referenced, and how a context may be created.¶
In order to support both synchronous and asynchronous interactions, the components are described in terms of the behavior and the interactions that occur between them. All interactions are described in terms of correlated messages, which a referencing specification MAY abstract at a higher level into request/response pairs. ¶
Faults and errors that may occur when a service is invoked are communicated back to other Web services in the activity via SOAP messages that are part of the standard protocol. The fault mechanism of the underlying SOAP-based transport isn't used. For example, if an operation fails because no activity is present when one is required, then it will be valid for the InvalidContextFault message to be received by the response service. To accommodate other errors or faults, all response service signatures have a generalFault operation as well as a transientFault operation.¶

Formatted: Bullets and Numbering

2.4 Referencing and addressing conventions

There are multiple mechanisms for addressing messages and referencing Web services currently proposed by the Web services community. This specification defers the rules for addressing SOAP messages to existing specifications; the addressing information is assumed to be placed in SOAP headers and respect the normative rules required by existing specifications.

However, the Coordination Framework message set requires an interoperable mechanism for referencing Web Services. For example, context structures may reference the service that is used to manage the content of the context. To support this requirement, WS-CAF has adopted an open content model for service references as defined by the Web Services Reliable Messaging Technical Committee [5]. The schema is defined in [6][7] and is shown in Figure 221

```
<xs:complexType name="ServiceRefType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax"/>
  </xs:sequence>
  <xs:attribute name="reference-scheme" type="xsd:anyURI"
      use="optional"/>
</xs:complexType>
```

Figure 2, service-ref Element

The **ServiceRefType** is extended by elements of the context structure as shown in Figure 3322

```
<xs:element name="context-manager" type="ref:ServiceRefType"/>
```

Figure 3, ServiceRefType example.

Within the **ServiceRefType**, the reference-scheme is the namespace URI for the referenced addressing specification. For example, the value for WSRef defined in the WS-MessageDelivery specification [4] would be http://www.w3.org/2004/04/ws-messagedelivery. The value for WSRef defined in the WS-Addressing specification [8] would be http://schemas.xmlsoap.org/ws/2004/08/addressing. The reference scheme is optional and need only be used if the namespace URI of the QName of the Web service reference cannot be used to unambiguously identify the addressing specification in which it is defined.

Messages sent to referenced services MUST use the addressing scheme defined by the specification indicated by the value of the reference-scheme element if present. Otherwise, the namespace URI associated with the Web service reference element MUST be used to determine the required addressing scheme.

> Note, it is assumed that the addressing mechanism used by a given implementation supports a reply-to or sender field on each received message so that any required responses can be sent to a suitable response endpoint. This specification requires such support and does not define how responses are handled.

To preserve interoperability in deployments that contain multiple addressing schemes, there are no restrictions on a system, beyond those of the composite services themselves. However, it is RECOMMENDED where possible that composite applications confine themselves to the use of single addressing and reference model.

Because the prescriptive interaction pattern used by WS-Coordination Framework is based on one-way messages with callbacks, it is possible that an endpoint may receive an unsolicited or unexpected message. The recipient is free to do whatever it wants with such messages.

Formatted: Bullets and Numbering

Deleted: Figure 2Figure 1Figure 1

Inserted: Figure 2Figure 1

Deleted: d

Deleted: d

Deleted: d

Deleted: d

Deleted: d

Deleted: d

Deleted: <xsd:schema targetNamespace="http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.1">¶

<xsd:complexType name="ServiceRefType">¶

<xsd:sequence>¶

<xsd:any namespace="##other" processContents="lax" /> ¶

</xsd:sequence>¶

<xsd:attribute name="reference-scheme" type="xsd:anyURI" use="optional" /> ¶

</xsd:complexType>¶

Deleted: 2

Inserted: 2

Deleted: 1

Deleted: Figure 3Figure 2Figure 2

Inserted: Figure 3Figure 2

Deleted: d

Deleted: 322

Inserted: 32

Deleted: A service that requires a service reference element MUST use the mustUnderstand attribute for the SOAP header element within which it is enclosed and MUST return a mustUnderstand SOAP fault if the reference element is ... [1]

3 WS-CF components

WS-CF provides three components that may be used to build collaborative protocols and complex composite applications: the Participant service, the Registration service, and the Registration Context Type. The components are described in terms of their behavior and the interactions that occur between them. All interactions are described in terms of message exchanges, which an implementation may abstract at a higher level into request/response pairs or RPCs, for example. Like WS-Context, the components are organized in a hierarchical relationship, where individual components may be used without reference to higher-level constructs that build on them. For example, the Registration and Participant services can be used without reference to an activity group.

3.1 Participant Service

Many distributed protocols require software agents to enlist as participants within a protocol to achieve an application visible semantic. For example, participants may enlist in a transaction protocol in order to receive messages at coordination points defined by the protocol.

A Participant will use coordination messages in a manner specific to the protocol and (optionally) return a result of it having done so. For example, upon receipt of a specific message, a Participant could commit any modifications to a database when it receives one type of message, or undo them if it receives another type. In some cases (e.g., monitoring protocols) Participants may register for protocols that do not include any subsequent signaling. In other cases, such as publish-and-subscribe scenarios, Participants may register for a stream of messages that have no fixed semantic content with respect to the protocol itself. In general, rules governing the subsequent interaction between Participants and Registration endpoints are defined by specifications that make use of WS-CF. As such, there is no WSDL interface defined for the Participant Service; it is an abstract entity that is given concrete representation by referencing specifications and is only discussed within the scope of this specification for clarity of the overall model concept.

3.2 Registration Service

In order to become a Participant in a protocol, a service must first enlist with a Registration service. The protocol that the Registration implementation uses will depend upon the type of activity, application or service using the Registration service. For example, if Saga model is in use then a compensation message may be required to be sent to Participants if a failure has happened, whereas a coordinator for a strict transactional model may be required to send a message informing participants to rollback.

How a Registration service for a specific protocol(s) is located or associated with the Context Service is out of scope of this specification. A Registration service MAY identify the type of protocol it supports using deployment specific mechanisms.

A Registration Service implementation provides support for the Registering Services to enlist Participant services with a specific protocol semantic. Operations on the Registration service MAY be implicitly associated with a Registration Context Type, i.e., it is propagated to the Registration service in order to identify which activity group the Participant is interested in joining. Services requiring protocols that rely explicitly on group membership like transactions or data replication will require that the Registration service MUST be invoked with a subtype of the Registration context.

In the following sections we shall discuss the different Registration service interactions and their associated message exchanges.

Formatted: Bullets and Numbering

Deleted: c

Deleted: <#>Interposition¶
WS-CF supports the notion of *interposition*: where a Participant Service that is enlisted with a Registration Service also behaves as a Registration Service to other Participant Services. In this way, WS-CF supports the building of graphs and trees by the addition of participants to an activity structure that are themselves registration endpoints.¶
The technique of interposition uses proxies (or subordinates). Each domain that imports a WS-CF context MAY create a subordinate registration service that enrolls with the imported registration service as though it were a participant. This specification does not prescribe how and when this may occur. Interposition then requires the importing domain to use a different context when communicating with services and participants that are required to register with the subordinate registration service, as shown in Figure 3Figure 3.¶
... [2]

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Deleted: A Registration Service implementation provides support for the Registering Services to enlist Participant services with a specific protocol semantic. Operations on the Registration service MAY be implicitly associated with a Registration context, i.e., it is propagated to the Registration service ... [3]

3.2.1 Service-to-Registration interactions

These interactions define how a service (the *Registering Service*) may enlist or delist a Participant (Service) with the Registration Service. The message exchanges are illustrated in Figure 4544. They are factored into two different roles:

- Registration Service: this accepts the addParticipant, removeParticipant, replaceParticipant, registrationReplaced, getParticipants and getStatus messages. All messages contain the Registering Service endpoint for callback messages, although it is OPTIONAL as to whether the Registration Service remembers these beyond a specific interaction.
- Registering Service: this accepts the participantAdded, participantRemoved, participantReplaced, participantList, status, replaceRegistration messages.

addParticipant

This message is sent to the coordinator in order to register the specified Participant with the protocol supported by the Registration service. A valid **wscf:RegistrationContext** MUST accompany this message and the participant will be added to the activity group identified in the context. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required. If an invalid **wscf:RegistrationContext** is used then an appropriate WS-Context error message MUST be returned.

The protocol based on the RegistrationContextType may support multiple sub-protocols (e.g., synchronizations that are executed prior to and after a two-phase commit protocol); in order to define with which protocols to enlist the participant, the list of **wscf:protocolType** URIs may be propagated in the message. The Registration Service MUST ensure that all protocols specified are supported before any registration happened. If some of the protocols are not supported by the Registration service then no registration occurs and the **wscf:InvalidProtocol** error message MUST be sent to the Registering Service indicating which protocols were at fault.

Upon success, the Registration service calls back to the Registering Service with the **wscf:participantAdded** message. Implementations MAY include in this message the unique OPTIONAL endpoint reference for the Participant to use for further interactions. How and when this endpoint reference should be used is outside the scope of this specification and is left to referencing specifications to determine. For example, it may be used by the Participant to send protocol specific coordination signals.

A referencing specification MAY decide to send the **wsctx:InvalidState** error message, for example if the activity has begun completion, or has already completed when this operation is attempted.

The termination of the activity group is triggered by the completion of the WS-Context service activity. The relationship between activity groups and participant services is undefined following the termination of an activity group.

If the same participant has been enrolled with the Registration service more than once and the referencing specification does not allow this, then the **wscf:DuplicateParticipant** error message is sent to the **ServiceRespondant**. How the registration of the same participant multiple times is dealt with at the protocol level is outside the scope of this specification and is left to referencing specifications to define, as the rules governing the protocol are defined by a referencing specification

removeParticipant

This message causes the Registration service to delist the specified Participant. A valid **wscf:RegistrationContext** MUST accompany this message to identify the activity group from which the participant should be removed. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic

Formatted: Bullets and Numbering

Deleted: Figure 5Figure 4Figure 4

Inserted: Figure 5Figure 4

Deleted: recoverParticipant

Deleted: registrationRecovered

Deleted: covered

Deleted: recoverRegistration

Deleted: ,

Deleted: generalFault, wrongState, duplicateParticipant, invalidProtocol, invalidParticipant, and participantNotFound

Deleted: i

Deleted: ,

Deleted: including

Deleted: wrong

Deleted: A

Deleted: d

Deleted: RegistrationContext

reference values is required. If successful, the ParticipantRemoved message is sent to the invoker.

If the Participant has not previously been registered with the Registration service for the specified activity group, then it will send the **wscf:ParticipantNotFound** error message to the Registering Service.

Removal of a participant need not be supported by the specific protocol and may also be dependant upon where in the protocol the system is as to whether a referencing specification will allow the participant to be removed. The rules governing removal of participants from participation in a protocol or activity group are governed by referencing specifications. A referencing specification MAY decide to send the **wsctx:InvalidState** error message if removal is disallowed; for example, the activity has begun completion, or has already completed when this operation is attempted.

In addition, some protocols may allow for Registration service to autonomously delist Participant services. In this case, the Registration Service will send an unsolicited ParticipantRemoved message to the service that was responsible for enlisting the Participant.

replaceParticipant

This operation is used by a participant that has previously successfully enlisted with a Registration service: when the Participant fails and subsequently recovers it may not be able to recover at the same address that it used to enlist with the Registration service. The replaceParticipant operation allows the participant to inform the Registration service that it has moved from the original address to a new address. It may also be used to start recovery operations by the protocol engine.

A valid **wscf:RegistrationContext** MUST accompany this message in order to identify the group in which the failed participant previously existed. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required.

If successful, the participantReplaced message is sent to the invoker. If the recovery handshake occurs in the context of an activity, the message also contains the current status of the activity. This status may be used by the recovering participant to perform local recovery operations, although this will depend upon the protocol in use. For example, if the participant was enrolled in a presumed-abort transaction protocol and recovery indicated that the transaction no longer exists, then the participant can cancel any work it may be controlling.

If the coordinator cannot be located, then the **wsctx:UnknownContext** error message is sent back.

If the status of the coordinator is such that recovery is not allowed at this time, the **wsctx:InvalidState** error message is sent to the Registering Service by the coordinator.

If the Registration Service cannot deal with recovery of the participant for a temporary reason, the **wscf:TransientFault** message is sent and the receiver MAY try again.

getParticipants

This operation returns the list of participants that have been enrolled with the activity group. A valid **wscf:RegistrationContext** MUST accompany this message. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required.

If successful, the participantList message is sent to the Registering Service.

A referencing specification MAY decide to send the **wsctx:InvalidState** error message if the Activity has begun completion, or has already completed when this operation is attempted.

Deleted: p
Deleted: wrong
Deleted: A
Deleted: recoverParticipant
Deleted: recoverParticipant
Deleted: RegistrationContext
Deleted:
Deleted: participantRecovered
Deleted: invalidActivityFault
Deleted: wrong
Deleted: t
Deleted: recoverRegistration
Deleted: replaceRegistration¶
This operation on the Registering Service MAY be used by a recovered Registration Service to indicate that it has recovered on a new endpoint address. When a Registration Service fails and subsequently recovers it may not be able to recover at the same address that prior ... [4]
Inserted: replaceRegi ... [5]
Deleted: recoverRegistration
Deleted: replaceRegist ... [6]
Deleted: ransientFault ... [7]
Inserted: replaceRegistration
Inserted: wscf:Regist ... [8]
Deleted: RegistrationContext
Deleted: information ... [9]
Inserted: wscf:Regis ... [10]
Deleted: RegistrationContext
Deleted: MUST accor ... [11]
Deleted: registrationR ... [12]
Deleted: registrationR ... [13]
Deleted: u
Inserted: registrationI ... [14]
Inserted: wscf:U
Deleted: nknownSer ... [15]
Deleted: t
Inserted: error
Inserted: wscf:T
Inserted: error
Inserted: ¶ ... [16]

The termination of the activity group is triggered by the completion of the WS-Context service activity. The relationship between activity groups and participant services is undefined following the termination of an activity group.

getStatus

The status of the activity group may be obtained by sending the getStatus message to the recovery coordinator. A valid **wscf:RegistrationContext** MUST accompany this message. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required.

The status, which may be one of the status values specified by the Context Service, or may be specific to the protocol, identified by its QName, is returned to the invoker via the status message. GetStatus will return the same Status value that is returned by the getStatus operation on the Context Service, assuming the queries occur at the same point in the activity lifecycle.

replaceRegistration

This operation on the Registering Service MAY be used by a recovered Registration Service to indicate that it has recovered on a new endpoint address. When a Registration Service fails and subsequently recovers it may not be able to recover at the same address that prior Registering Services used to enlist with the Registration service. This OPTIONAL operation allows the Registration Service to inform Registering Services that it has moved from the original address to a new address. It may also be used to start recovery operations by the protocol engine.

The use of replaceRegistration SHOULD only be attempted when the Registration Service has failed and recovered on another endpoint because to do otherwise MAY result in continued use of stale **wscf:RegistrationContext** information elsewhere in the application; the context refers to the old endpoint address for the Registration Service.

A valid **wscf:RegistrationContext** MUST accompany this message. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required.

If successful, the registrationReplaced message is sent to the Registration Service. If the recovery handshake occurs in the context of an activity, the message also contains the current status of the activity. This status may be used by recipients to perform local recovery operations, although this will depend upon the protocol in use

If the Registering Service cannot be located, then the **wscf:UnknownService** error message is sent back.

If the Registering Service cannot deal with recovery of the Registration Service for a temporary reason, the **wscf:TransientFault** error message is sent and the receiver MAY try again.

Deleted:

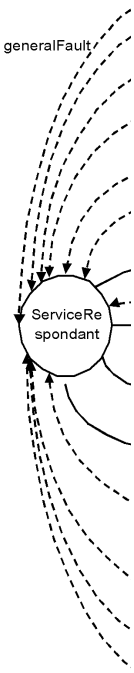

Figure 4, Service-to-coordinator interactions.

The Registration Service and Registering Service roles are elucidated in WSDL form in Figure 5655

```
<wsdl:portType name="RegistrationServicePortType">
  <wsdl:operation name="addParticipant">
    <wsdl:input message="tns:AddParticipantMessage"/>
  </wsdl:operation>
  <wsdl:operation name="removeParticipant">
    <wsdl:input message="tns:RemoveParticipantMessage"/>
  </wsdl:operation>
  <wsdl:operation name="replaceParticipant">
    <wsdl:input message="tns:ReplaceParticipantMessage"/>
  </wsdl:operation>
  <wsdl:operation name="registrationReplaced">
```

Deleted: 544

Inserted: 54

Comment: All of this needs changing.

Deleted: Figure 6Figure 5Figure 5

Inserted: Figure 6Figure 5

Deleted: recoverParticipant

Deleted: RecoverParticipantMessage

Deleted: registrationRecovered

```
    <wsdl:input message="tns:RegistrationReplacedMessage"/>
  </wsdl:operation>
  <wsdl:operation name="getStatus">
    <wsdl:input message="tns:GetStatusMessage"/>
  </wsdl:operation>
  <wsdl:operation name="getParticipants">
    <wsdl:input message="tns:GetParticipantsMessage"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="RegisteringServicePortType">
  <wsdl:operation name="participantAdded">
    <wsdl:input message="tns:ParticipantAddedMessage"/>
  </wsdl:operation>
  <wsdl:operation name="participantRemoved">
    <wsdl:input message="tns:ParticipantReplacedMessage"/>
  </wsdl:operation>
  <wsdl:operation name="participantReplaced">
    <wsdl:input message="tns:ParticipantRecoveredMessage"/>
  </wsdl:operation>
  <wsdl:operation name="replaceRegistration">
    <wsdl:input message="tns:ReplaceRegistrationMessage"/>
  </wsdl:operation>
  <wsdl:operation name="status">
    <wsdl:input message="tns:StatusMessage"/>
  </wsdl:operation>
  <wsdl:operation name="participantList">
    <wsdl:input message="tns:ParticipantListMessage"/>
  </wsdl:operation>
</wsdl:portType>
```

Figure 5, WSDL portType Declarations for Registration Service and Registering Service Roles.

3.2.2 Registration Context Type

In order to support registration in activity groups it is necessary for the participants to be made aware of the Registration Service associated with the activity group via some mechanism. In a distributed environment, this requires information about the Registration service (essentially its network endpoint) to be available to remote participants. WS-Context provides mechanisms for propagating basic activity context information between services. The information contained within this basic activity context is the unique activity identity and optional information associated with demarcation of the activity lifecycle and management of the context. WS-Coordination Framework extends the **wsctx:ContextType** defined in WS-Context to allow services to register as Participants in an activity. The **wscf:RegsitrationContextType** is shown in Figure 5.

```
<xs:complexType name="RegistrationContextType">
  <xs:complexContent>
    <xs:extension base="wsctx:ContextType">
      <xs:sequence>
      <xs:element name="registration-service" type="ref:ServiceRefType"
        minOccurs="1"/>
      <xs:element name="sub-protocol" type="xs:anyURI"
        maxOccurs="unbounded"/>
      <xs:element name="participant-service" type="ref:ServiceRefType"
        maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Deleted: RegistrationRecoveredMessage

Deleted: ParticipantRemovedMessage

Deleted: participantRecovered

Deleted: recoverRegistration

Deleted: RecoverRegistrationMessage

Deleted:
<wsdl:operation name="generalFault">¶
<wsdl:input message="tns:GeneralFaultMessage"/>¶
</wsdl:operation>¶
<wsdl:operation name="wrongState">¶
<wsdl:input message="asw:WrongStateFaultMessage"/>¶
</wsdl:operation>¶
<wsdl:operation name="duplicateParticipant">¶
<wsdl:input message="tns:DuplicateParticipantFaultMessage"/>¶
</wsdl:operation>¶
<wsdl:operation name="invalidProtocol">¶
<wsdl:input message="tns:InvalidProtocolFaultMessage"/>¶
</wsdl:operation>¶
<wsdl:operation name="invalidParticipant">¶
<wsdl:input message="tns:InvalidParticipantMessage"/>¶
</wsdl:operation>¶
<wsdl:operation name="participantNotFound">¶
<wsdl:input message="tns:ParticipantNotFoundFaultMessage"/>¶
</wsdl:operation>¶

Deleted: *655*

Inserted: *65*

Formatted: Bullets and Numbering

Deleted: any

Figure 6, WS-CF RegistrationContextType derives from the WS-Context ContextType.

The Registration Context Type contains the following elements in addition to the WS-Context **wsctx:ContextType** structure:

- A service reference to a Registration service. This enables Participant services to be enlisted or delisted in an activity group.
- A list of zero or more sub-protocol URIs that are used to specify the sub-protocols in which a service may register as a Participant. For example, a transaction protocol may support synchronization and two phase commit subprotocols.
- A list of zero or more service references indicating the list of services registered as Participants in the activity group.

Referencing specifications define contexts derived from the RegistrationContextType. As per WS-Context, the QName of the derived context represents the protocol type for the activity. The XML below shows an example of a subtyped Registration context.

```
<example:cfContext
  xmlns="http://docs.oasis-open.org/wscaf/2005/06/wsctx"
  xmlns:wscf="http://docs.oasis-open.org/wscaf/2005/07/wscf"
  xmlns:example="http://example.com/cf/"
  expiresAt="2005-07-26T22:50:00+01:00">
    <context-identifier>
      http://example.org/abcdef:012345
    </context-identifier>
    <context-service>
      <example:address>
        http://example.org/wscf/service
      </example:address>
    </context-service>
    <parent-context expiresAt="2005-07-27T22:50:00+01:00">
      <context-identifier>
        http://example.org/5e4f2218b
      </context-identifier>
      <context-service>
        <example:address>
          http://example.org/wsctx/service
        </example:address>
      </context-service>
    </parent-context>
    <wscf:registration-service>
      <example:address>
        http://example.org/wscf/RegistrationService
      </example:address>
    </wscf:registration-service>
</example:cfContext>
```

3.2.3 WS-CF faults

This section defines well-known error codes to be used in conjunction with an underlying fault handling mechanism.

Invalid Protocol

This fault is be sent by the Registration Service if an attempt is made to register a participant with a protocol that is not supported. This is an unrecoverable condition.

The qualified name of the fault code is:

```
wscf:InvalidProtocol
```

Deleted: 766
Inserted: 76
Deleted: c
Deleted: s
Deleted: ¶
Deleted: The XML below shows an example of a Registration context.¶
Deleted: context
Deleted:
Deleted: 4
Deleted: 9
Deleted: tx
Deleted:
Deleted: 4
Deleted:
Deleted:
Field Code Changed
Deleted: docs.oasis ... [17]
Deleted:
Deleted:
Field Code Changed
Deleted: tx
Deleted:
Deleted: ¶
Deleted: <ty ... [18]
Deleted:
Deleted: 4
Deleted:
Deleted:
Deleted:
Deleted:
Deleted:
Deleted: . ... [19]
Deleted:
Deleted:
Deleted:
Deleted:
Deleted: c
Formatted: Bullets and Numbering ... [20]

Duplicate Participant

This fault is be sent by the Registration Service if an attempt is made to register a participant multiple times and the referencing specification does not allow this.

The qualified name of the fault code is:

```
wscf:DuplicateParticipant
```

Participant Not Found

This fault is be sent by the Registration Service if an attempt is made to remove a participant that has not been registered.

The qualified name of the fault code is:

```
wscf:ParticipantNotFound
```

Transient Fault

This fault is sent if an attempt is made to replace an endpoint when recovery is not currently allowed. Retrying the operation SHOULD eventually result in success.

The qualified name of the fault code is:

```
wscf:TransientFault
```

Unknown Service

This fault is sent if an attempt is made to replace a Registration Service endpoint and the recipient does not recognise the Registration Service to be replaced.

The qualified name of the fault code is:

```
wscf:UnknownService
```

Formatted: Bullets and Numbering

3.2.4 Message exchanges

The WS-CAF protocol family is defined in WSDL, with associated schemas. All the WSDL has a common pattern of defining paired port-types, such that one port-type is effectively the requestor, the other the responder for some set of request-response operations.

portType for an initiator ("client" for the operation pair) will expose the responses of the "request/response" as input operations (and should expose the requests as output messages); the responder (service-side) only exposes the request operations as input operations (and should expose the responses as output messages).

Each "response" is shown on the same line as the "request" that invokes it. Where there are a number of responses to a "request", these are shown on successive lines. The initiator portTypes typically include various fault and error operations.

Initiator (as receiver of response)	**Responder**	**"requests"**	**"responses"**
RegisteringService	**RegistrationService**	addParticipant	participantAdded wsctx:UnknownContext wsctx:InvalidState wscf:DuplicateParticipant wscf:InvalidProtocol

Deleted: wscf:InvalidParticipant wscf:ParticipantNotFound

Initiator (as receiver of response)	Responder	"requests"	"responses"
		removeParticipant	participantRemoved wsctx:UnknownContext wsctx:InvalidState wscf:ParticipantNotFound
		replaceParticipant	participantReplaced wsctx:UnknownContext wsctx:InvalidState wscf:TransientFault
		getParticipants	participantList wsctx:InvalidState wsctx:UnknownContext
		getStatus	status wsctx:UnknownContext wsctx:InvalidState
RegistrationService	RegisteringService	replaceRegistration	registrationReplaced wsctx:InvalidState wscf:TransientFault wscf:UnknownService wsctx:UnknownContext

Deleted: .

Deleted: wscf:DuplicateParticipant . wscf:InvalidProtocol . wscf:InvalidParticipant .

Inserted: wscf:DuplicateParticipant . wscf:InvalidProtocol . wscf:InvalidParticipant . wscf:ParticipantNotFou ... [21]

627

4 Conformance considerations

The WS-CF specification defines an *activity group* model where participant services may be enrolled with the group for purposes defined by referencing specifications. WS-CF is itself a referencing specification of WS-Context and extends the basic context structure (**wsctx:ContextType**) defined by that specification. A conformant implementation of WS-CF MUST be based on a conformant WS-Context implementation. Activity group lifecycle demarcation and control SHOULD be managed by the WS-Context Context Service.

Conformant implementations of the Coordination Service MUST follow the rules stated in Section 4, including supporting the **wscf:RegistrationContext** structure, which MAY be passed by reference or by value.

All messages based on the normative WSDL provided in this specification MUST be augmented by a Web services addressing specification to support callback-style message exchange.

Specifications that build on WS-CF MUST satisfy all requirements for referencing specifications that are identified for contexts, participant-services and registration-services.

Deleted:

Formatted: Bullets and Numbering

5 References

[1] WSDL 1.1 Specification, see http://www.w3.org/TR/wsdl

[2] "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, S. Bradner, Harvard University, March 1997.

[3] "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, T. Berners-Lee, R. Fielding, L. Masinter, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[4] WS-Message Delivery Version 1.0, http://www.w3.org/Submission/2004/SUBM-ws-messagedelivery-20040426/

[5] WS-Reliability latest specification, http://www.oasis-open.org/committees/download.php/8909/WS-Reliability-2004-08-23.pdf. See Section 4.2.3.2 (and its subsection), 4.3.1 (and its subsections). Please note that WS-R defines BareURI as the default.

[6] Addressing wrapper schema, http://www.oasis-open.org/apps/org/workgroup/wsrm/download.php/8365/reference-1.1.xsd

[7] WS-R schema that uses the serviceRefType, http://www.oasis-open.org/apps/org/workgroup/wsrm/download.php/8477/ws-reliability-1.1.xsd

[8] Web Services Addressing, see http://www.w3.org/Submission/ws-addressing/

[9] OASIS Web Services Context Specification, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf

Deleted: [1] . OMG, Additional Structuring Mechanisms for the OTS Specification, September 2000, document orbos/2000-04-02.¶
[2] . WSDL 1.1 Specification. See http://www.w3.org/TR/wsdl¶

Deleted: 3

Deleted: .

Deleted: ¶
[4] .

Formatted: Bullets and Numbering

Appendix A. Acknowledgements

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

Formatted: Bullets and Numbering

Appendix B. 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 Executive Director.

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 Executive Director.

Page 8: [1] Deleted **Mark Little** **23/05/2005 12:30 PM**

A service that requires a service reference element MUST use the mustUnderstand attribute for the SOAP header element within which it is enclosed and MUST return a mustUnderstand SOAP fault if the reference element isn't present and understood.

Page 9: [2] Deleted **Mark Little** **30/06/2005 7:13 PM**

4.1Interposition

WS-CF supports the notion of *interposition*: where a Participant Service that is enlisted with a Registration Service also behaves as a Registration Service to other Participant Services. In this way, WS-CF supports the building of graphs and trees by the addition of participants to an activity structure that are themselves registration endpoints.

The technique of interposition uses proxies (or subordinates). Each domain that imports a WS-CF context MAY create a subordinate registration service that enrolls with the imported registration service as though it were a participant. This specification does not prescribe how and when this may occur. Interposition then requires the importing domain to use a different context when communicating with services and participants that are required to register with the subordinate registration service, as shown in Figure 3Figure 3.

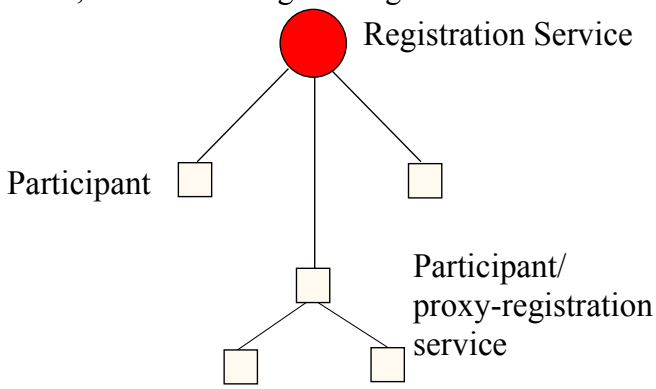

Figure 33, Participant coordinator.

This specification does not define what are allowable forms of graphs that may be created using interposition. Such definitions are the responsibility of referencing specifications.

Page 9: [3] Deleted **Mark Little** **23/05/2005 12:31 PM**

A Registration Service implementation provides support for the Registering Services to enlist Participant services with a specific protocol semantic. Operations on the Registration service MAY be implicitly associated with a Registration context, i.e., it is propagated to the Registration service in order to identify which activity group the Participant is interested in joining. Services requiring protocols that rely explicitly on group membership like transactions or data replication will require that the Registration service MUST be invoked with a Registration context.

Page 11: [4] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

replaceRegistration

This operation on the Registering Service MAY be used by a recovered Registration Service to indicate that it has recovered on a new endpoint address. When a Registration Service fails and subsequently recovers it may not be able to recover at

the same address that prior Registering Services used to enlist with the Registration service. This OPTIONAL operation allows the Registration Service to inform Registering Services that it has moved from the original address to a new address. It may also be used to start recovery operations by the protocol engine.

The use of

Page 11: [5] Inserted **Mark Little** **21/05/2005 10:39 PM**

replaceRegistration

Page 11: [6] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

replaceRegistration SHOULD only be attempted when the Registration Service has failed and recovered on another endpoint because to do otherwise MAY result in continued use of stale wscf:RegistrationContext

Page 11: [7] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

ransientFault error message is sent and the receiver MAY try again.

Page 11: [8] Inserted **Mark Little** **21/05/2005 9:50 PM**

wscf:RegistrationContext

Page 11: [9] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

information elsewhere in the application; the context refers to the old endpoint address for the Registration Service.

A valid wscf:RegistrationContext

Page 11: [10] Inserted **Mark Little** **21/05/2005 9:50 PM**

wscf:RegistrationContext

Page 11: [11] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

MUST accompany this message. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required.

If successful, the

Page 11: [12] Deleted **Mark Little** **21/05/2005 10:39 PM**

registrationRecovered

Page 11: [13] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

registrationReplaced message is sent to the Registration Service. If the recovery handshake occurs in the context of an activity, the message also contains the current status of the activity. This status may be used by recipients to perform local recovery operations, although this will depend upon the protocol in use

If the Registering Service cannot be located, then the wscf:U

Page 11: [14] Inserted **Mark Little** **21/05/2005 10:39 PM**

registrationReplaced

Page 11: [15] Deleted **Kevin Conner** **01/08/2005 11:50 AM**

nknownService error message is sent back.

If the Registering Service cannot deal with recovery of the Registration Service for a temporary reason, the wscf:T

Page 11: [16] Inserted **Mark Little** **21/05/2005 10:44 PM**

getParticipants

This operation returns the list of participants that have been enrolled with the activity group. A valid **wscf:RegistrationContext** MUST accompany this message. This context MAY be passed by reference or by value. It is implementation dependant as to whether any context information other than the basic reference values is required. If successful, the participantList message is sent to the Registering Service.
A referencing specification MAY decide to send the **wsctx:InvalidState** error message if the Activity has begun completion, or has already completed when this operation is attempted.

The termination of the activity group is triggered by the completion of the WS-Context service activity. The relationship between activity groups and participant services is undefined following the termination of an activity group.

Page 15: [17] Deleted **Unknown**

docs.oasis-open.org/wscaf/2004/09/wsctx

Page 15: [18] Deleted **Mark Little** **23/05/2005 12:34 PM**

<type>
http://docs.oasis-open.org/wscaf/2004/09/wsctx/context/type1
</type>

Page 15: [19] Deleted **Mark Little** **23/05/2005 12:34 PM**

<type>
http://example.org/wsctx/context/type1
</type>

Page 15: [20] Change **Unknown**

Formatted Bullets and Numbering

Page 17: [21] Inserted **Mark Little** **28/06/2005 6:41 PM**

		removeParticipant	participantRemoved wsctx:UnknownContext wsctx:InvalidState wscf:DuplicateParticipant wscf:InvalidProtocol wscf:InvalidParticipant wscf:ParticipantNotFound
		replaceParticipant	participantReplaced wsctx:UnknownContext wsctx:InvalidState wscf:TransientFault
		getParticipants	participantList wsctx:InvalidState wsctx:UnknownContext
		getStatus	status wsctx:UnknownContext wsctx:InvalidState
RegistrationService	**RegisteringService**	replaceRegistration	registrationReplaced wsctx:InvalidState wscf:TransientFault wscf:UnknownService wsctx:UnknownContext