Transactions in the world of Web services, Part 1

An overview of WS-Transaction WS-Coordination

Tom Freund (tjfreund@uk.ibm.com), IBM
Tony Storey (tony_storey@uk.ibm.com), IBM

August 2002

This paper presents and illustrates a high-level overview of the Web service specifications for WS-Coordination and WS-Transaction. The new specifications outline the mechanisms required when creating reliable applications by connecting together Web services. These Web services need to participate and cooperate on the agreement of the overall application outcome. The WS-Coordination specification provides a generic foundation for Web services coordination. It provides support for standard transaction mechanisms that exist in today's marketplace. The WS-Transaction specification includes a definition of atomic and business transaction protocols. It is anticipated that additional patterns and protocols will emerge and be based on an extensible coordination framework defined in the specification. These specifications tackle the growing need for consistent support of transactions and addresses the more general requirement to guarantee the reliable coordination of operations across Web services.

Background

Web services are self-contained, modular business process applications that are based on the industry standard technologies of WSDL (to describe), UDDI (to advertise and syndicate), and SOAP (to communicate). Web services provide a means for different organizations to connect their applications with one another to conduct business across a network in a platform and language independent manner.

However, missing so far from these technologies is the support of a facility to provide consistency and reliability for Web service applications.

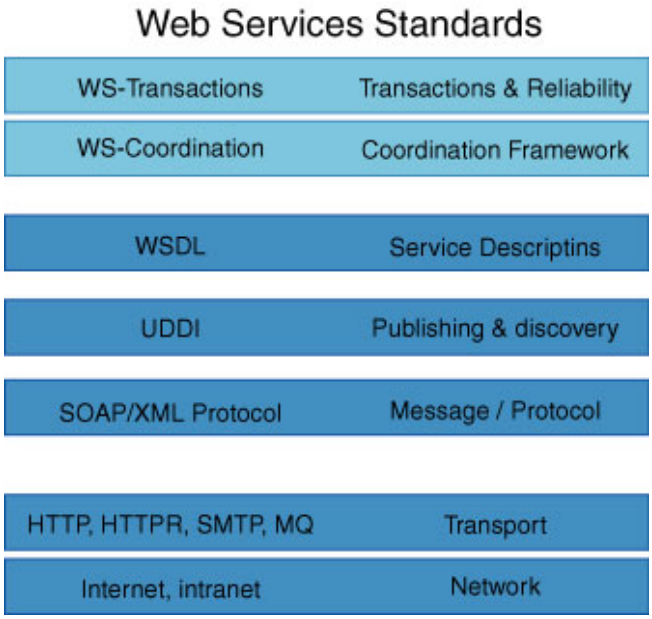
Transactions are a fundamental concept in building reliable distributed applications. A transaction is a mechanism to insure all the participants in an application achieve a mutually agreed outcome. Traditionally, transactions have held the following properties collectively referred to as *ACID*:

- Atomicity: If successful, then all the operations happen, and if unsuccessful, then none of the operations happen.
- Consistency: The application performs valid state transitions at completion.
- Isolation: The effects of the operations are not shared outside the transaction until it completes successfully
- Durability: Once a transaction successfully completes, the changes survive failure.

A Web service environment requires the same coordination behavior provided by a traditional transaction mechanism to control the operations and outcome of an application. However it also requires the capability to handle the coordination of processing outcomes or results from multiple services, in a more flexible manner. This requires more relaxed forms of transactions -- those that do not strictly have to abide to the ACID properties--such as collaborations, Workflow, Realtime processing, etc. Additionally, there is a need to group Web services into applications that require some form of correlation, but do not necessarily require transactional behavior. The WS-Coordination and WS-Transaction specifications provide a WSDL definition for such coordinated behavior.

The current set of Web services specifications [WSDL, SOAP] (see Figure 1) defines protocols for Web services interoperability. Web services increasingly tie together large number of participants to form distributed applications. The resulting applications can be potentially quite complex in structure, with complex relationships between their participants.

Figure 1. Web services standards



The execution of such an application consists of a series of *activities*. An activity, as defined here, is a general-purpose computation carried out as a set of scoped operations on a collection of Web services that require a mutually agreed outcome. Such applications often take some time to complete due to business latencies, network latencies, and waiting for users to interact with the application.

The system described in the WS-Coordination specification is a generalized facility to allow the management of the activities or tasks related to the overall application. WS-Coordination supports, integrates, and unifies several popular coordination models that provide mechanisms and technologies that will allow a variety of systems to interoperate transactionally; for example, the specification will allow for interoperability between the Web services models from IBM and Microsoft.

WS-Coordination provides standard mechanisms to create and register services, using the protocols defined in the WS-Transaction specification that coordinate the execution of distributed operations in a Web services environment (for example, atomic transaction protocols, long-running business transaction protocols, etc.). It also provides an important foundation layer that will help developers control operations that span across broadly interoperable Web services.

Developers can incorporate these new specifications, as needed, into implementations that support the different levels of Web services applications.

A new design for transactions
This article describes an extensible new approach to transactions achieved through the *Coordination Framework* and the *Coordination Protocols* defined in two new specifications.

The Coordination Framework as defined in the WS-Coordination specification supports the following services :

- *Activation Service* to create an activity.
- *Registration Service* to coordinate protocol selection and register participants.
- *Coordination Service* for activity completion processing.

The Coordination Protocols as defined in the WS-Transaction specifications are:

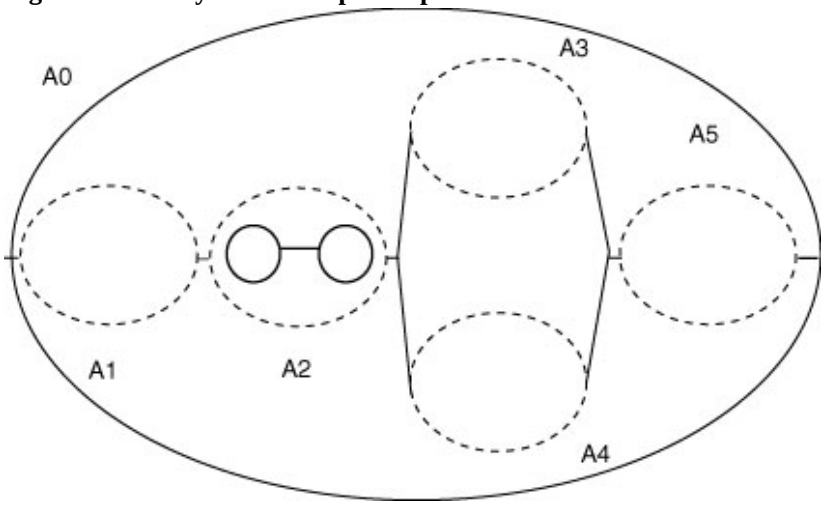
- **Protocols for Atomic Transactions**. The protocols for atomic transactions handle activities that are short-lived. Atomic transactions are often referred to as providing a two-phase commitment protocol. The transaction scope states that all work is completed in its entirety, that is, that the result of an activity, if successful, is that all operations are performed, or if unsuccessful, that no operations have been performed. Upon successful completion the results of the activity are available to other users.
- **Protocols for Business Transactions**. The protocols for business transactions handle long-lived activities. These differ from atomic transactions in that, such activities can take much longer to complete, and to minimize latency of access by other potential users of the resources used by the activity, the results of interim operations need to be released before the overall activity has completed. In light of this, mechanisms for fault and compensation handling are introduced to reverse

the affects of previously completed business activities (for example, compensation, reconciliation, etc).

It is possible to use the above protocols in combination with each other. For example, short running atomic transactions can be part of a long running business transaction. The actions of the embedded atomic transactions are committed and made visible before the long running business transaction completes, and in the event of the long running business transaction failing, the effects of such atomic transactions need to be compensated for, that is, open nested transactions.

For example, consider Figure 2, which shows a series of related activities cooperating during the lifetime of an application. The ellipses represent coordination boundaries, with a number of activities labelled A0 through A5. Particularly relevant are activity A1 that uses one coordination point during its execution and A2 that shows a nesting of activities within its execution. Additionally, the figure illustrates that an application consisting of complex computations across a distributed environment can also be represented as a single activity (A0). The framework is responsible for distributing information that describes the activity between execution environments in order for the hierarchy to be fully distributed.

Figure 2: Activity relationships in a process flow

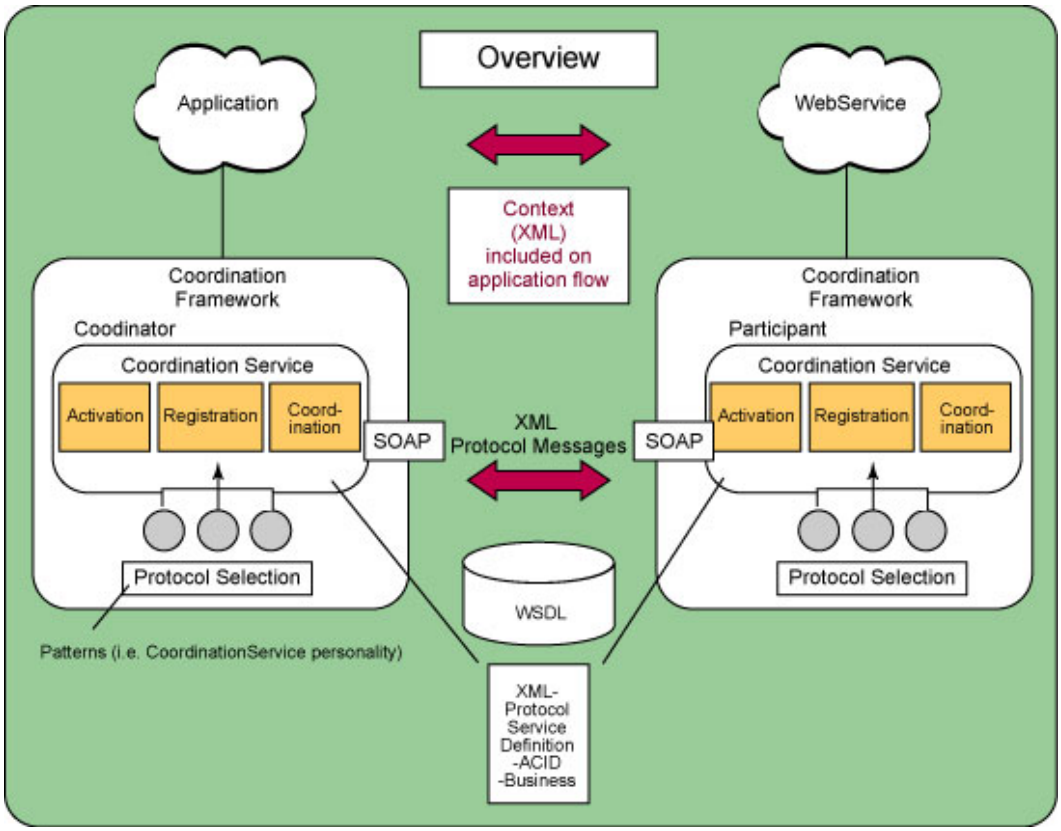


Coordination Framework (WS-Coordination)
The coordination framework (see Figure 3) consists of three elements:

- the Activation service
- the Registration service
- the Coordination service

The operations, and messages, for the services are defined using WSDL.

Figure 3. An overview of WS-Coordination



Activation Service
The activation service uses the `create` message to:

- begin a new activity
- specify the coordination protocols available to the activity

Additionally, the activation service optionally allows the user to specify a relationship between a newly created activity and an existing activity (that is, to establish a subordinate or nested relationship between the activities).

Registration service
The registration service 'register' allows a Web service to register and to select a protocol for the activity. Enrollment and selection allow the Web services involved in the activity to establish the traditional roles of coordinator and participant. The registration process identifies the specific protocol used for activity coordination.

The specification authors envision that other mechanisms will provide the means for a Web service supporting WS-Transaction to set its coordination protocol definitions. The information includes a specification of the transaction protocols and properties supported by the Web service. The mechanism for passing the protocol information between coordinator and participant is not defined.

Coordination service
The coordination service controls the activity completion processing for the registered Web services using the selected coordination protocol (defined in WS-Transaction). A particular coordination protocol provides a definition of the behavior requirements and the operations supported for completion processing. Typically, the operations are identified by role, for example, for atomic transactions, the coordinator provides an interface to the application to direct `completion` (that is, commit and rollback) and the participant provides an interface to the coordinator to direct `agreement` (that is, prepare, commit, rollback).

Context

For each newly created activity, the activation service returns a *CoordinationContext*that contains the following fields:

- `Identifier`: a unique name to identify the `CoordinationContext`
- `Expires`: an activity timeout value
-
- `CoordinationType`: a defined set of coordination protocols that describe the supported completed processing behaviors

`Registration Service`: address of the registration service, the service is used to register interest and participation in a coordination protocol for determining the overall outcome of the activity.

- `Extensibility element`: provides for optional implementation-specific extensions (see Listing 1)

Listing 1. Extensibility elements for implementation-specific extensions to WS-Coordination

```
<soapenv>
 <soapbody>
   <wscoor:CoordinationContext
       <Identifier> ... </Identifier>
       <Expires> ... </Expires>
       <wscoor:CoordinationType> ... </wscoor:CoordinationType>
       <wscord:RegistrationService>
              <Address/>
       </wscoord:RegistrationService>
       <!--extensibility element ->
    </wscoor:CoordinationContext>
 </soapbody>
</soapenv>
```

The following is an example of `CoordinationContext` for an atomic transaction. An `IsolationLevel` element has been added to illustrate an example of a product-specific extension:

Listing 2. An example of a CoordinationContext for an atomic transaction

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
    <S:Header>
        . . .
        <wscoor:CoordinationContext
            xmlns:wsme="http://www.w3.org/2002/06/msgext"
            xmlns:wscoor=http://www.w3.org/2002/06/Coordination
            xmlns:myApp="http://www.w3.org/2002/06/myApp">
            <wsme:Identifier>
                 http://foobaz.com/SS/1234
            </wsme:Identifier>
            <wsme:Expires>
                 2002-06-30T13:20:00.000-05:00
            </wsme:Expires>
            <wscoor:CoordinationType>
                http://xml-soap.org/2002/06/AtomicTransaction
            </wscoor:CoordinationType>
            <wscoor:RegistrationService>
                <Address>
                    http://myservice.com/mycoordinationservice/registration
                </Address>
                <myApp:BetaMark> ... </myApp:BetaMark>
                <myApp:EBDCode> ... </myApp:EBDCode>
            </wscoor:RegistrationService>
            <myApp:IsolationLevel>
                  RepeatableRead
            </myApp:IsolationLevel>
        </wscoor:CoordinationContext>
        . . .
            </S:Header>
```

In order for an activity to span a distributed environment the context information must be associated with an application messages.

Implementations generally append context to the application message at the source in order to establish the execution environment at the target.

Scenarios of Web services transactions
We have explained the basic layout of how the coordination and transaction system in Web services work. The Coordination Framework provides a system to coordinate communications across a distributed network of Web services and can work with strict transaction-based systems with ACID properties as well as other forms of transactions, while the Coordination Protocols can implement actual ACID transactions.

In a following article, we will discuss two scenarios for Web services transactions in step-by-step detail.

Resources

- Participate in the discussion forum on this article by clicking **Discuss** at the top or bottom of the article.
- See the first part of this two-part article Transactions in the world of Web services, Part 2.
- Automating Business Processes and Transactions offers an introduction to the concepts of workflow and transactions in Web services.
- Read The Web Services Coordination specification document.
- See The Web Services Transaction specification document.
- The Business Process Execution Language for Web Services specification defines the higher abstraction level of workflow across multiple services.
- This paper explains in technical detail how Business process will work with Web services.

About the authors
Biographical text about author goes here; remove the text between the author tags if no biographical information exists. Paragraph tags are needed only if there is a second paragraph. You can contact Tom at tjfreund@us.ibm.com.

Biographical text about author goes here; remove the text between the author tags if no biographical information exists. Paragraph tags are needed only if there is a second paragraph. You can contact Tony at tony_storey@us.ibm.com.