Web Services Advanced Topics
Beyond SOAP, WSDL, and UDDI (Part 2)

Kelvin R. Lawrence
CTO, Emerging Internet Software Standards
IBM Software Group
klawrenc@us.ibm.com

Related Talks This Week

- Other talks this week that are related to this one (there may be others)…
- Arthur Ryman
  - *Developing Web Services with Eclipse*
  - *What’s New in WSDL 2.0?*
- Michael Perks (was Tony Nadalin)
  - *SOA Security Programming Model*
  - *Model Driven Security Architecture*
- Paul Freemantle
  - *Apache Axis2 – The New Generation of Open Source Web Services*
- Neil Graham
  - *Who's Afraid of XML Schema?*
- Andre Tost
  - *Practical Lessons Learned in Web Services Design and Implementation*
  - *Service Data Objects*
An overview of several new technologies for Web Services:

- **The Web Services “stack” of technologies**
  - A quick update on the basic web services specs.

- **Detailed look at some advanced web services topics:**
  - Security and the Security Roadmap
  - Policy
  - Trust, Secure Conversation and Federation
  - Addressing
  - Reliable Messaging
  - Transactions
  - Management
  - Business Process Modeling and Execution

- **Web Services and Interoperability**
  - WS-I Status and work in progress
  - Industry profiles
Web Services – A “Stack” View

- Business Process Execution Language (BPEL)
- WS-Coordination
- WS-Transactions
- WSDL
- SOAP, SOAP Attachments
- XML, XML Infoset
- Transports
- WS-Policy
- WS-Security family of specifications
- WS-Reliable Messaging
- WS-Distributed Management
- UDDI
- Other protocols
- Other services
- Description and Discovery
- Messaging and Encoding
- Business Processes
- Quality of Service
- Transport
Technologies Discussed in This Session

- Business Process Execution Language (BPEL)
- WS-Coordination
- WS-Transactions
- WS-Security family of specifications
- WS-Reliable Messaging
- WS-Distributed Management
- WSDL
- WS-Policy
- UDDI
- SOAP, SOAP Attachments
- Other protocols
- Other services
- XML, XML Infoset
- Transports
- Description and Discovery
- Messaging and Encoding
- Transport
Web Services and Addressing
WS-Addressing

- **Goals**
  - Allow specific service endpoint instances to be referenced.
  - Allow endpoint descriptions to be dynamically created/customized.
  - Enable asynchronous messaging.
    - Can be used to help build reliable/asynchronous message exchanges
    - When combined with WS-ReliableMessaging for example.
  - Independent of transport or messaging system (*i.e.* app. level).
  - Allow other specs to be built easily on top of this one.

- **Status**
  - W3C Candidate Recommendation (August 17th 2005)
  - Core spec
    - [http://www.w3.org/TR/ws-addr-core/](http://www.w3.org/TR/ws-addr-core/)
  - SOAP Binding
    - [http://www.w3.org/TR/ws-addr-soap/](http://www.w3.org/TR/ws-addr-soap/)
Endpoint Reference (EPR) Syntax (Pseudo Schema)

```xml
<wsa:EndpointReference>
  <wsa:Address>
    xs:anyURI
  </wsa:Address>

  <wsa:ReferenceParameters>
    xs:any *
  </wsa:ReferenceParameters> ?

  <wsa:Metadata>
    xs:any *
  </wsa:Metadata> ?

</wsa:EndpointReference>
```
EPR Example (SOAP mapping next slide)

```xml
<wsa:EndpointReference
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsaw="http://www.w3.org/2005/03/addressing/wsdl"
    xmlns:fabrikam="http://example.com/fabrikam"
    xmlns:wsdli="http://www.w3.org/2005/08/wsdl-instance"
    wsdli:wsdlLocation="http://example.com/fabrikam
    http://example.com/fabrikam/fabrikam.wsdl">
    <wsa:Address>http://example.com/fabrikam/acct</wsa:Address>
    <wsa:ReferenceParameters>
        <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
        <fabrikam:ShoppingCart>ABCDEFG</fabrikam:ShoppingCart>
    </wsa:ReferenceParameters>
    <wsa:Metadata>
        <wsaw:InterfaceName>fabrikam:Inventory</wsaw:InterfaceName>
    </wsa:Metadata>
</wsa:EndpointReference>
```
EPR to SOAP Mapping...

```
<S:Envelope
  xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="..." xmlns:f="... ">
  <S:Header> ...
  <wsa:To>http://example.com/fabrikam/acct</wsa:To>
    <f:CustomerKey>123456789</f:CustomerKey>
    <f:ShoppingCart>ABCDEFG</f:ShoppingCart>
  </S:Header>
  <S:Body> ...
</S:Envelope>
```
Another Example of WS-Addressing in a SOAP Header

```xml
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:MessageID>
      http://example.com/6B29FC40-CA47-1067-B31D-00DD010662DA
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>
        http://example.com/business/client1
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://example.com/fabrikam/Purchasing</wsa:To>
    <wsa:Action>http://example.com/fabrikam/SubmitPO</wsa:Action>
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```
A Word About Faults

- WS-Addressing defines a `<FaultTo>` element.
  - Can have faults sent to different place than main-line application messages.
  - A particular piece of software may not care about the main line processing of messages but may be setup to specifically handle error notifications.
Web Services and Reliable Messaging
**Goal:** Define a protocol to assure reliable message exchange between distributed applications exchange in the presence of software component, system, or network failures.

- Errors in transmission may disrupt a conversation
  - Messages can be lost, duplicated, or arrive in a different order than they were sent
  - Host systems may fail and lose volatile state

- Delivery Assurances supported
  - At-Most-Once, At-Least-Once, Exactly-Once, Ordered
  - When this is not possible, a fault is raised on the Initial Sender, or the Ultimate Receiver, or both
WS-ReliableMessaging: Features

- WS-RM (WS-ReliableMessaging) defines:
  - A messaging protocol to identify, track, and manage reliable delivery between a source and a destination.
  - Defines a SOAP binding for interoperability

- WS-RM is extensible:
  - Bindings for other protocols may also be defined
  - Additional functionality (e.g. security) can be composed.

- WS-RM integrates with and complements other specs
  - Integrating WS-RM and WS-Security yields secure and reliable message exchange
  - WS-RM uses the WS-Policy specifications for defining and attaching reliable messaging policy assertions
The Reliable Messaging Model

1. Requester App sends a message for reliable delivery
2. Source transmits the message (one or more times)
3. Destination receives and acknowledges the message
4. Destination delivers the message to the Provider App
Setup for Reliable Messaging

- There are three requirements that must be satisfied prior to using Reliable Messaging:
  1. Source must resolve Destination’s endpoint reference
  2. Source must obtain Destination’s policies, if any, and send messages that conform to these requirements
  3. A security context must be set up if required
Protocol Elements

- `<Sequence>`
  - Carries the Identifier and MessageNumber that uniquely identifies the message within the Sequence context

- `<SequenceAcknowledgement>`
  - Carries the Identifier that uniquely identifies the Sequence context
  - Carries AcknowledgementRange elements that cover the entire set of messages received by the RM Destination for the Sequence

- `<AckRequested>`
  - Requests that the RM Destination send a SequenceAcknowledgement immediately

- `<LastMessage>`
  - Carried in the Sequence header of the last message in a Sequence
  - RM Destination must send a SequenceAcknowledgement
Sequence Lifecycle

- **<CreateSequence operation>**
  - RM Source invokes the CreateSequence operation on the RM Destination endpoint
  - A request to establish a new Sequence context
  - RM Destination creates a new Sequence context and assigns it a unique Identifier and sends CreateSequenceResponse
  - RM Source, upon receipt of CreateSequenceResponse can begin sending messages in the Sequence

- **<TerminateSequence>**
  - RM Source sends this to RM Destination upon receipt of the SequenceAcknowledgement that covers the complete set of messages in the Sequence

- **<Bilateral Sequence Negotiation>**
  - Optimization of the case in which the RM Source endpoint can anticipate that the RM Destination endpoint will be requesting a Sequence for reliably delivered response messages
Example

- A sequence is initiated using `<CreateSequence>`
  - This is a required part of the protocol
- The RM Destination creates the Sequence ID
- The RM Source labels messages with a `<Sequence>`:
  - Constructs the `<sequence>` using the identifier returned from the destination during `<CreateSequence>` (a unique sequence group id e.g. “http://fabrikam123.com/abc”)
  - Sends first message with id and message number 1
  - Sends second message with id and message number 2
  - Sends third message with id and message number 3, along with a token to indicate that this is the last message of the sequence

- The `<Sequence>` element looks like this for the third message:

```xml
<wsrm:Sequence />
  <wsrm:Identifier>http://fabrikam123.com/abc</wsrm:Identifier>
  <wsrm:MessageNumber>3</wsrm:MessageNumber>
  <wsrm:LastMessage/>
</wsrm:Sequence>
```
Example (Continued)

- Suppose message 2 is lost or delayed. The Destination:
  - Receives message 1
  - Receives message 3
  - Acknowledges receipt of messages 1 and 3, like so:

```xml
<wsrm:SequenceAcknowledgement>
  <wsrm:Identifier>http://fabrikam123.com/abc</wsrm:Identifier>
  <wsrm:AcknowledgementRange Lower="1" Upper="1"/>
  <wsrm:AcknowledgementRange Lower="3" Upper="3"/>
</wsrm:SequenceAcknowledgement>
```

- Notes:
  - The `<AcknowledgementRange>` indicates a range of received messages, from a lower number to an upper number
  - More than one `<AcknowledgementRange>`s can be used when there are gaps in the sequence of received message (as here)
Example (Continued)

- **The Source:**
  - receives acknowledgement for messages 1 and 3
  - decides to resend message 2 with same sequence group **ID**, along with a tag requesting immediate acknowledgement

- **The Destination:**
  - receives re-sent message 2, sends acknowledgement

- **The Source** receives the acknowledgement. The sequence is now complete.

- **Meanwhile:**
  - Destination later receives the lost copy of message 2
  - Destination identifies and drops duplicate message (sequence id and number were retained to detect duplicates).
WS-RM Protocol

Reliable Messaging Protocol

Establish Protocol Preconditions

CreateSequence()

CreateSequenceResponse( Identifier = http://fabrikam123.com/abc )

Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 1 )

Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 2 )

Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 3, LastMessage )

SequenceAcknowledgement( Identifier = http://fabrikam123.com/abc,
AcknowledgementRange = 1,3 )

Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 2, AckRequested )

SequenceAcknowledgement( Identifier = http://fabrikam123.com/abc,
AcknowledgementRange = 1...3 )

TerminateSequence( Identifier = http://fabrikam123.com/abc )
Reliable Messaging Policy Assertions

- WS-RM defines a policy assertion for use with the WS-Policy framework specifications
  - `<wsrm:RMAssertion>`

- The policy assertion contains a number of properties:
  - **InactivityTimeout** assertion: specify an interval as milliseconds, after which a retry may be attempted
  - **RetransmissionInterval**: how often retries will be attempted
  - **ExponentialBackoff**: modifies retry algorithm
  - **AcknowledgementInterval**: specifies the minimum interval between receipt of a message and the sending of a `<SequenceAcknowledgement>`

- WS-PolicyAttachment may be used to associate a policy with a `<wsdl:port/>` or `<wsdl:binding/>`
RMAssertion Example

```xml
<wsp:Policy wsu:Id="MyRMPolicy" >
  <wrm:RMAssertion>
    <wrm:InactivityTimeout
        Milliseconds="600000" />
    <wrm:BaseRetransmissionInterval
        Milliseconds="3000" />
    <wrm:ExponentialBackoff />
    <wrm:AcknowledgementInterval
        Milliseconds="200" />
  </wrm:RMAssertion>
</wsp:Policy>
```
Fault Management

- `<SequenceFault>`, used with the SOAP fault mechanism, signals specific exceptions in reliable message processing

- Some fault codes:
  - `wsrm:SequenceTerminated`
  - `wsrm:UnknownSequence`
  - `wsrm:InvalidAcknowledgement`
  - `wsrm:MessageNumberRollover` (message number overflows unsigned long)
  - `wsrm:LastMessageNumberExceeded` (message number is greater than number of previously received message that was marked “LastMessage”)
  - `wsrm:SequenceRefused` (can’t start requested sequence)
Security Considerations

- WS-RM recommends use of WS-Security when security is required

  - The `<wsrm:Sequence>` header needs to be signed with the body in order to "bind" the two together

  - `<wsrm:SequenceAcknowledgement>` header MAY be signed independently (this reply, independent of the message, may not be a security concern)

  - Because Sequences commonly exchange a number of messages, it is recommended that a security context be established using WS-SecureConversation.
A new OASIS Technical Committee (TC) was formed in June 2005

- Web Services Reliable Exchange (WS-RX) TC

- The TC has produced a Working Draft (July 2005)
  - WS-Reliable Messaging 1.0

- The TC hopes to have its first Committee Draft published by the first week of Nov. 2005
  - No significant changes to the protocol.
Reliable Messaging – Further Reading

- Spec as submitted to OASIS (input document)

- Whitepapers
  - Reliable Message Delivery in a Web services world
  - Implementation Strategies for WS-Reliable Messaging
  - WS-RM Reloaded
  - WS-RM and WS-R: Can SOAP be reliably delivered from confusion

- Sample code available in the IBM ETTK
  - Completed 2nd round of interop testing May 2004
Web Services and Transactions
Web Services Transactions

- Business Process Execution Language (BPEL)
- WS-Coordination
- WS-Transactions
- WS-Security family of specifications
- WS-Reliable Messaging
- WS-Distributed Management
- WSDL
- WS-Policy
- UDDI
- SOAP, SOAP Attachments
- Other protocols
- XML, XML Infoset
- Other services
- Transports
- Business Processes
- Quality of Service
- Description and Discovery
- Messaging and Encoding
- Transport
Why Transactions?

- Data must be kept “consistent”

No matter what software or hardware failure, Jim expects his money to obey the law of conservation of cash: it neither evaporates nor suddenly appears from nowhere (the latter is acceptable to him, but not to the bank).
The Application Programmer View*

```
TransferCash(fromAcct, toAcct, amount)
OpenDatabases(fromAcct, toAcct)
BeginTransaction
  fromAcct -= amount
  toAcct += amount
Commit**
EndTransaction
CloseDatabases(fromAcct, toAcct)
Return
```

* Lots of ways to signal that you want transactional semantics. Middleware has to implement the transactional semantics, and middleware might have to coordinate actions with other middleware to accomplish it.

** Or Rollback means undo it all – make like it never happened.

Note (a container will likely, in reality do all this for you)
The Problem – The Need for Coordination

- **Web Services are self-contained business applications**
  - Based on industry standard technologies of WSDL, UDDI and SOAP
  - Provide a means for different organizations to connect their applications to conduct business across a network.

- **Currently lack the facility to ensure consistency and reliability.**

- **Require a mechanism for all participants in a distributed application to achieve a mutually agreed outcome.**

- **Activities may have large spectrum of different behaviors**
  - There is no *one size fits all* transaction model appropriate for all web-service-based applications.
  - Trying to define one is more futile than herding cats.
    - Need to consider ACID 2PC, open nested, compensation, long-running with reconciliation, client-session scoping, ....
    - ACID = Atomicity (all or none), consistency, isolation (lock), durability (long lasting)
Web Services Focus in Three Areas

- WS-Coordination
- WS-AtomicTransaction
- WS-BusinessActivity

IBM: Developer Works – Web Services: Technical Library/Standards
http://www.ibm.com/developerworks/webservices

MSFT: Web Services Development Center
http://msdn.microsoft.com/webservices

BEA: Dev2Dev – WebServices
http://dev2dev.bea.com/technologies/webservices/index.jsp
Specifications

- WS-C defines a framework for deploying coordination protocol sets
  - Activation Service
  - Registration Service
  - Coordination Context

- WS-AT & BA define coordination types for specific transaction models
  - Atomic transactions where the results of operations are not made visible until the completion of the unit of work.
  - Business transactions where the results of operations are made visible before the completion of the unit of work and need to be compensated rather than rolled back to undo the work.
Simplified WS-BA / WS-AT Comparison

**WS-AT**

- Short duration
  - Locks de rigueur
- Suited for more controlled environment
- Classical resource manager mapping
  - think database (not business processes crossing business boundaries).
- Easier to think about and program
  - “Rollback” or “commit”
  - Automatic rollback in abnormal/error termination case.
- All RM’s move in one direction (everybody commits or rolls back in unison).

**WS-BA**

- Longer duration
  - Avoid locks
  - Treat even small things as individual transactions “reserve a seat” not “schedule a trip”. Do things step by step. Undo a “mess” using compensation logic.
- Suited for loosely coupled environment
- Business process mapping
- More complex
  - “Compensate”
- More flexible RM participation
  - They don’t have to trust applications so much
1. Make an airline reservation
   • confirmed – COMPLETED transaction
   • Airline holding no database locks
2. Yikes, can’t get a hotel in that city
3. Cancel the airline reservation
   • Also a COMPLETE transaction
   • COMPENSATES for the first transaction

The “holiday scheduling” application is harder to write, because it must know to issue compensating transactions, instead of just say “rollback.”

*e.g.*, what if the application crashes after it got the airline reservation? It must assume a sort of “resource manager” role. (Or the middleware helps out.)
Elements of WS-Coordination

- Defines the coordination context and provides a mechanism for resource managers to register interest in the context so that (for example) they are driven by termination protocols.

- Activation service
  - How to create a context

- Registration service
  - How to register interest in a context
A Basic Example – Using Atomic Transactions

- Application (or middleware) starts a coordination scope via Activation service.
  - Application specifies coordination type to be used by participants in negotiating the outcome of the activity.
  - It typically registers for the completion protocol in order to be able to initiate completion.
  - The endpoint reference for the completion protocol's Coordinator is returned to the application.

- The application initiates business invocations on partner services (e.g., take money out of one account and put it back into another account – funds transfer)
  - CoordinationContext is passed along with application messages
  - The endpoint reference for the Registration service is contained in the CoordinationContext.

- A Participant in the invoked domain may register for any of the protocols defined by the coordination type that are to be used in the completion of the activity.
  - In the example, the databases register for the 2PC protocol
Atomic Transaction Flow

APP

RM-1 (DB)
(fromAcct)

RM-2 (DB)
(toAcct)

COOR

WS-C Registration Service

WS-AT Completion Protocol

WS-AT 2PC Protocol

Begin Tran

UR RR (Undo/Redo rec)

Prepare (ready?)

Commit

Committed

Take Money Out

Put Money In

Begin

Prepare

Commit

Committed

Prepared

Prepared

Take Money Out

Put Money In

(undo/redo rec)

T02

B01

Key:
Application Messages
Protocol Messages
Boxes = log records
Activation Service (WSDL)

```xml
<wsdl:portType name="ActivationCoordinatorPortType">
<wsdl:operation name="CreateCoordinationContextOperation">
<wsdl:input message="wscoor:CreateCoordinationContext"/>
</wsdl:operation>
</wsdl:portType>

<wsdl:portType name="ActivationRequesterPortType">
<wsdl:operation name="CreateCoordinationContextResponseOperation">
<wsdl:input message="wscoor:CreateCoordinationContextResponse"/>
</wsdl:operation>
</wsdl:portType>
```
CreateCoordinationContext Message

CoordinationType for supported protocols (e.g. 2PC)

```
<w scoor:CreateCoordinationContext>
  <wscoor:CoordinationType ... />
  <wsu:Expires ... />?
  <CurrentContext ... />?
  <!-- Extension Element -->*
</w scoor:CreateCoordinationContext>
```

Optional: Imported Context e.g. superior coordinator
Think of context as the “thing” that enables all of the related flows to be related.
Register Message

```xml
<wscoor:Register>
  <ProtocolIdentifier />
  <ParticipantProtocolService />?
  ...
</wscoor:Register>
```

Protocol to register for

Endpoint Reference of the registered participant

In the example, the databases register for 2PC protocol
WS Atomic Transaction: Completion Protocol

```xml
<wsdl:portType name="CompletionCoordinatorPortType">
  <wsdl:operation name="CommitOperation">
    <wsdl:input message="wsat:Commit"/>
  </wsdl:operation>
  <wsdl:operation name="RollbackOperation">
    <wsdl:input message="wsat:Rollback"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="CompletionInitiatorPortType">
  <wsdl:operation name="CommittedOperation">
    <wsdl:input message="wsat:Committed"/>
  </wsdl:operation>
  <wsdl:operation name="AbortedOperation">
    <wsdl:input message="wsat:Aborted"/>
  </wsdl:operation>
</wsdl:portType>
```
Architecture for Distributed WebService Activities

WS-TX participants

Web service

Application message set

XML message with context

Activity context

WS-TX participants

Web service

security
QoS...

WS-TX coordinator

Registration service

WS-Coordination

Web service platform (Vendor A)

Key
Middleware
Application
Protocol-specific service

security
QoS...

WS-TX resource

WS-Coordination

Web service platform (Vendor B)
Standards Update

The following specs are now all submitted to OASIS.
- WS-AT
- WS-BA
- WS-Coordination

OASIS has formed the WS-TX Technical Committee


A call for participation went out in Oct 2005.
WS Transactions Downloads From IBM

- ETTK available now
  - WS-C, WS-AT, and WS-BA
  - Example code

- Websphere Application Server 6.0
  - WS-C, WS-AT

- Code (ETTK, WAS, more) at

- Articles, specifications at
Web Services and Management
Web Services & Management

Business Process Execution Language (BPEL)

WS-Coordination
WS-Transactions
WS-Security family of specifications
WS-Reliable Messaging
WS-Distributed Management

Business Processes

WSDL
WS-Policy
UDDI

Quality of Service

SOAP, SOAP Attachments
XML, XML Infoset
Other protocols
Other services

Description and Discovery

Transports

Messaging and Encoding

Transport
Two Major Facets of Web Services Management

- Management Using Web Services (MUWS)
  - Management applications on a web services platform
  - Using web services to describe and access manageability of resources

- Management Of Web Services (MOWS)
  - An implementation of Management Using Web Services where the resource being managed is also a Web Service.
Management Using Web Services (MUWS)

Diagram:
- **Manageability Consumer** (e.g., a Management System, any Web service application, etc.)
- **Message Exchanges**
  - Information
  - Events
  - Acknowledgments

**Discovery**
- Requests, Subscriptions, Control

**Manageable Resource** (e.g., a printer that can notify when a toner is low)

**Web service endpoint**

**WS-Resource**
Why Add in This New Layer?

- Managers need “end to end” access to manageability
  - Across platforms, languages, applications, AND existing management technologies
  - Federated management is required.
  - SLA Monitoring, Workflows, Work balancing, Utility computing, pay-per-Quality of Service...
  - Standards are just starting, we’re developing technology to help us solve these upcoming challenges

- Ubiquitous, low entry point infrastructure
  - HTTP & the Web

- It’s just distributed computing, again
  - So leverage Web services infrastructure for scalability, security, etc., don’t re-invent it

- Integration/interoperability between business and IT management domains of the enterprise
  - Management systems gain visibility into business applications and processes
  - Business applications and processes can take advantage of the manageability of resources
Web Services Distributed Management (WSDM)

- Web services architecture replaces or ‘hides’ the traditional Manager/Agent architecture

- Managers always ‘talk’ to the resource while the actual Web Service endpoint may be supported by any number of management agents

- Web Services de-couple manageability capabilities from
  - HOW you access the resource
  - WHERE you access the resource
  - HOW the resource is implemented
  - WHEN the resource was implemented
WSDM Status

- OASIS Technical Committee formed in March, 2003

- Committee Drafts in February, 2004
  - Successful interoperability testing among 5 vendors
    - April 2004

- Revised Committee Drafts in November 2004

- **WSDM V1 is now an OASIS Standard as of May 2005**
Business Process Modeling and Execution
Business Process Execution Language (BPEL)

- WS-Coordination
- WS-Transactions
- WSDL
- SOAP, SOAP Attachments
- XML, XML Infoset
- Transports
- WS-Security family of specifications
- WS-Policy
- WS-Reliable Messaging
- WS-Distributed Management
- UDDI
- Other protocols
- Other services

Business Processes

Quality of Service

Description and Discovery

Messaging and Encoding

Transport
Requirements for Business Processes

- We need a model for describing simple or complex exchanges that characterize business partner interactions
  - **Stateful**, long-running interactions involving two or more parties
  - **Sequences** of peer-to-peer message exchanges
    - Synchronous exchanges
    - Asynchronous exchanges **with correlation**
WSDL Provisions for Web Services

- Organizes Web services interfaces as
  - “port types” – groups of related operations
  - the operations themselves

- Defines Web services as
  - a **stateless** interaction model of
  - **individual** peer-to-peer message exchanges
    - Synchronous exchanges or
    - **Uncorrelated** asynchronous exchanges
Separation of WHAT from HOW

- **Business Process:** what to do
  - a sequence of activities models a business process
  - IT provides tools to allow business people to define, monitor, and manage business processes

- **WSDL:** how to execute activities
  - an activity can be a Web service, defined by a SOAP interface and a WSDL description; internal, or from a business partner
  - a business process can be externalized as an activity for a client app or another business process
The WS-BPEL Specification
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

- A model for describing simple or complex exchanges that characterize business partner interactions
  - use standard Web services to invoke partner’s process
  - expose resulting business process as a Web service
  - define control elements for workflow
  - create a fully-executable, portable script

- Technology proposal by IBM, BEA, and Microsoft
  - version 1.0 published in August 2002
  - version 1.1 published in April 2003
  - a merger of IBM’s WSFL and Microsoft’s XLang
  - Submitted to OASIS TC with royalty-free terms

- Builds on and extends XML and Web Services specifications
  - expressed in XML
  - uses and extends WSDL
  - WSDL and XML Schema for data model
  - XPath for assignments, conditions, etc.
Web Services and Choreography

A Business Process

- is composed of choreography elements ("activities") to define behavior
- activities include ability to invoke Web services, control flow, etc
- resulting business process is exposed as one or more Web services

The BPEL model describes:

1. Operation sequencing constraints
2. Service Behavior (ordered activities)
3. Service identity management
4. Dynamic partner and service selection
A BPEL script will run on any BPEL-compliant engine, so it’s platform- and vendor-neutral.

Create with your favorite BPEL Modeling Tool.

Run on any BPEL-compliant platform.
Handling an Incoming Request

A - The `<receive>` activity
- specifies partner, port type, operation it expects to receive
- does a blocking wait
- wakes up when the specified message is received
- proceeds to next activity
- optionally specifies that a new BP instance should be created on receiving the message

B - The `<reply>` activity
- specifies same partner, port type, and operation as `<receive>`
- sends the response message
- proceeds to next activity

Note: this is the synchronous model
- Asynch model discussed on next page.
Invoking a Web Service

- A partner can invoke a service from another partner using SOAP and WSDL.

- Two models:
  - **Synchronous**
    - `<invoke>` sends a message and the protocol waits for the response
  - **Asynchronous**
    - `<invoke>` sends a message and the BPEL engine waits for a response on the “callback” operation

Note: services that are invoked can be ordinary Web services or other business processes.
The `<sequence>` and `<flow>` Activities

`<sequence>` activities run one at a time in the order they are listed.

`<flow>` activities run concurrently:
- the flow activity does not complete until all its activities complete (synchronization)
- flow branches are often `<sequence>`s
Combining Flows and Sequences

- `<flow>`s and `<sequence>`s can nest to any required depth
  - a `<sequence>` can contain `<flow>`s
  - a `<flow>` can contain `<sequence>`s
  - activities link other Business Processes or Web services
Cross-dependencies

- A <link> can be used to alter the behavior of a <flow>, crossing the boundaries of <sequence> and <flow> as required.
  - In this example:
    - X is declared as the source of the link
    - Y is declared as the target of the link
    - When X completes, the link becomes “active
    - Both W and X must complete before Y can run. If either is not completed, Y waits until both are completed.
BPEL Data Model

- Variables* represent `<process>` context
  - Like object instance data
  - Persistent messages shared between activities in a business process
  - Can also be used for any required non-message data
  - Define input/output of activities or context for fault- and compensation handlers
  - Defined by WSDL messages or using XML Schema
  - Global or scoped definition
  - Can be manipulated via `<assign>` activity often using the `<copy>`, `<from>` and `<to>` elements.

* Variables were called “containers” in BPEL 1.0
Process Instances and Correlation

- Manage interaction between stateful service instances
  - Instance identification via selected “token” in messages exchanged between services
  - `<correlationSet>` identifies tokens
  - Used by activities to address appropriate service instances
  - Global or scoped definition

Other BPEL Features

- These can be defined (or redefined) within a `<scope>`:
  - Fault handling
  - Event handling
  - Compensation
  - Variables
  - Correlation sets
  - Concurrency

- Compensation handling
  - define flow for undoing previously completed activities

- Fault handling
  - define steps for handling a fault thrown by any activity

- `<wait>`
  - for interval
  - until specified time

- `<switch>`
  - Like C++/Java switch except condition for each case

- `<if>, <then>, <else>, <elseif>`
  - Works as you would expect.

- `<pick>`
  - Combination of `<receive>` and switch
  - Handle one of a list of expected incoming messages

- Event handling
Executable and Abstract Processes

- **Executable** processes
  - Complete business process details
  - Can be run on all compliant environments

- **Abstract** processes
  - Specify constraints of message exchange
  - Describe business protocol
  - Simplified model for use in business partner integration

---

Property 1

Property n

Variable 1

Variable n

Hide Complexity

Property = 42
BPEL and Standardization

- An OASIS TC is now working to standardize BPEL 2.0

- Latest Committee Draft (1st September 2005)

- BPEL 1.1 Specification – published April, 2003
  - Submitted as input to the OASIS work.
Changes in BPEL 2.0 (from 1.1)

- Major differences between 1.1 and 2.0:
  - Added if-then-else, repeatUntil, validate, forEach
    - Completion condition in forEach activity
  - Added extensionActivity element.
  - Variable initialization
  - XPath access to variable data: "$variable[.part]/location"
  - XML schema variables for WS-I compliant doc/lit-style WS interactions
  - Locally declared messageExchange for correlating receive and reply activities.
**BPEL4People**


- **WS-BPEL Extension for People – BPEL4People**

- **Goal:**
  - Define BPEL extensions for Human user interactions that
    - Allow for the definition of human user interactions as part of a BPEL process
      - Simple scenarios, such as manual approval
      - Complex scenarios where the data input will be performed by the human user
    - Allow for the reuse of independently defined human tasks
BPEL Extensions for Sub-Processes


- WS-BPEL 2.0 Extensions for Sub-Processes
- Key features:
  - Modularization and re-use, in a portable, interoperable way.
  - Allows for the definition of sub-processes that can be reused within the same or across multiple WS-BPEL processes.
  - Invocation of a business process as a sub-process of another business process, such that its lifecycle is coupled to the lifecycle of the parent process.
  - Allows “fragments” to be defined and invoked without having to <invoke> an entire new process with its own context.
  - Describes different invocation scenarios and introduces an appropriate coordination protocol used for interoperable invocation of sub-processes across infrastructures from different vendors.
Resources – BPEL Whitepapers and Specs

- Visit http://ibm.com/developerworks/webservices

  - BPEL4WS 1.1 Specification

  - Paper: “Automating business processes and transactions in Web services: An introduction to BPELWS, WS-Coordination, and WS-Transaction”


  - A series of papers: “Understanding BPEL4WS” (explains the new alphaWorks BPEL editor and runtime)

- Search for “BPEL4WS” and “BPEL” for full list.
Web Services and Interoperability
The Role of the WS-I Organization

http://www.ws-i.org/

- Achieve Web services interoperability
- Encourage Web services adoption
- Accelerate Web services deployment

“WS-I will act as a standards integrator therefore bringing some coherence to the effort carried out concurrently by the W3C, Oasis, OAG and other informal groups.” - Gartner Group
WS-I Organization

- An industry initiative for Web services, founded February 2002
  - Open to any organization committed to Web services
  - Promote and accelerate adoption, deployment
- Focused on promoting Web service interoperability
  - Across platforms, applications, and programming languages
  - Promote a common, clear definition for Web services
- Mission statement

“The Web Services Interoperability Organization is an open industry effort chartered to promote Web Services interoperability across platforms, applications, and programming languages. The organization brings together a diverse community of Web services leaders to respond to customer needs by providing guidance, recommended practices, and supporting resources for developing interoperable Web services.”
WS-I in the Marketplace

- Tool vendors will advertise that their products support development and deployment of WS-I conformant services
- Middleware vendors will advertise that their products support conformant Web service hosting
- Customers will look for WS-I conformance on:
  - products
  - deployed instances, and
  - vertical standard interface descriptions
WS-I Deliverables

- **Use Cases and Usage Scenarios**
  - Use Case – business usage of Web services, Usage Scenario - technical usage of Web services
  - Formalized way to communicate community requirements
  - Specific emphasis on “real-world” use cases and scenarios

- **Profiles**
  - Named sets of specifications at given version levels
  - Constraints, clarifications and conventions about how they are used together

- **Sample Applications**
  - Demonstrated use of Profiles as defined in Use Cases and Scenarios

- **Test suites and supporting materials**
  - Conformance testing tools
  - Test assertions for the profile
WS-I.org Profiles

- A profile is a named set of Web services specifications and their versions.
  - Base specifications are normative

- Profile adds constraints and guidance as to their interoperable usage based upon implementation experience.

- General format is statement, refinement, rationale, examples where appropriate.
Philosophy of the Profile

- No guarantee of interoperability
- Does not address application semantics
- Focus on testable requirements
- Makes strong requirements
  - MUST vs SHOULD
- Never relaxes requirements
- Chooses between multiple mechanisms
- Preserve backwards compatibility
- Focus on interoperability
- Conformance of measurable targets
  - MESSAGE, DESCRIPTION, etc.
- Address issues at application layer
Profile General Format

- Reference section of an underlying specification
- Statement (prose)
  - Say it in English
- Requirement(s)
  - `<RequirementProse xml:lang="en-specese"/>
- Rationale
  - Explain why decision was taken
- Examples
  - Incorrect and correct usage
Basic Profile 1.x is based on these specifications:

- SOAP 1.1
- WSDL 1.1
- UDDI 2.0
- XML Schema
- XML 1.0 (Second Edition)
- HTTP 1.1
- SSLv3
Basic Profile – Technical Highlights

**SOAP1.1**
- Use of SOAP encoding disallowed
- “Trailers” (element content after soap-env:Body) disallowed
- Most spec ambiguity issues resolved in alignment with SOAP1.2
- Use of SOAPAction, soap-env:actor clarified

**WSDL1.1**
- Limited to use of rpc/literal and document/literal
- SOAP/HTTP binding required
  - Other bindings out of scope, but *may* be used
- Schema errata fixed
  - Spec treated as normative
- Exclude use of wsdl:import for XSD files
- Numerous spec clarifications
Basic Profile – Technical Highlights (Continued)

- **UDDI2.0**
  - Require WSDL1.1 as description language
  - Established category to identify WS-I conformant entities

- **Security**
  - May use SSLv3 (HTTP/S)
  - HTTP1.1 Basic Auth
  - Identify risks and countermeasures within Basic Profile
    - Mapped these risks/threats to use cases and scenarios

- **XML Schema**
  - Any valid XSD constructs may be used (all, choice, sequence, etc.)
  - Recommend use of xsi:nil xs:nillable to designate NULL values

- **HTTP1.1**
  - Clarify use of HTTP response status codes
    - soap:Fault == 500, redirect == 307
  - Cookies permitted, but must not be required
Attachments Profile 1.0
http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html

- Composes with Basic Profile 1.1

- SOAP Messages with Attachments
  - Based on RFC2557
    - (MIME Encapsulation of Aggregate Documents)

- WSDL1.1 MIME binding extension (sect 5)

- Optional swaRef schema type defined
  - restriction of xs:anyURI with special semantic meaning inferred
SOAP Messages with Attachments Clarifications

- **Root Part**
  - Need not be first part
  - Must be identified with start parameter if not first
- **Encoding of Root Part**
  - Type parameter required
  - Must be text/xml
- **Messages with No Attachments**
  - Serialized as text/xml OR multipart/related and only one part
- **Dereferencing Attachments**
  - Application specific
- **Carrying Additional SOAP Envelopes**
  - Permitted
- **Fault Messages with Attachments**
- **Ordering of MIME Parts**
  - Order carries no semantic, order need not be preserved
Basic Security Profile 1.0 is based on these specifications:

- WS-I Basic Profile
- WS-Security 1.0
- WS-Security Token Profiles

Related WS-I work

“Security Challenges, Threats and Countermeasures”

http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0.pdf
Testing Tools

- Objectives
  - Conformance testing for Web service instances
  - **not** focused on conformance of runtimes and tools

- Two testing tools
  - Message Monitor and Analyzer
  - Java and C# implementations
    - Either can be used regardless of platform

- Deliverables
  - Testing tools architecture document
  - Machine-readable version of profile definition
  - Implementation of testing tools with User’s Guide
Conformance

- Web service instance and artifacts only
  - Not conformance of runtime or development tools
- Conformance is based on Profile spec
- Must be *capable* of passing WS-I Testing Tools
  - Best indicator of conformance with Profile(s)
  - Tools do not cover *all* requirements
- Self-certification process
  - Claimant tests instance and artifacts
  - Others can run test tools to verify claim
  - Resolve conformance bugs through usual update process
WS-I Sample Applications WG Goals

- Define and implement applications using Web services that conform to the WS-I Profiles
- These applications must
  - Validate and demonstrate the Profiles
  - Assist in the creation of test tools
  - Provide usage guidance for web services
- Close collaboration with other WS-I work groups
Sample Apps WG Deliverables

- Functional specifications
  - Use cases
  - Platform neutral architecture specification
  - Test cases

- Applications
  - Source Code
  - Conformance Reports

- Showcase Application
Usage Scenarios for BP 1.0

- One Way Messaging
  - Fire and forget
  - No SOAP response

- Synchronous Message Exchange
  - Blocking Web service invocation
  - SOAP request / response

- Basic Callback
  - Asynchronous call
  - Pair of SOAP requests / responses
  - Application level message correlation
Resources: WS-I.org

WS-I.org
http://www.ws-i.org/

Basic Profile 1.1
http://ws-i.org/Profiles/BasicProfile-1.1.html
http://ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html
http://ws-i.org/Profiles/AttachmentsProfile-1.0.html

Basic Security Profile 1.0 (DRAFT)
http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

Whitepapers:
“First look at the WS-I Basic Profile 1.0”

"New WS-I Profiles Explained"

"First look at the WS-I Usage Scenarios"

“Security Threats, Challenges and Countermeasures”
http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0.pdf
Industry Profiles
Industry Profiles

Basic B2B Profile

- WS-ReliableMessaging Feb 2005
- WS-Addressing Aug 2004

WS-I Basic Security Profile 1.0

- WS-Security 1.0
  - UserPassword Token Profile
  - X509 Cert Token Profile
  - REL Token Profile?
  - SAML Token Profile?
  - Kerberos Token Profile

WS-I Basic Profile 1.1

Simple SOAP Binding Profile 1.0

- SOAP 1.1
- WSDL 1.1
- UDDI 2.0
- HTTP 1.1
- XML 1.0 Second Edition
- XML Schema 1.0
RAMP Profile 1.0

WS-Addressing (Aug 2004)
WS-ReliableMessaging (Feb 2005)
WS-SecureConversation (Feb 2005)

WS-I Basic Security Profile 1.0

• Username Token Profile
• X.509 Token Profile
• SAML Token Profile
• REL Token Profile
• Kerberos Token Profile

WS-I Basic Profile 1.1

Simple SOAP Binding Profile 1.0

• SOAP 1.1
• WSDL 1.1
• UDDI 2.0
• XML 1.0 Second Edition
• XML Schema 1.0
• HTTP 1.1
RAMP Toolkit
http://alphaworks.ibm.com/tech/ramptk

- Reliable Asynchronous Messaging (RAMP) Toolkit

- Posted to alphaWorks on Oct 14th 2005

- Includes:
  - Run time environment
  - Configuration tools
  - Demonstrations/samples/documentation
### RAMP Monitor

![RAMP Monitor](image)

#### Client

- RequestSecurityToken Message
- Message signed and encrypted

#### Server

- SOAP Message Received
- Message decrypted and verified
- RequestSecurityTokenResponse Message
- Message signed and encrypted

- SOAP Response Message Received
- Message decrypted and verified
- CreateSequence Message
- Message signed and encrypted

- SOAP Message Received
- Message decrypted and verified
- CreateSequenceResponse Message
- Message signed and encrypted

- SOAP Response Message Received
- Message decrypted and verified
- SOAP Message
- WS-ReliableMessaging
- Message signed and encrypted
Time for a Break!

END OF PART 2

Thanks for sticking with me, I hope this was useful.