IBM Software Group

WebSphere® Process Server V6.0
WebSphere Integration Developer V6.0
WS-BPEL Support Details

@business on demand.

This presentation will provide an overview of the WS-BPEL specification and support provided for it in WebSphere Process Server V6.0 and WebSphere Integration Developer V6.0.

Goals

- Describe the IBM enhancements and additions to the WS-BPEL specification

2

The goal of this presentation is to describe in detail, the enhancements to the WS-BPEL specification that are supported by IBM.
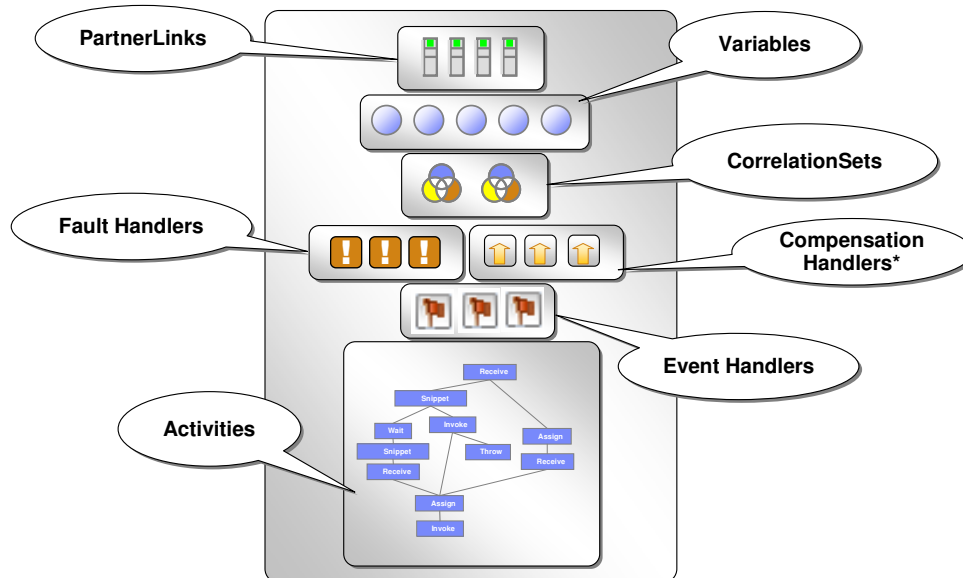
Agenda

- Details of WS-BPEL Support
- Summary

This section of the presentation is to explain the specifics of WS-BPEL support.

Main Elements/Concepts of a BPEL Process

PartnerLinks

Variables

CorrelationSets

Fault Handlers

Compensation Handlers*

Event Handlers

Activities

Receive
Snippet
Wait
Invoke
Snippet
Throw
Assign
Receive
Receive
Assign
Invoke
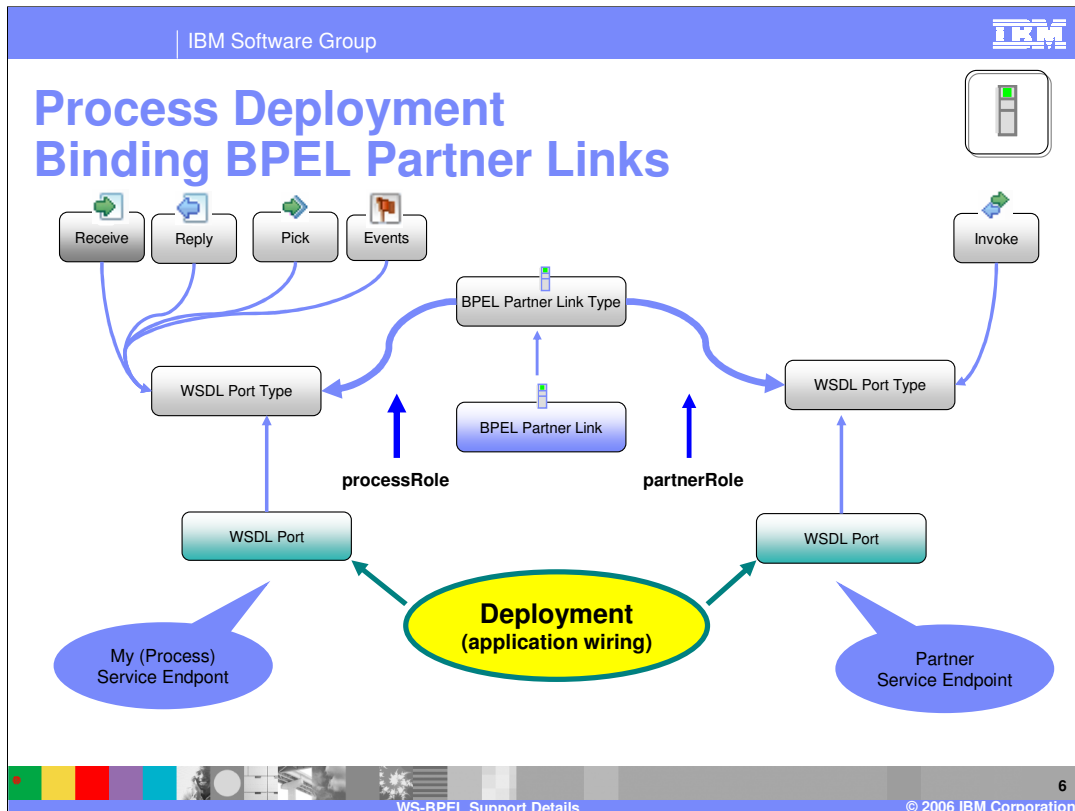
WS-BPEL Support Details

© 2006 IBM Corporation

4

The BPEL specification describes a number of areas for defining business processes. These areas detail how a business process is organized and how it might be run. The primary elements or concepts are PartnerLinks, Variables, Correlation sets, Fault handlers, Compensation handlers, Event handlers, and Activities. Each one of these will be covered in more detail in this presentation.

Partner Links and Roles

- With Business-to-Business scenarios, the business process of one business entity needs to interact with the business processes of another
  - ▸ BPEL introduces the business partner as a construct in the business process
  - ▸ Partner links may be one way or two-way
  - ▸ The sending and receiving sides of a partnership are distinguished by the **role** they play, for example, buyer/seller, producer/consumer, and so on
- Partner Links only supported at the process level

The BPEL specification utilizes a service-oriented approach to define business processes. The various steps that make up the business process are exposed as services. Within the definition of the business process, there is some type of representation of that service, known as a partner link, which is essentially an endpoint that represents the service you are going to call. The partner link is only defined with the interface of that service. No implementation details are included in the partner link information. Partner links also utilize roles, which are used to define, in an abstract manner, the relationship between the partner link and a particular process. For situations where there are multiple communications between a process and a service, you might have different roles being played. The endpoint or service is identified by a particular role. Partner links are usually one-way relationships and roles really only become significant in two-way relationships. For WebSphere Process Server V6.0, partner links are only supported at the process level while the BPEL specification outlines support for partner links to be specified at other levels.

Process Deployment
Binding BPEL Partner Links

Receive Reply Pick Events Invoke

BPEL Partner Link Type

WSDL Port Type   BPEL Partner Link   WSDL Port Type

processRole   **partnerRole**

WSDL Port   WSDL Port

My (Process)
Service Endpont

Deployment
(application wiring)

Partner
Service Endpoint

6
WS-BPEL Support Details   © 2006 IBM Corporation

The BPEL process uses partner links not only to define services that are invoked from the business process, but also to define the interface of the BPEL process. This includes how clients or outside entities contact and interact with the business process. That is why a WSDL port is defined for the invoke action (right side of slide) and another defined for the receive and reply activities (on the left side of the slide) that are used by clients to send in data and receive a response. Pick and events are also used for sending in data and are defined with a service interface. At deployment time, interface information for a particular step is mapped or bound to the actual implementation. Similarly, the protocol (such as Web services, JMS, or SCA) that is used to call the business process is also defined at deployment time. Implementation details are not needed or used within the definition of the BPEL process.

Variables

Enhanced in V6
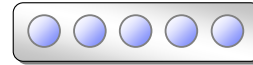
- Hold data that represents the state of a process
  - ▸ May be received from or sent to partners
  - ▸ Can be specified as input or output variables for invoke, receive, and reply activities
  - ▸ May hold state data related to the process and never exchanged with partners
- Typically associated with WSDL message types
  - ▸ XSD data types also available

7

© 2006 IBM Corporation

Variables represent the state of the business process. Variables store information about data coming in to the business process that is then reused and sent to the various steps that are part of the business process. They are also used to hold the data that is returned from a service or to hold internal information such as counter values for iterative processes. The variable capability has been enhanced to allow the use of XSD data types, as suggested by the BPEL specification. WSDL message types can also be used now to facilitate the service oriented architecture that BPEL builds upon. The business process steps are defined as an interface that has messages and those messages can be used as the basis for your variable types.

Correlation of Messages to Instances

- The Business Process Execution Engine:
  - Accepts an incoming message
  - Correlates the message parts to the parts in the correlation set
  - Compares the values of each part to the initialized values in the set
  - If all the parts and all the values match, the message is directed to the appropriate business process instance

Order Number

Customer Number Order Number

Order Number Customer Number Timestamp
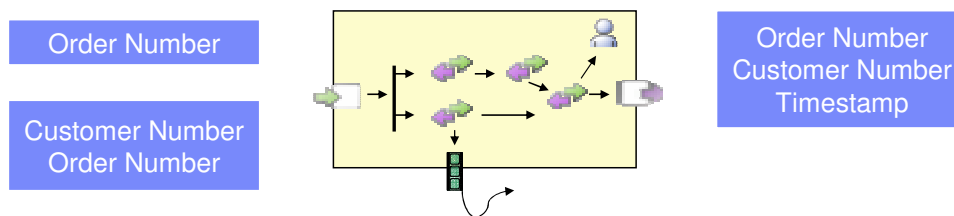
In a long running process there are points in the process where state data is captured and the process goes into 'waiting' state, waiting for some sort of response, such as a manager approving an expense. To make sure that the approval is associated with the right expense, some unique identifier is used.

The unique identifier could be generated and assigned by the system or it could be a combination of business relevant data, such as a timestamp and employee number combination.

Through the lifetime of the process this correlation id may change. BPEL addresses this with correlation sets.

An important point to understand about the correlation ID is that it is something that is known by the outside activity without any notification provided to the outside activity ahead of time.   Also, the correlation ID is not something that is generated by the Process Choreography runtime (although the Business Process Container does enforce unique instances). The correlation ID is required at the time the business process instance is started. Typically the correlation ID value is some unique value known to the starter of the business process or is generated before the Business Process is started. When the business process is started the correlation ID is passed on the Input message.

Consider a Stock Trading process where many different customers may be looking to buy or sell stocks. Each customer would contact their broker with the intention to trade a particular stock. The broker would start an instance of the Stock Trade process with a unique identifier of the customer number.   The customer number is obviously known to the customer and unique, and when it comes time to buy or sell a stock, the customer number can be specified on the request and matched to a particular running instance.

Correlation Sets

- Correlation Sets route an incoming message to the existing business process instance that should handle it
  - Each inbound message must contain a value or set of vaules that uniquely identify its contents with a specific process instance
    - Examples: customer number, order ID, trade ID, Invoice Number
- A Correlation Set has a unique name (called an alias), which:
  - Defines a name for the set of parts
  - Lists the message parts to be included in the set
- At run-time, the *values* of the message parts in the received message determine the business process instance to which that message will be routed
  - The unique values must be initialized the first time the correlation set is used
- Correlation Sets only supported for processes

Correlation Sets match incoming requests to the correct business process instance. For example, if there are 1000 loan applications in progress and the credit department is sending in credit report messages for multiple applications, correlation sets ensure that the message is sent to the correct instance. Within a correlation set, you will define the parts of an incoming message that indicate uniqueness. These will then map to some value in the business process state. The messages can then be associated with the instance that contains the same unique values. Note that correlation sets are provided from application data. The business process must be designed using some value that is unique within the enterprise. Examples include a social security number, order number, customer number or other value that is defined as part of the business process. Correlation sets can consist of a single value or multiple values and there can be multiple correlation sets for long-running business processes. As the business process state changes, there might be different unique identifiers that must be used to identify an instance. The correlation set must be initialized, typically at the beginning of a long-running business process, if there is any interaction coming into that business process later in the cycle. WebSphere Process Server V6.0 supports correlation sets at the process level, even though the BPEL specification supports correlations sets at other levels.

Fault Handler

- Every fault handler is associated with a scope
  - ▶ Handles faults thrown in their scope

- A scope can have multiple fault handlers
  - ▶ Different kinds of faults can have different fault-handling activities

- The handler specifies the activities to be performed when a fault has been thrown

- A fault reaching a fault handler means that the remaining processing in the current scope cannot continue
  - ▶ All active work within the scope will be terminated

Fault handlers are designed to detect and signal a failure in the running of a business process. They are designed to catch primarily business exceptions such as inventory falling below a set limit or a customer requesting an item that is no longer stocked. The fault handler can catch these types of failures and perform an operation to either handle the exception and continue processing or generate a failure to be handled in some other way. A fault handler is associated with an activity or with a scope that encompasses a set of activities. When a fault is caught by a handler, the activity is ended and control is transferred to the fault handler. Fault handlers can be nested similar to the way that scopes are nested. Scopes can contain a set of activities and one scope can contain another scope (or scopes) and each of these scopes can in turn have its own fault handler. If a fault is not caught at the current scope, it will propagate up to the next scope and be caught by the fault handler for that scope and handled at that level or processing could continue up the hierarchy. If the fault reaches the upper-most scope of the business process and is not handled, that indicates a failure of the entire business process. In this case, the client or some other fault monitoring tool is expected to handle the fault. The Fault Handler can also catch runtime failures. However, it is better not to catch specific runtime problems because that exposes implementation details in the business process.

Compensation Handlers

Enhanced in V6

- BPEL Compensation Handler support provided for nested scopes
  - Compensation Support provided by WebSphere Business Integration Server Foundation V5.1 also still supported
- Compensation Handlers contain actions which perform reverse operations for a particular scope or activity
- Compensate activities must explicitly call Compensation Handlers
  - No automatic invocation of compensation handler
- Only available for long running business processes
  - Non-interruptible processes have a single transaction which has rollback capabilities
- Processing in a compensation handler is part of the normal processing
  - Access is available to variables and partnerlinks of scope

WebSphere Process Server and WebSphere Integration Developer both include support now for using compensation handlers, which are defined under the BPEL specification as well as the IBM extension compensation support provided by WebSphere Business Integration Server Foundation V5.1. This support was not included in V5.1 releases, although it was part of BPEL. During this timeframe, IBM had a compensation strategy of it's own and that strategy is still supported in V6.0. However, the BPEL compensation is now supported and you should these rather that the IBM implementation whenever possible. Compensation means to perform some sort of opposite operation to undo something that has completed.   Because short-running, noninterruptible business processes consist of transactions that can be rolled back in the event of failure, compensation only applies to long-running business processes comprised of multiple activities. The roll back capability can be used during compensation as part of an undo operation, and the IBM compensator extends this capability to allow for roll back of short-running noninterruptible processes that do not have transaction awareness. Compensation handlers provide roll back capability for long-running processes and include all the steps that must be taken to roll back the completed portion of a process. For example, if a loan application process includes some sort of red flag warning based on information from the credit agency, it might be necessary to begin undoing some of the steps that have already completed. Some of the steps might include notifying the office where the application originated that the loan process has been cancelled. Compensation handlers can only be invoked by a compensate activity, which will be discussed later. During compensation for a particular scope, all the variables and partner links that are defined are available so that the process can be restored to its original state. In WebSphere Process Server V6.0 compensation handlers are only available for nested scopes. While the BPEL specification outlines other areas where compensation handlers can be used.

Event Handlers

New V6

- Event Handlers accept outside requests for a particular scope
  - ▸ Requests may be one-way or request response
- Correlation sets must be enabled on event handlers to direct event to correct business process instance
- Any type of processing can occur within the Event Handler
- Alarm events handlers available to perform actions after time period has expired
  - ▸ Timer starts when scope is entered

12

Support for event handlers, originally defined under the BPEL 1.1 specification, is new in WebSphere Process Server V6.0. Event handlers accept requests from external clients into the business process.

Event handlers are defined at a particular scope of activity and events can be accepted during the entire cycle of the business process or they can be limited to a specific scope.

Event handlers use correlation to provide an interface to outside entities and to match up requests to the correct instance. When an event is received by a handler, a set of steps defined in the handler will be run on a separate thread from the business process itself. The event handler does have access to business process state information such as variables and partner links. When the event handler activities are complete, the thread goes away.

Alarm events can also be set up within an event handler. Alarm events are triggered after a predefined period of time and the timer begins when the scope containing the event handler is reached.

Event handlers are only available for a long-running business processes.

Overview of WS-BPEL Basic Activities

IBM Software Group

Receive
Wait for a message to arrive. Optionally start a new process instance when the message arrives.

Reply
Reply to a message that was received.

Invoke
Invoke a one-way or a request-response operation offered by a partner.

Assign
Update the values of variables with new data.

Throw
Generate a fault from within the business process.

Rethrow
Rethrow a fault which was caught
New V6

Wait
Wait for a given period or until a certain time has passed.

Compensate
Call a compensation handler
New V6

Terminate
Immediately terminate the process instance.

Empty
A "no-op" instruction in the business process.

WS-BPEL Support Details

13

© 2006 IBM Corporation

Here is an overview of the basic BPEL activities which are used primarily to carry out the different steps in the business process. The Receive activity is an asynchronous activity which can be added to a business process. The Reply activity is used in conjunction with a Receive activity when a request/response is used by an outside action to communicate with the business process. Invoke activities are the main activities which interact with outside entities and service providers. Assign activities are the primary way that data is transformed and moved from one variable to another. Throw activities are for indicating some type of failure has occurred within the business process and it needs to either be handled in the business process or passed back to the client because no further processing can occur.   If a failure is caught a Rethrow activity can be used to throw the failure without needing to use an Assign activity to move the failure contents from one failure to another. The Wait activity allows for a business process to stop and wait for a specific amount of time. The Terminate activity can be used in a business process were all processing should end with no way of compensating or performing reverse processing on the instance. The Empty activity acts as a placeholder in a business process for a subsequent activity that may be implemented later. When the Empty activity is reached, the business process continues without stopping, treating the activity as a no op. The Empty activity can be changed to any of the other activity types.

Each of these will be covered in detail on the next few slides.

Receive and Reply Activities

Activity

- Receive
    - ‣ Receive a message sent to a business process
        - Starts a new business process
        - Restarts an existing process via Correlation
    - ‣ Request may be synchronous or asynchronous

Receive

- Reply
    - ‣ Reply to a message that was received
    - ‣ Only available as a response to a synchronous request.
    - ‣ Can return either
        - response message
        - fault message

Reply

14

The support for Receive and Reply activities has not changed from WebSphere Business Integration Server Foundation V5.1. Receive is a means of sending information into a business process and is typically used at the beginning of the process to accept data and start the instance. A correlation set can be specified, establishing a unique instance from information in the message. Reply activities can only be used in conjunction with a Receive activity and can return a response message indicating success or a fault message. Receive and Reply activities can be used in long-running interruptible or in short-running non-interruptible business processes. Even in a long running process with a synchronous interface, there are asynchronous implementations, such as JMS, that can be used to carry out request/response types of interactions.

Invoke Activity

Activity

- Invoke
  - ▸ Invoke a one-way or a request-response operation offered by a partner
  - ▸ Calls another business process

Invoke

- IBM Extensions to Invoke Activity
  - ▸ transactionalBehavior – explicit checkpointing
  - ▸ continueOnError – enter a stopped state after infrastructure fault
  - ▸ expiration – Java™ timeout expression

The invoke activity is used for performing the various steps of a business process. This activity calls out to the different services by means of the partner links, using services or request/response calls. With a long-running service or communications that utilize JMS queues, the invoke activity can be used. It is not necessary to differentiate between a call to a synchronous or an asynchronous activity within the business process. The business process engine can handle this for you by looking at the how the invoke and partner link are bound to the implementation. There are a number of IBM extensions to the Invoke such as the transaction behavior of a long-running process. Transactional behavior determines whether a transaction should be committed around this invoke or if it should run in it's own transaction or if it should participate in an existing transaction. In some cases, grouping invokes or other types of snippets with invokes can result in better performance because a transaction does have to be committed for each step of the process. The continue on error flag allows the business process engine to continue on even if there is a failure for certain situation. For example, a publish operation where no partner is reached is not considered a critical failure and should not cause processing to halt. Expiration can also be set for long-running invocation on the invoke activity. If no response is received in a specified period of time, the business process can raise a failure.

Assign Activity

Activity

Assign

- Update the values of variables with new data.
- Values can be copied from a source to a destination
- Snippets can be used:
  - to build a complex message part
  - when using custom properties for message parts
- Full XPath 1.0 support *Enhanced in V6*

prepAutoClean

| 1 | From: | Variable | ▼ | To: | Variable | ▼ |

From:
- ⊞ ● AutoIn : ClipBG
- ⊞ ● ManualOut : ClipBG
- ⊟ ● InClipBG : ClipBG
  - ⊞ ☑ schema
  - ☑ package : anyType
  - ☑ changeSummary : anyType
  - ☑ properties : anyType
  - ☑ eventSummary : anyType
  - ☑ verb
  - ⊞ ☑ Clip : Clip

To:
- ⊞ ● PublishIn : ClipBG
- ⊞ ● ManualIn : ClipBG
- ⊟ ● AutoIn : ClipBG
  - ⊞ ☑ schema
  - ☑ package : anyType
  - ☑ changeSummary : anyType
  - ☑ properties : anyType
  - ☑ eventSummary : anyType
  - ☑ verb
  - ⊞ ☑ Clip : Clip

New
Delete
Query:
Query:
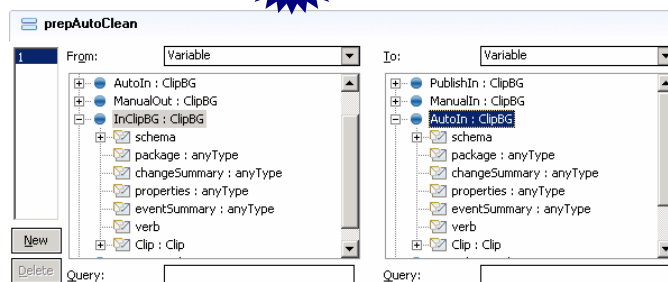
Assign activities are one of the primary means of working with variables defined within the business process. These variables represent the state of the business process as well as information that will be sent to and received from the different services. Variables are based upon business objects, used to define the data types. Different services have different messages that must be populated with data and the assign activity is the primary way to move information from one variable or message to another. Assign activities perform basic mapping of information. For more complex types of mappings, the use of snippets, which can be Java based or created using a visual means, is supported. There is also a business object map capability in the BPEL editor, which can be used for mapping specific business objects. In WebSphere Process Server Version 6.0, support for XPath 1.0 is provided, allowing you to specify an XPath query string or to drill down into more complex types of business objects.

Throw

Activity

Throw

- Signals a situation that the immediate logic can not handle
  - ▸ Faults have a name, and an optional message, which can contain additional information about the nature of the fault.

- May be caught by a Fault Handler in a parent Scope
  - ▸ The fault name, the optional message, or both for use in decision making

- Uncaught faults signal a failed business process state
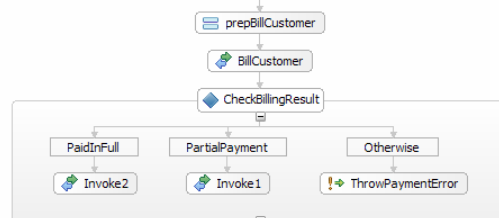
The throw activity signifies a problem that the business process logic cannot handle. A fault is thrown within the business process and a signal that there is a problem is sent to an outside entity or a higher level in the business process. There are a number of built-in fault messages that can be used with this activity or you can create custom faults that can be thrown to the next higher scope or the scope enclosing the throw activity if it contains a fault handler. If the fault is not caught and handled before it reaches the uppermost level, a business process failure is signified.
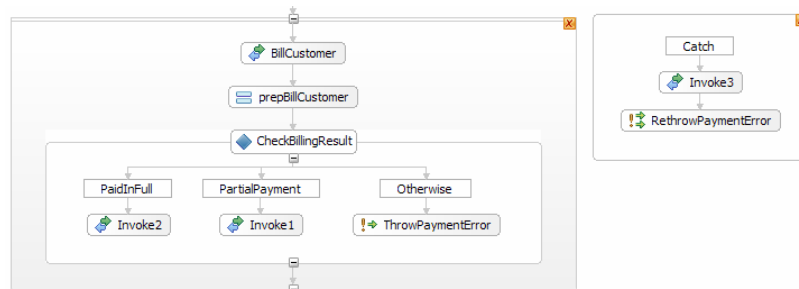
The rethrow activity is newly supported in WebSphere Process Server V6.0 and is defined under the BPEL specification. With a rethrow, if a fault is caught, it is rethrown to the next higher scope. In the BillCustomer example, if the CheckBillingResult returns PaidInFull or PartialPayment, processing continues. If the customer does not pay or overpays and neither of these conditions are met, this signals a failure and a fault will be thrown signifying the fault cannot be handled using the current logic. The catch would then be for the specific fault and notification can be made and the account updated in some way. However, instead of issuing a new fault, the initial fault is reused and passed on, simplifying fault notification. In WebSphere Business Integration Server Foundation V5.1, you would have had to use an assign or a new throw activity and move the data from the caught fault into the new fault message, requiring more custom programming.

Wait Activity

Activity

Wait

- A **Wait** activity stops the business process for a specific amount of time:
  - ▸ Duration in seconds
  - ▸ By calendar date
  - ▸ XPath
  - ▸ By Java expression (IBM extension)

  Enhanced in V6

- Only available for long-running business processes

```
          Sequence1
              │
        prepBillCustomer
              │
          BillCustomer
              │
     Waitfor24HourShippingDelay
              │
         prepShipping
              │
          ShipOrder
```
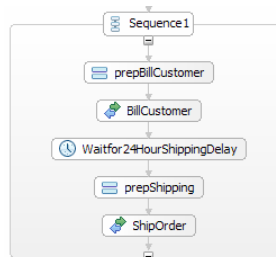
The wait activity can be used to stop a business process on the current processing path for a specific period of time or until a specific date has been reached. The duration or date can be either a hard coded value or calculated dynamically. XPath support, which is new in WebSphere Process Server V6, can be used, or a more complex calculation can be performed using Java. This function is an IBM extension. Any length of time can be used for a wait activity, therefore it is only available for use with long-running interruptible processes.

Compensate

New V6

Activity

- Invoke a reverse operation on an activity or a scope encompassing activity

Compensate

- Invokes Compensation Handler of Invoke activity or scope

- Only scopes or activities which have completed normally may be compensated

- Compensate activity only available in a Fault Handler or Compensation Handler

BillCustomer

RefundCustomer

CompleteOrder

RestockItems
CompensateBillCustomer

ShipOrder

Catch All
CompensateCompleteOrder

20

WS-BPEL Support Details

© 2006 IBM Corporation
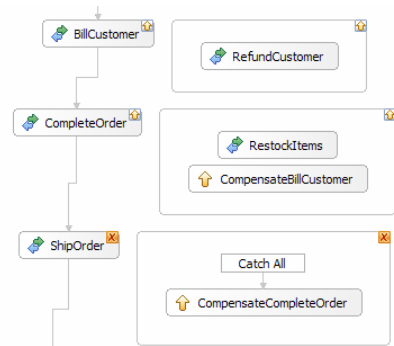
The compensate activity is defined in the BPEL specification and is supported in WebSphere Process Server V6.0. This activity signifies that an activity or scope of activities that has completed must be undone. This occurs when a failure situation is reached; the compensate activity can only be used within a fault handler or within another compensation handler. The compensate activity points to a specific compensation handler, which contains the steps necessary to undo the completed action. For example, if an order fails to ship, an error is returned. The failure, signaling that there is a problem that requires some processing to be undone, can be caught using catch all (no need to catch the specific fault). The compensation activity will be set to point to the appropriate compensation handler and the logic in that handler will take over, processing the activities in the handler, and the items will be restocked. The next compensation handler will then be called, which would handle refunding the customer or crediting their account. This compensation handler has access to all the partner links in the business process and to the variables that were set before the action began. Using these variables, the compensation handler can undo the completed activities or any other processing required to return the instance of the Business Process to the original state. In the example on the slide, although the compensation handlers in this example are at the invoke, they can be put at scopes to include multiple activities as well.

Terminate Activity

Activity

- A Terminate activity ends processing of business process instance immediately (hard stop)

Terminate

- All processing ends, including parallel processing paths

- Recovery of terminated processes (compensation) is unavailable

21
© 2006 IBM Corporation
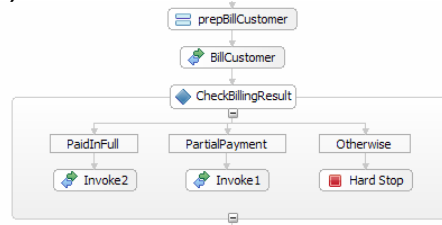
The terminate activity is the equivalent of pulling the plug on an electrical device. This is a hard stop that signifies a failure has occurred or the business process should be halted at this point immediately. If a process is terminated, no recovery is possible (or wanted) and no compensation is possible.

Overview of WS-BPEL Structured Activities

Sequence
Multiple activities that are performed sequentially

Flow
Multiple activities that are performed concurrently

Switch (Choice)
Select one activity branch from a set of choices

Pick (Receive Choice)
Synchronize two activities in a Flow to enforce a particular processing order

While
Repeat an activity until a boolean condition has been met
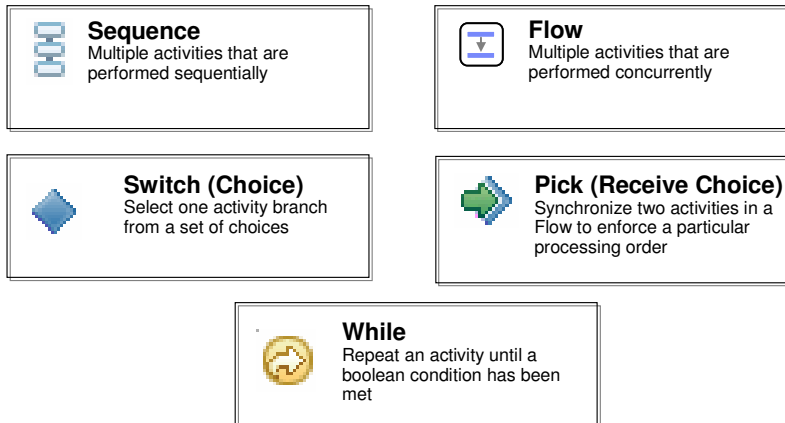
Here is an overview of structured activities, which are used to perform structured programming within a business process such as sequencing activities or running them in parallel with the flow activity. The Switch, also known as Choice activity, provides a way to perform conditional logic in a business process. The Pick or Receive Choice activity is very similar to a Receive activity in that the business process will stop and wait at the activity until one of n operations, defined on the Pick, is called. Finally, the While activity is a way to repeat a group of activities based on a Boolean condition which is evaluated prior to each iteration.
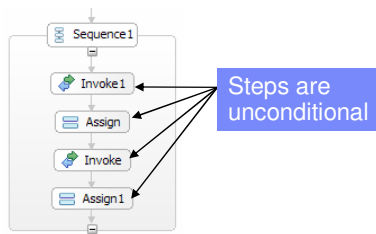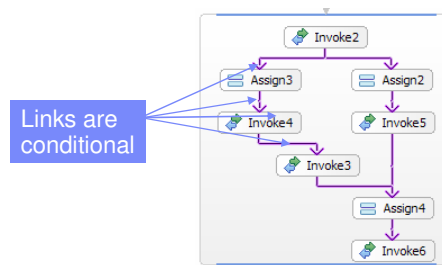
Shown here are the sequence and flow activities. A sequence consists of activities occurring one right after another. Each activity must complete successfully before continuing on to the next one. A flow allows a business process to have parallel processing of activities. Within the parallel path, there are links specified. These links can include logic to check whether processing should continue. This logic can use variable state information from the business process to determine if processing should proceed. Flow activities utilize multiple threads and are designed to be used with long-running interruptible business processes. In a short-running noninterruptible process, activity would be processed sequentially in random order.

Switch (Choice) Activity

Activity

Switch

- A **Switch** implements a decision chosen from multiple conditions
  - ▶ Condition can be a Visual Expression, Java Expression (IBM Extension), XPath, True or False

Enhanced in V6

- The first case condition to evaluate to true is run

- If no case resolves to true the Otherwise case is run

- If the Otherwise is omitted, an implied Otherwise with condition true is used

Check Operation

Create	Read	Update	Delete	Otherwise
Invoke	Invoke1	Invoke2	Invoke3	Invoke4
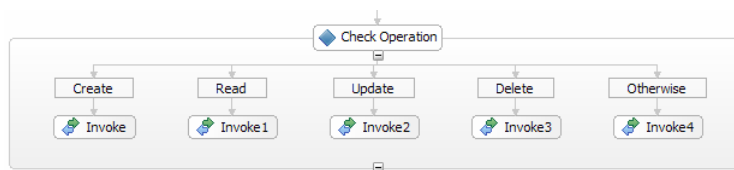
24

WS-BPEL Support Details

© 2006 IBM Corporation

A switch or choice activity is similar to a programming case statement where conditions are defined. Whichever condition resolves to true first determines what processing takes place. Conditions are checked from left to right and when one evaluates to true, the activities defined under that condition will be run. An otherwise clause can also be used in the case that none of the conditions evaluates to true. If no otherwise clause is specified, it is implicitly generated by the business execution engine as defined under the BPEL specification. Enhancements for WebSphere Process Server V6 include support for XPath, which can be used for checking the conditions defined. Java can be used as well as visual expressions that can be created. This support is provided as an IBM extension. You could even hard code in true or false.

Pick (Receive Choice) Activity

Activity

Pick

- A **Pick** activity is a combination of a **Receive** and a **Switch**
- The business process waits for 1 of n possible messages before continuing
- Correlation Set must be specified for each message pick
- Each message pick can have different security permissions
- The OnAlarm property is available to time-out, wait, and continue

  Enhanced in V6

  ▶ Time-out in duration (seconds), by specific date, by calendar, by XPath
  ▶ IBM extension

Order Status

Order Shipped	Order Pending	Order Canceled	Timeout
Invoke	Invoke1	Invoke2	
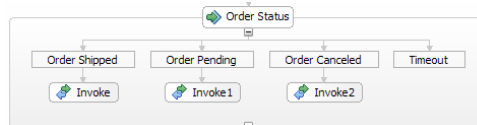
The pick or receive choice activity provides a point within the business process where information can be received from outside entities or from different clients. These points or interfaces can be different in different circumstances. For example, if you are waiting for a message to indicate order status, such as shipped, pending, or cancelled, each status could have different information passed in and use a different interface. When any one of these is called or comes into the business process, processing will continue just from the message that was received and from that line. For example, if the order status of pending came in, only the invoke1 activity would be invoked. Each of these different interfaces can be secured with different permissions. Order shipped might be coming from the warehouse, whereas order cancelled might be coming from a customer service representative and these activities can be secured in different ways. A timeout capability is also available to prevent a stalled process. Enhancements include support for XPath to calculate the alarm and the timeout value. Timeout values can be hard coded or can be determined dynamically using a calendar which is an IBM extension.

While Activity

Activity

While

- The **While** activity will repeatedly run the activities in its scope as long as the *condition* is **true**
  - The *condition* is checked before the first iteration, so it's possible that *none* of the nested activities will be run
- The *condition* may be
  - A visual Expression (IBM extension)
  - A Java Expression (IBM extension)
  - XPath
  - True or False
- A fault (handled or unhandled) will terminate the loop
  - It's more efficient to end the loop by meeting the condition than by throwing an exception

Enhanced in V6

While Order not Filled

ItemAddedtoShipment

prepUpdateItemInOrder

UpdateIteminOrder

prepCheckOrderFill

CheckOrderFill
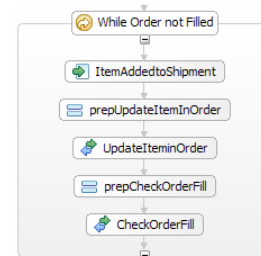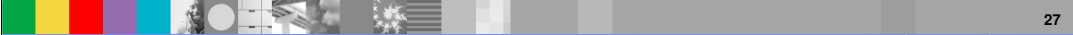
The while activity is used to run activities iteratively. The activities under the scope will be run as long as a defined condition remains true. The condition is evaluated prior to activity processing, so it is possible that none of the while activities will be run. The while activity has been enhanced in V6 to include support for evaluating data using XPath. The while activity is a scope and can have a fault handler associated with it to handle failure of any activity in the while loop that is not caught at the while loop construct level. If a fault does occur and reach the while scope, any processing in that loop that is executing in parallel will halt and the fault handler will be called. This is not the best way to exit a while loop, so you should try to meet the condition of the while loop and exit that way, because there is additional overhead associated with throwing a fault. The use of faults should be reserved for indicating failures in the business process.

Scopes

New
V6

Activity

Scopes

- Provide encapsulation to Partner Links, Variables and Correlation Sets (state)
- Limits Event and Compensation coverage
- Nesting support
  - Variables by the same name are different instances
- Specify Fault, Event, and Compensation Handlers on scopes

Scopes are supported in WebSphere Process Server V6.0 as defined in the BPEL specification. Scopes encapsulate correlation sets and the state of a business process in order to make particular values available to a specific set of activities. Scopes can be established to define event handlers as well compensation handlers. Scopes can be nested as well, allowing variables with the same name to exist in the business process as a unique instance. If a variable is defined at a particular scope, that variable will be accessed when called by name. Variables defined at a parent scope will be available as long as the name is different. Once the scope ends, the variable is no longer available to downstream activities. When a scope with variables is established in the business process editor, the business process editor will only show variables defined in that scope. It will not show variables for the global business process. This is currently a limitation of the business process editor. The global variables are still available, even though only the current scope variables are shown in the editor.

Event Handlers

New
V6

Events

- Provide an active receive for a specific scope
- May be repeated multiple times until scope is completed
- Restrictions on who sends events can be set

28

WS-BPEL Support Details

© 2006 IBM Corporation
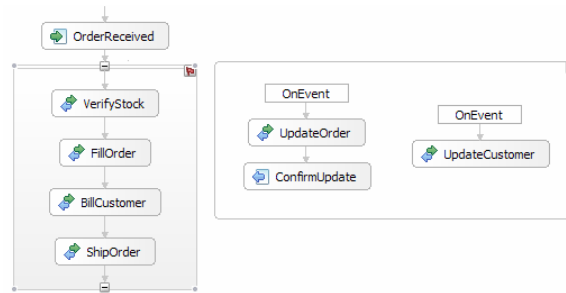
Events are associated with a scope. When the scope is started, events can be received until the scope is completed. This example lists four activities that are grouped into a scope with an event handler defined on the scope. Because these four activities are run at any time, different events can be received. There are two events that could come in while these four activities are run. Each one of those events would initiate a different set of activities. The event can be a one-way operation such as that on the right or a request/response like the event handler on the left. Each event handler must have a correlation set specified in order to insure an incoming message reaches the correct process instance. A separate processing thread with access to partner links and variables within the scope is used for each event. Events can be secured differently, allowing only authorized users to call an event. If an event comes in prior to this event handler becoming available when the scope is reached, the business process engine will hold that event and wait for that scope to be reached. If the event comes in after the scope is ended, a runtime exception will be generated indicating that instance is not available for an event.

Links and Transition Conditions

Enhanced in V6

- Links are conditional processing paths *leaving* an activity

- The boolean condition (true, false, otherwise) derived from:
  - ▶ Visual expression (IBM extension)
  - ▶ Java expression (IBM extension)
  - ▶ Simple (true, false)
  - ▶ XPATH

- If multiple links in a **Flow** activity evaluate to **true,** and the activity is in a long-running process, then the activities may be run in parallel

ReserveFlight

ReserveCar   ReserveHotel
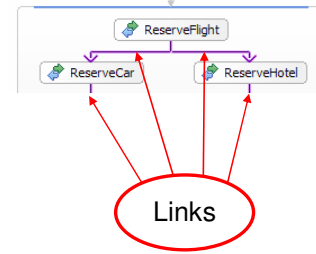
Links

Links and link conditions are associated with flow activities and can include conditional logic to determine if processing should proceed on a particular path. This has been enhanced in WebSphere Process Server V6.0 to include support for XPath. An IBM extension allows conditions to be evaluated using Java programming. A value of true or false can also be hard coded into the link. Parallel processing can be performed during a long-running business process if multiple conditions evaluate to true.

Join Conditions

Enhanced in V6

- Joins are conditional processing paths *entering* an activity
- Join conditions are specified on the target activity
- Join conditions may be specified:
  - ‣ Visual (IBM extension)
  - ‣ Java expression (IBM extension)
  - ‣ Simple (IBM extension)
  - ‣ XPath
- A join condition can be:
  - ‣ **Any –** logical *or*
  - ‣ **All –** logical *and*
- Control flow enters the activity following a join
  - ‣ if the join condition is **All** and *all* of the links are **true**
  - ‣ If the join condition is **Any** and *one* of the links is **true**
- If none of the links evaluate to **true**, the activity is skipped and Dead Path Elimination occurs

Merge Assign   Merge Assign

Merge

Return Glass and Pay Bill
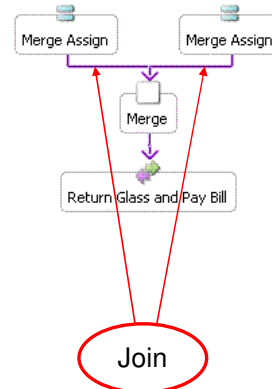
Join

Join conditions are logic to indicate if the process should wait for processing from all paths or just a single path before continuing to run an activity in a flow. In some situations, it might be appropriate to spawn off different paths, but it is not critical that both paths complete before continuing on with processing. Join conditions can be used to accomplish this. For the Join conditions, there are two settings.   A value of All will require all links to be true before continuing and a value of Any will only require that one link evaluates to true before continuing. If there is a single link which resolves to false or one of the links resolves to false with a Join condition setting of All, then Dead Path Elimination is used. The activity will be skipped and any links coming out of the activity will have a negative value. This allows for the business process to continue, but not run incorrect activities. Join Conditions have been enhanced in WebSphere Process Server V6.0 to include support for XPath. An IBM extension provides for conditions to be evaluated using the Java programming language.

Human Tasks

Enhanced in V6

Activity

Human Task

- IBM Extension to WS-BPEL

- Allows specific steps in a business process to be assigned and completed by a person

- Supports escalations, transfer, suspend and resume capabilities

- Integration with different user registries

- Separate component Human Task Manager provides the runtime support
  - ▸ Usable in situations when all of the Business Process Choreography support is not required

WS-BPEL Support Details

Human tasks are an IBM extension to the BPEL specification which allows specific steps in a business process to be assigned and completed by an individual. Human tasks in business processes are wired to other activities directly. The human task support is provided by the Human Task Manager which is a stand-alone component and can be used in situations where all of the business process capabilities are not required. With human tasks, escalation chains can be declared, work items can be transferred, suspended or resumed for robust support in an enterprise environment. Integration with user registries allows for tasks to be assigned to different groups of individuals.

Java Snippets

Enhanced in V6

Activity

Snippet

- IBM Extension to WS-BPEL

- Special type of invoke activity which performs inline processing

- Implementation is local to business process
  - No actual call is made

- Available for working with and modifying variables
  - Examples: Increment Value, Check for unstable state and throw fault, Canonicalize or Normalize variables data for internal processing

- Visual programming editor or Java editor available for specifying implementation

32

© 2006 IBM Corporation

Snippets are an IBM extension to the BPEL specification. Formerly known as a Java snippet, this has been extended to include support for other types of snippets as well. WebSphere Process Server V6.0 includes a visual snippet editor, or you can program in Java directly. Snippets can be used to perform more robust processing on variables and business process state when the intended action cannot be performed using assign activities. An example is concatenation or parsing a string such as changing the format of person's name. Snippet activities should not be used for performing any type of external calls or processing.

Agenda

- Details of WS-BPEL Support
- Summary

WS-BPEL Support Details

33

© 2006 IBM Corporation

In summary,

Summary

- WS-BPEL provides a standard-based language for defining the business process model independent of the implementation

- Business Process Choreographer (BPC) provides the runtime to suport WS-BPEL and addition extensions provided by IBM for executing processes in an enterprise environment on WebSphere Process Server

34

WS-BPEL provides a description language for defining business processes independent of the implementation.   WebSphere Process Server and WebSphere Integration Developer V6.0 support the proposed BPEL 2.0 specification and address a number of issues that have been raised with it. There are many enhancements over V5.1, like event and compensation handlers, compensate activity, and rethrow. These enhancements all build upon the core support provided in WebSphere Business Integration Server Foundation V5.1.

Trademarks, Copyrights, and Disclaimers