WS-BPEL Standards Roadmap

Web Services – Business Process Execution Language 2.0 and related standards

Dieter König, IBM Senior Technical Staff Member (dieterkoenig@de.ibm.com)

SOA on your terms and our expertise
Abstract

Business Processes not only play a key role in Business-to-Business (B2B) and Enterprise Application Integration (EAI) scenarios by exposing the appropriate invocation and interaction patterns but they are the fundamental basis for building heterogeneous and distributed applications (workflow-based applications).

Web Services Business Process Execution Language (WS-BPEL) provides the language to specify business processes that are composed of Web services as well as exposed as Web services. Business Processes specified via WS-BPEL are portable; they can be carried out by every WS-BPEL compliant execution environment.

This presentation gives an overview of the WS-BPEL language and shows how it can be used to compose Web services. It provides highlights of WS-BPEL, including structured activities, correlation, compensation, and fault handling. Finally, the OASIS WS-BPEL Technical Committee work, the current status of the standard, and an outlook on follow-on activities is presented.
WS-BPEL 2.0 and Related Standards

WS-BPEL 2.0 Extensions for People (BPEL4People)

WS-BPEL 2.0 Extensions for Sub-Processes (BPEL-SPE)

WS-BPEL 2.0 Extensions for Java (BPELJ)

Service Component Architecture (SCA)
SCA Assembly Model
SCA WS-BPEL Client and Implementation Model
WS-BPEL 2.0

Web Services
Business Process Execution Language
WS-BPEL 2.0

- Motivation
- OASIS and WS-BPEL
- Main Concepts
- Examples
- Status and support
Motivation

• Application integration is a key problem facing businesses
  – Intra enterprise integration (Enterprise Application Integration)
  – Integrating with partners (Business Process Integration)

• Web services → move towards service-oriented computing
  – Applications are viewed as “services”
  – Loosely coupled, dynamic interactions
  – Heterogeneous platforms
  – No single party has complete control

• Service composition
  – How do you compose services in this domain?
Two-level Programming Model

• Programming in the large
  – Non-programmers implementing flows
    • Flow logic deals with combining functions in order to solve a more complex problem (such as processing an order)

• Programming in the small
  – Programmers implementing functions
    • Function logic deals with a discrete fine-grained task (such as retrieving an order document or updating a customer record)
Process Usage Patterns

• Aiming for a single approach for both …
  – Executable processes
    • Contain the partner’s business logic behind an external protocol
  – Abstract processes
    • Define the publicly visible behavior of some or all of the services an executable process offers
    • Define a process template embodying domain-specific best practices
Process Model Requirements

- Portability and Interoperability
- Flexible Integration
  - Rich, and easily adaptable to changes in the services it is interacting with
- Recursive, type-based composition, enables …
  - third-party composition of existing services
  - providing different views on a composition to different parties
  - inter-workflow interaction
  - increased scalability and reuse
- Separation and composability of concerns
  - Decoupled from the supporting mechanisms (quality of service, messaging frameworks)
- Stateful conversations and lifecycle management
  - Can carry multiple stateful long-running conversations
- Recoverability
  - Business processes, and in particular long running ones, need a way to build-in fault handling and compensation mechanisms to handle and recover from errors
WS-BPEL enables …
- Defining business processes as coordinated sets of Web service interactions, recursively into new aggregated Web services
- Defining both abstract and executable processes
  - Abstract processes for e-commerce specifications
  - Executable processes provide a model to integrating enterprise applications
- Creating compositions of Web services
  - Composition based on abstract descriptions

WS-BPEL provides portable, interoperable process models

WS-BPEL comes from …
- Strong roots in traditional flow models
- Plus many concepts from structured programming languages
- All laid on top of WSDL and core XML specifications
- Merges WSFL and XLANG concepts
WS-BPEL Specifications

• BPEL4WS 1.0 (7/2002)
  – Original proposal from BEA, IBM, Microsoft
  – Combined ideas from IBM’s WSFL and Microsoft’s XLANG

• BPEL4WS 1.1 (5/2003)
  – Revised proposal submitted to OASIS
  – With additional contributions from SAP and Siebel

• WS-BPEL 2.0
  – Currently in OASIS undergoing standardization
  – Public Review draft specification available
WS-BPEL in the WS-* Stack

- WS-BPEL
- WSDL, Policy, UDDI, Inspection
- Security
- Reliable Messaging
- Transactions
- Coordination
- SOAP (Logical Messaging)
- Other protocols
- Other services
- XML, Encoding
- Quality Of Service
- Description
- Transport and Encoding
- Business Processes

You are here
WS-BPEL 2.0

- Motivation
- OASIS and WS-BPEL
- Main Concepts
- Examples
- Status and support
Getting the Players Together

BPEL4WS 1.1

OASIS

WS-BPEL 2.0

(*) BPEL4WS 1.1 authors, May 2003

Advancing E-Business Standards Since 1993
• Charter of the WS-BPEL Technical Committee
  – Standardize …
    • Common concepts for a business process execution language for usage patterns including both the process interface descriptions and executable process models
  – Explicitly do not address …
    • Bindings to specific hardware/software platforms and other mechanisms required for a complete runtime environment for process implementation
• > 280 committee members, including observers
  ~ 30 active voting members, attending weekly calls
• Issues Process
  – List of all issues available at
    http://www.choreology.com/external/WS_BPEL_issues_list.html (original list - completed)
    http://www.choreology.com/external/WS_BPEL_review_issues_list.html (public review list)
WS-BPEL Design Goals

- Business processes defined using an **XML-based language**
- **Web services** are the model for process decomposition and assembly
- **The same orchestration concepts** are used for both the **external** (abstract) and **internal** (executable) views of a business process
- Both **hierarchical and graph-like** control regimes are used, reducing the fragmentation of the process modeling space
- An **identification mechanism for process instances** is provided at the application message level
- The **basic lifecycle mechanism** is in implicit creation and termination of process instances.
- A long-running transaction model is defined to support **failure recovery** for parts of long-running business processes
- Language built on **compatible Web services standards in a composable and modular manner**
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
WS-BPEL Language Structure

- Process
- Partner links
- Data handling
- Properties and correlation
- Basic and structured activities
- Scopes
BPEL and WSDL

- BPEL processes are exposed as WSDL services
  - Message exchanges map to WSDL operations
  - WSDL can be derived from partner definitions and the role played by the process in interactions with partners
Recursive Composition

- BPEL processes interact with WSDL services exposed by business partners

Interfaces exposed by the BPEL process

Interfaces consumed by the BPEL process

BPEL processes interact with WSDL services exposed by business partners.
Composition of Web Services

Service A

A’s WSDL

Service P

P’s WSDL

Service B

B’s WSDL

Partner Link Type

Φ

Partner Link Type
Partner Links

- Partner link: instance of typed connector
  - Partner link type specifies required and/or provided portTypes
  - Channel along which a peer-to-peer conversation with a partner takes place
Scoped variables typed as WSDL messages or XML Schema elements/types

Activities’ input and output kept in scoped variables

Assignment activities move data around
Properties and Correlation

- How to identify stateful instances via stateless Web service interfaces?
- Messages in long-running conversations are correlated to the correct process instance
  - Typed properties defined in WSDL are named and mapped (aliased) to parts of several WSDL messages used by the process
Basic Activities

- **receive**: Do a blocking wait for a matching message to arrive
- **throw**: Generate a fault from inside the business process
- **reply**: Send a message in reply to a formerly received message
- **rethrow**: Forward a fault from inside a fault handler
- **invoke**: Invoke a one-way or request-response operation
- **exit**: Immediately terminate execution of a business process instance
- **assign**: Update the values of variables or partner links with new data
- **wait**: Wait for a given time period or until a certain time has passed
- **validate**: Validate XML data stored in variables
- **compensate**: Invoke compensation on all completed child scopes in default order
- **empty**: No-op instruction for a business process
- **compensateScope**: Invoke compensation on one completed child scope
- **extensionActivity**: Wrapper for language extensions
Structured Activities

- Contained activities are executed in parallel, partially ordered through control links.

- Contained activities are performed sequentially in lexical order.

- Contained activity is repeated while a predicate holds.

- Contained activity is repeated until a predicate holds.

- Block and wait for a suitable message to arrive (or time out).

- Contained activity is performed sequentially or in parallel, controlled by a specified counter variable.

- Select exactly one branch of activity from a set of choices.

- Associate contained activity with its own local variables, fault handlers, compensation handler, and event handlers.
<sequence>
  <receive .../>
  <flow>
    <sequence>
      <invoke .../>
      <while ...>
        <assign>...</assign>
      </while>
    </sequence>
    <sequence>
      <receive .../>
      <invoke ... />
    </sequence>
  </flow>
  <reply .../>
</sequence>
Scopes and Handlers

• Scope
  – Local variables
  – Local partner links
  – Local correlation sets
  – Set of activities (basic or structured)

• Handlers
  – Event handlers
    • Message events or timer events (deadline or duration)
  – Fault handlers
    • Dealing with different exceptional situations (internal faults)
  – Compensation handler
    • Undoing persisted effects of already completed activities
  – Termination handler
    • Dealing with forced scope termination (external faults)
• Business processes defined in BPEL represent stateful Web services
  – When a process is started, a new instance is created according to the process definition
  – The creation and destruction of BPEL process instances is by design implicit
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
1. A customer asks for a loan, providing name and amount info
2. Two services are involved:
   a) A risk assessor which can approve the loan if the risk is low
   b) A loan approver which checks the name and approves/disapproves the loan
3. The reply is returned to the customer
Structured Authoring Style

if \(\text{buyer} < \text{seller}\)

else

receive (buyer)

receive (seller)

assign "success"

assign "failure"

reply (seller)

reply (buyer)
Parallel Processing

- Flow
- Parallel forEach
- Scope
- Event handler

Advancing E-Business Standards Since 1993

www.oasis-open.org
Dynamic Partner Link Assignment

Process A
- assign EPR
- invoke service
- receive response
- partner link

Process B
- receive request
- assign EPR
- partner link
- invoke callback
- PortType-B
- PortType-A
- A-role
- B-role
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
WS-BPEL

• Portable, interoperable process model for long running business processes
• Flexible integration of Web services
  – WSDL abstract interfaces alone used to define composition
    • Enables two levels of adaptive behavior
      – Abstract partners can be bound to actual services at runtime
      – The process can choose a protocol for communicating with the service at runtime
  – Services whose data definitions do not match can be composed
    • Data transformations can be inlined in process definition
What’s new since BPEL4WS 1.1?

• Conditional behavior – `if {elseif} [else]
  – Replacing the BPEL4WS 1.1 switch activity
• Repetitive execution – `repeatUntil`
  – Like BPEL4WS 1.1 while activity with at least one iteration
• Processing multiple branches – `forEach`
  – Sequential or parallel iteration controlled by a counter variable
• Reversing work – `compensate` and `compensateScope`
  – Compensate all child scopes or one specific child scope
• Propagating faults – `rethrow`
  – Rethrow the fault caught in a fault handler
• Validating data – `validate`
  – Explicit XML schema validation of WS-BPEL variable content
• Immediately ending a process – `exit`
  – Replacing the BPEL4WS 1.1 terminate activity
• Adding new activity types – `extensionActivity`
  – Designated WS-BPEL extension point for new activity types
What’s new since BPEL4WS 1.1?

- Variable initialization
- XSLT for variable transformations
  - New XPath extension function `bpel:doXslTransform(...)`
- XPath access to variable data
  - XPath variable syntax `$variable[.part]/location`
- XML schema variables in Web service activities
  - Usability enhancement for WS-I compliant doc/lit-style WS interactions
- Scope termination handler
- Locally declared messageExchange
  - Internal correlation of receive/pick/onEvent and reply activities
- Extension namespace declarations
  - Mandatory and optional extensions
- Abstract processes
  - Common base (syntax) and profiles (semantics)
  - Opacity and omission of activities, expressions, attributes
WS-BPEL Adoption: Products

- Active Endpoints ActiveBPEL Designer
- Active Endpoints ActiveBPEL Enterprise Server
- BEA WebLogic
- bexe BPEL Execution Engine (open source)
- Cape Clear Orchestrator
- FiveSight Process eXecution Engine (PXE)
- IBM WebSphere Studio Application Developer – Integration Edition
- IBM WebSphere Business Integration – Server Foundation
- IBM WebSphere Integration Developer
- IBM WebSphere Process Server
- Microsoft BizTalk Server
- MidOffice BPEL Editor (open source)
- MidOffice BPEL Engine (open source)
- OpenLink Virtuoso Universal Server
- OpenStorm ChoreoServer
- Oracle BPEL Process Manager
- Parasoft BPEL Maestro
- SAP NetWeaver
- SeeBeyond eInsight BPM
- Twister (open source)
WS-BPEL Application Areas

- Business Process Design
- Autonomic Computing
- Grid Computing
- Semantic Web
WS-BPEL 2.0 Schedule

• Status (December 2006)
  – First public review completed
  – Approved draft for second public review

• Next steps
  – Second public review
    → December 2006
  – OASIS standard
    → Early 2007
WS-BPEL Follow-on Work

- **BPEL4People**
  - Human user interactions

- **BPEL Subprocesses**
  - Based on a coordination protocol

- **BPELJ**
  - Inline Java code in activities and expressions

- **Currency with related standards**
  - WSDL 2.0, XPath 2.0, XQuery, etc.
WS-BPEL Resources

  http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.pdf
• OASIS Technical Committee
  http://www.oasis-open.org
• Info aggregator sites
  – Wikipedia
    http://en.wikipedia.org/wiki/BPEL
  – BPEL Resource Guide
    http://bpelsource.com
• Numerous books and conference papers
• Analyst reports
• BPEL4WS 1.1
WS-BPEL 2.0 and Related Standards

WS-BPEL 2.0 Extensions for People (BPEL4People)

WS-BPEL 2.0 Extensions for Sub-Processes (BPEL-SPE)

WS-BPEL 2.0 Extensions for Java (BPELJ)

Service Component Architecture (SCA)
SCA Assembly Model
SCA WS-BPEL Client and Implementation Model

© 2006 IBM Corporation
WS-BPEL Extensions for People (BPEL4People)

- 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
- Independently defined human tasks reused in a BPEL process
WS-BPEL Extensions for People – Human Task

Translation Process

receive document

automatic translation

manual translation

reply translation

Web Service Endpoint

Abstract Organizational Group
WS-BPEL Extensions for People – Logical People Group

Brochure Creation Process

- **create**
- **approve**
- **revise**

"Authors"

Deployment Descriptor

- Select staff
  - Where qualification = "tech writer"

"Approvers"

Deployment Descriptor

- Select staff
  - Where responsibility = "marketing"

Org Database

- Departments
  - Department1
    - Member1
    - Member2
  - Department2
    - ...

- Users
  - Group1
    - Member1
    - Member2
  - Group2
    - ...

- Roles
  - Role1
    - Member1
    - Member2
  - Role2
    - ...
WS-BPEL Extensions for People – Composition Options

- Inline Task Definition
  - Task can only be used inside of process context
  - Task is interoperable as well as portable

- Standalone Task Definition
  - Task can be used outside of process context
  - Communication with task is based on coordination protocol
  - Task is interoperable as well as portable

- Invoke activity
  - Task is just a Web service with no additional constraints
  - Easy to switch between performing the task by humans or programs
WS-BPEL Extensions for Sub-Processes (BPEL-SPE)

- The services used by an activity may be implemented as another process
  - In “basic” BPEL this is hidden from the engine(s) running both processes
  - If this is transparent both processes can be coupled, i.e. can be considered as constituents of a larger “virtual” process
    - Their lifecycle can be coupled
    - They may share context

- Benefits
  - Reuse of process knowledge
    - Same business problems
    - Best practices
  - Enhanced modeling capabilities
    - Decomposition of large processes, outsourcing
    - Composition – modular process structure
  - Large-scale distributed execution
    - Process “parts” on different platforms
      - …within one enterprise or even across enterprises (“B2B processes”)
WS-BPEL Extensions for Sub-Processes

WFMS 1

Parent Process

Local Sub-Process

Inline Sub-Process

Remote Sub-Process

WFMS 2
WS-BPEL Extensions for Sub-Processes – Invocation

call ...
  partnerLink=...
  requestOperation=op

Parent Process

Sub-Process

receive (input)

reply (output)

Partner Link Type
WS-BPEL Extensions for Sub-Processes – Invocation

call ...
  partnerLink=...
  requestOperation=op
  responseOperation=op'

EPR(op')

op'

op

receive (input)

invoke (output)
WS-BPEL Extensions for Java (BPELJ)

- Include inline Java code in BPEL processes
  - Activities
  - Conditions
  - Variable initialization
- Orchestrate long-running interactions with Java components
- Support advanced transactional capabilities
  - Atomic Scopes and Atomic Processes
WS-BPEL Extensions for Java – Inline Java Activities

- Extension Activity

```
<process name="purchaseOrderProcess" bpelj:xmlBinding="bpelj:DOM3" ...>
  <variables>
    <variable name="justificationDoc" type="lns:justificationDocument"/>
    <variable name="po" type="lns:POMsg" bpelj:xmlBinding="bpelj:SDO1.0"/>
  </variables>
  <sequence>
    ...
    <extensionActivity>
      <bpelj:snippet>
        // Get the approver using SDO accessor
        Approver approver = po.getApprover();
        // Get the approver's comments
        NodeList commentNodeList =
            justificationDoc.getElementsByTagName("approverComment");
        ...
      </bpelj:snippet>
    </extensionActivity>
    ...
  </sequence>
</process>
```
WS-BPEL Extensions for Java – Inline Java Expressions

- **Condition (Boolean Expression)**

```xml
<if>
  <condition
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:java1.4">
    widget.equals(getVariableProperty("PO", "productName"))
  </condition>
  ...
  <else>...</else>
</if>
```

- **Variable initialization (General Expression)**

```xml
<variable name="salesTax" type="xsd:float">
  <from
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:java1.4">
    subtotal * taxRate
  </from>
</variable>
```
WS-BPEL 2.0 and Related Standards

WS-BPEL 2.0 Extensions for People (BPEL4People)

WS-BPEL 2.0 Extensions for Sub-Processes (BPEL-SPE)

WS-BPEL 2.0 Extensions for Java (BPELJ)

Service Component Architecture (SCA)
SCA Assembly Model
SCA WS-BPEL Client and Implementation Model
SCA Assembly Model

- Unified declarative model for service assemblies
  - Service components with interfaces (WSDL port types or Java interfaces)
  - Dependency resolution and configuration
  - Declarative policies for infrastructure services
    - Security, Transactions, Reliable messaging
- Binding model for multiple access methods
  - SOAP/HTTP, JCA, JMS, Java RMI/IIOP, …
- Language-independent way to expose implementations as services
  - Java, WS-BPEL, PHP, other languages
- Facilities for dynamic service configuration
  - Properties / protocols / qualities of service
  - Profiles
SCA Assembly Model – Component
Service Component Architecture

SCA Assembly Model – Composite

Service
- Java interface
- WSDL PortType

Property setting

Component A

Wire

Component B

Wire

Reference
- Java interface
- WSDL PortType

Binding
- Web Service
- SCA
- JCA
- JMS
- SLSB
- ...

Binding
- Web Service
- SCA
- JCA
- JMS
- SLSB
- ...

Wire
SCA Assembly Model – Deployment
SCA WS-BPEL Client and Implementation Model

- Specifies how WS-BPEL can be used with SCA
  - Use any valid WS-BPEL 2.0 process definition as the implementation of a component within SCA
  - Use WS-BPEL to implement any SCA Component Type that uses only WSDL interfaces to define services and references, possibly with some SCA specific extensions used in process definition.
  - Create a WS-BPEL process definition that uses SCA extensions and generate an SCA Component Type and use that type within an SCA assembly
    - Some SCA capabilities (such as properties and multi-party references) can only be used by WS-BPEL process definitions that use SCA extensions.
SCA WS-BPEL Client and Implementation Model

WS-BPEL inside 😊
SCA WS-BPEL Client & Implementation Model
Services and References
SCA WS-BPEL Client & Implementation Model
Endpoint Reference Assignment
SCA WS-BPEL Client & Implementation Model
Multi-Valued References
SCA WS-BPEL Client & Implementation Model
Properties

- property currency
  - variable
    - currency
  - flow
    - ...
WS-BPEL 2.0 and Related Standards

WS-BPEL 2.0 Extensions for People (BPEL4People)

WS-BPEL 2.0 Extensions for Sub-Processes (BPEL-SPE)

WS-BPEL 2.0 Extensions for Java (BPELJ)

Service Component Architecture (SCA)
SCA Assembly Model
SCA WS-BPEL Client and Implementation Model

© 2006 IBM Corporation