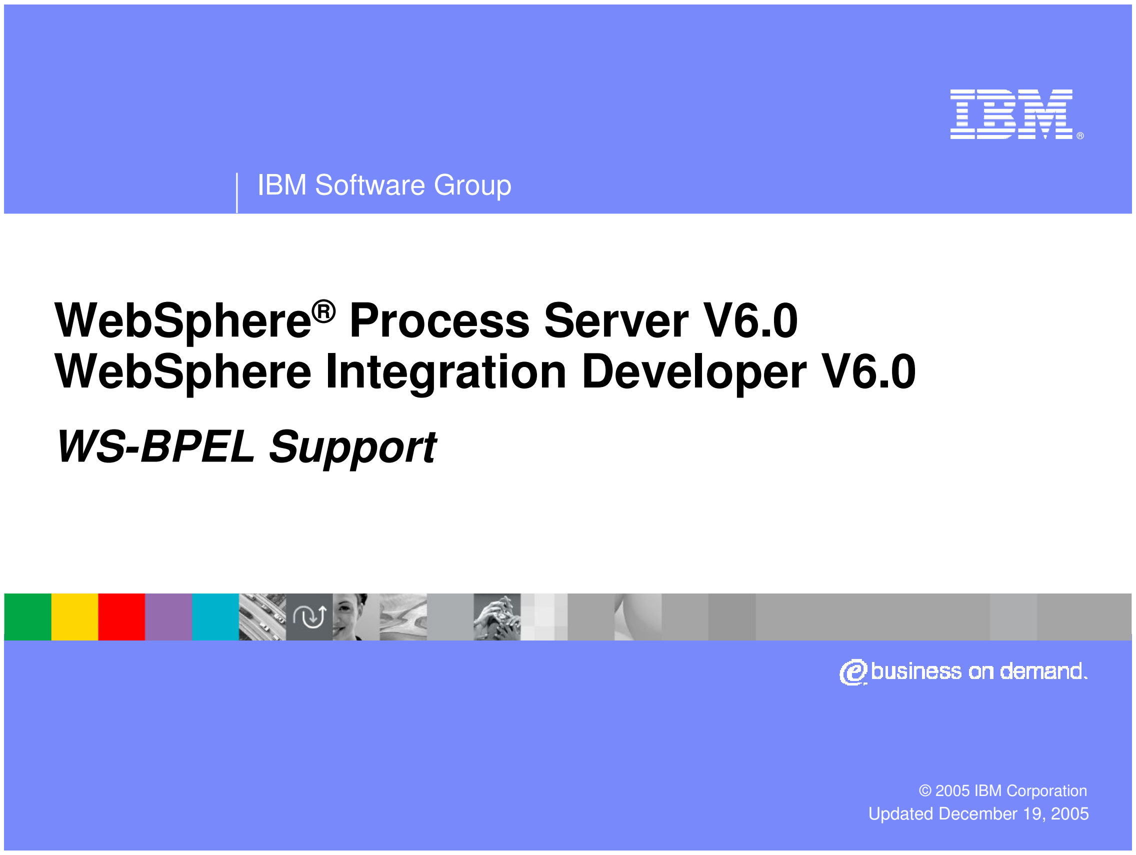


IBM Software Group

WebSphere® Process Server V6.0
WebSphere Integration Developer V6.0

WS-BPEL Support

business on demand.

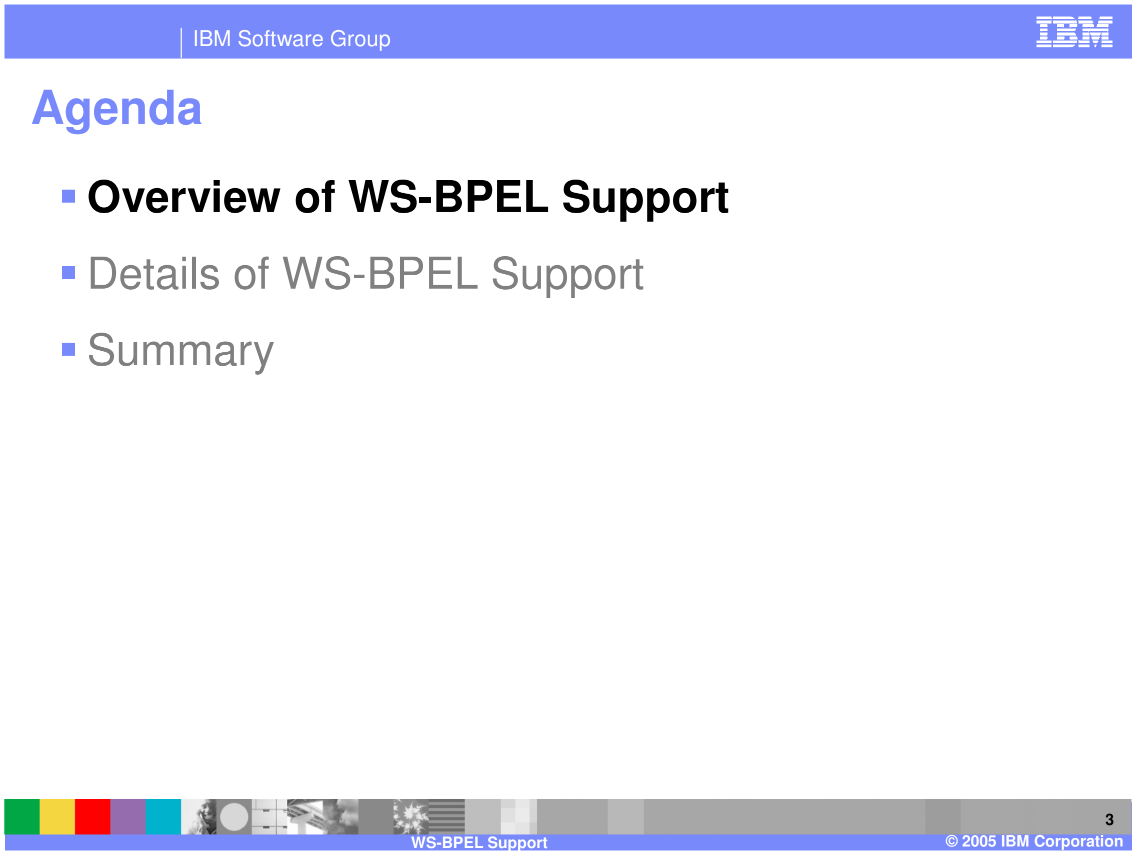
Updated December 19, 2005

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

- Introduce Web Service Business Process Execution Language (WS-BPEL)
- Describe the IBM enhancements and additions to the WS-BPEL specification

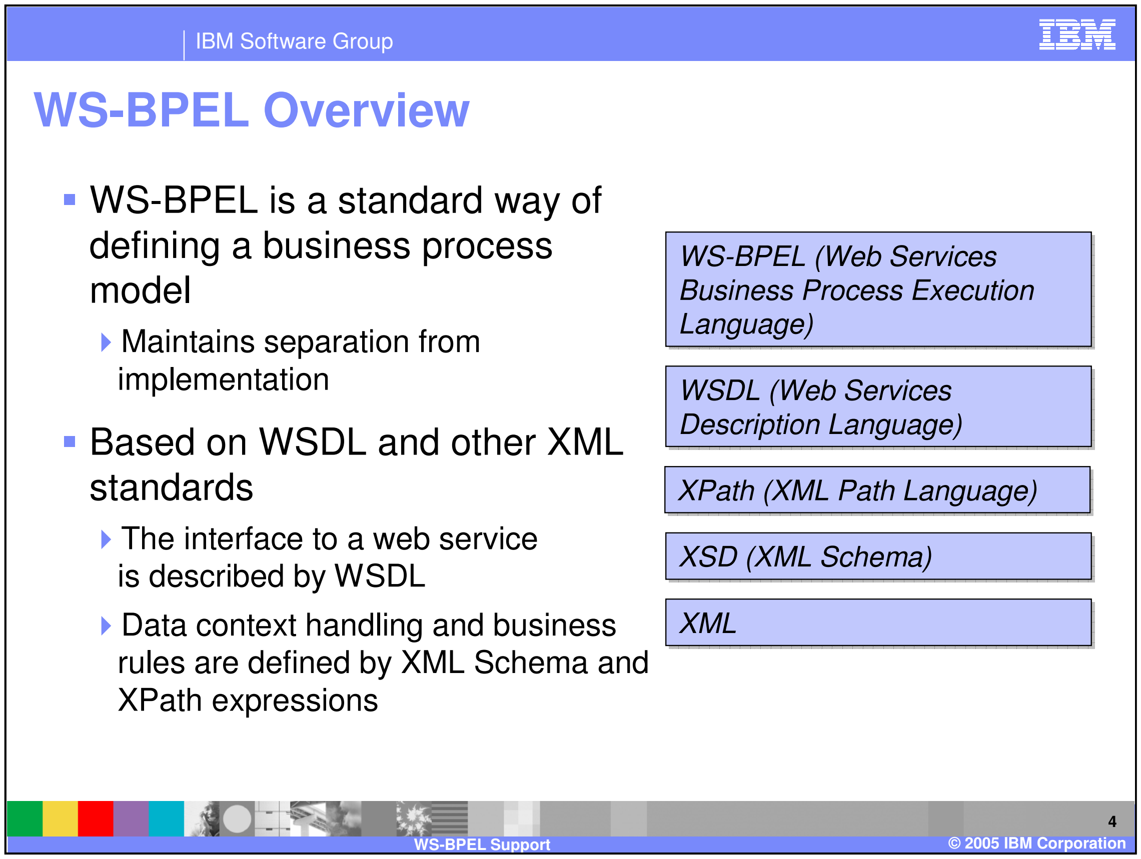
The two primary goals for this presentation are to introduce the Web Service Business Process Execution Language specification also known as WS-BPEL or BPEL and to identify and describe the enhancements to the specification that are supported by IBM.



Agenda

- **Overview of WS-BPEL Support**
- Details of WS-BPEL Support
- Summary

This section will provide an overview of the support for the WS-BPEL specification and support.



WS-BPEL Overview

- WS-BPEL is a standard way of defining a business process model
  - Maintains separation from implementation
- Based on WSDL and other XML standards
  - The interface to a web service is described by WSDL
  - Data context handling and business rules are defined by XML Schema and XPath expressions

WS-BPEL (Web Services Business Process Execution Language)

WSDL (Web Services Description Language)

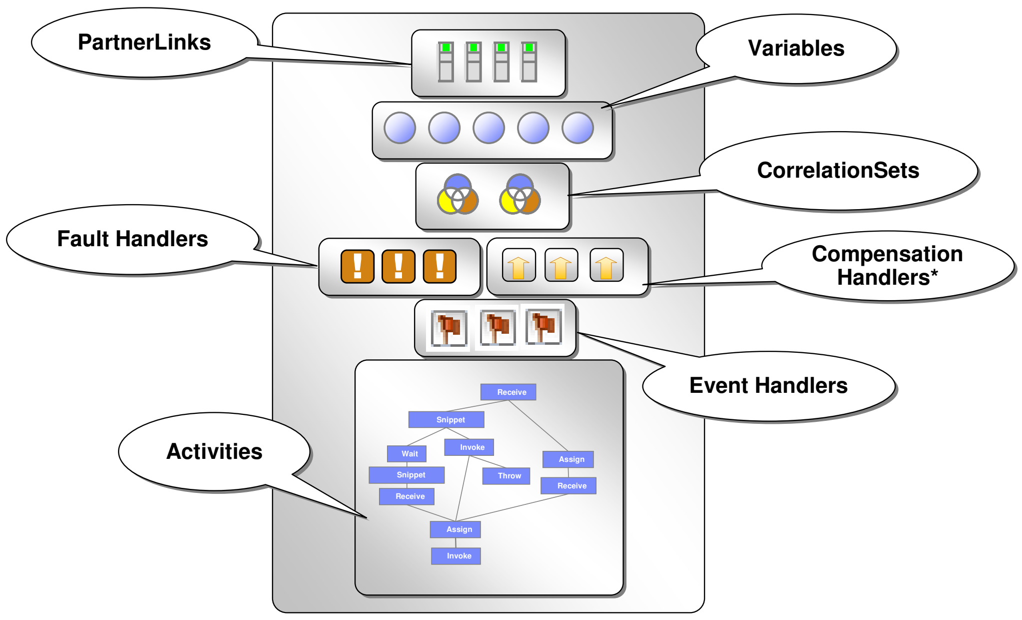
XPath (XML Path Language)

XSD (XML Schema)

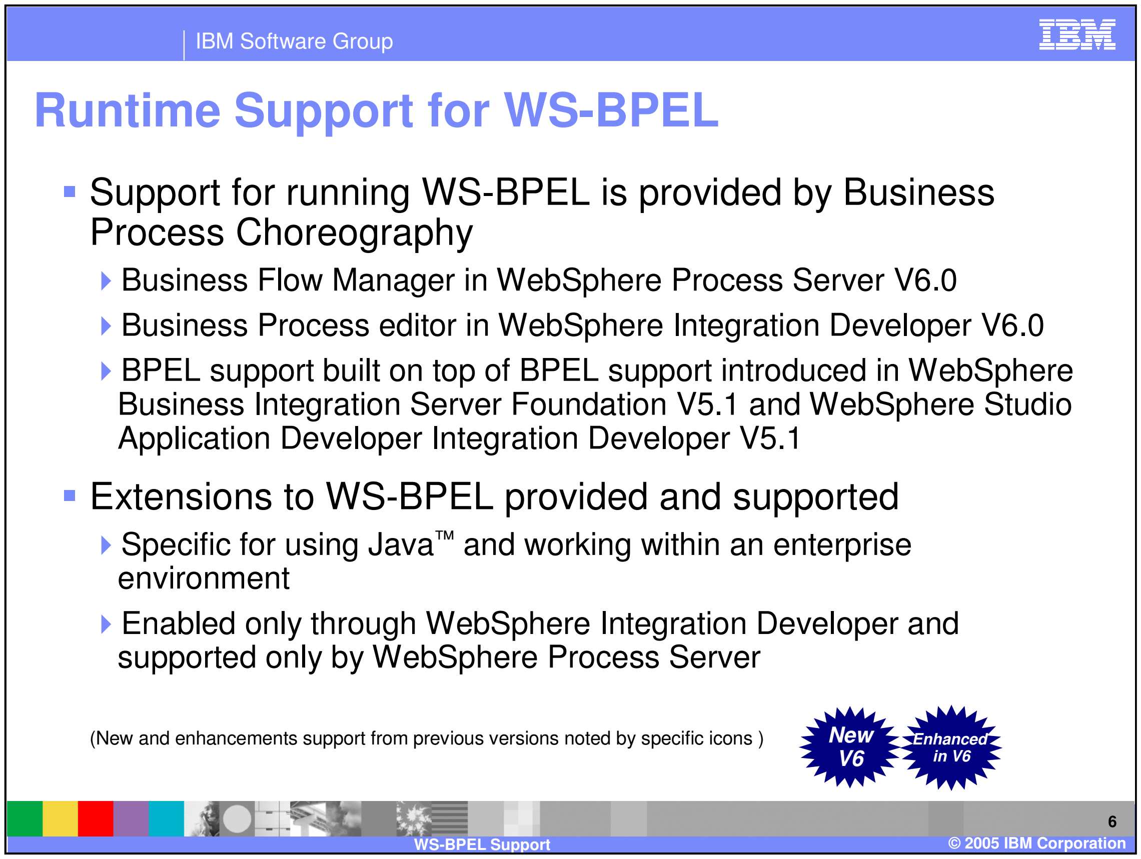
XML

The WS-BPEL standard is finalizing at the version 2.0 level at the time of this recording. The standard allows business process models to be defined independent of the implementation, keeping the processes separate from the underlying infrastructure or technology. This fits nicely with the concept of a service oriented architecture (SOA) where interfaces are kept separate from implementations. WS-BPEL uses services and service interfaces as a means of defining the connections between the different steps. For example, in a business process with 5 steps, interfaces on the steps indicate the type of data that will be passed and potentially received and the type of operation to be performed. The WS-BPEL standard utilizes other industry standards such as Web Services Description Language (WSDL) to define steps and interfaces and XML Schema Definition (XSD) to define data structures. The BPEL process is actually an XML file that is interpreted at runtime to indicate the sequence of steps that make up a business process. XPATH support is also provided as a primary means of working with data objects passed between steps.

Main Elements/Concepts of a BPEL Process



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 executed. 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 later in this presentation.

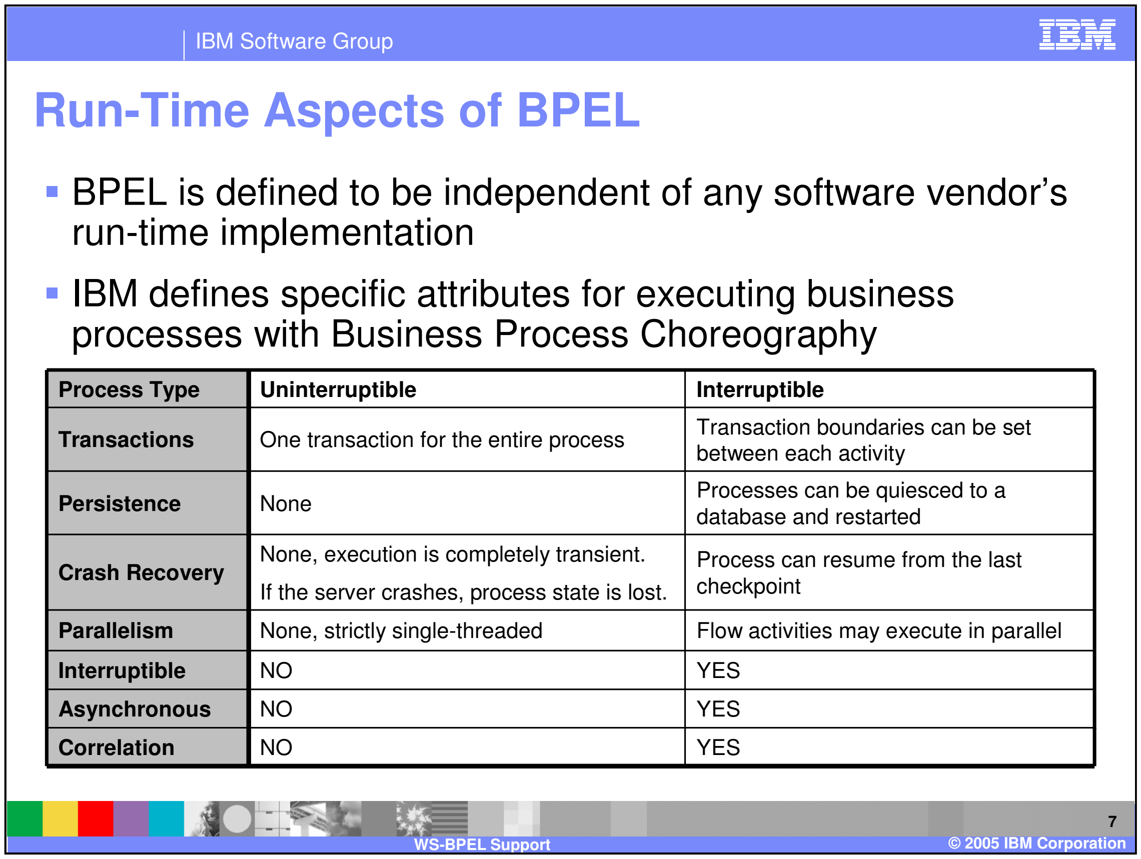


Runtime Support for WS-BPEL

- Support for running WS-BPEL is provided by Business Process Choreography
  - Business Flow Manager in WebSphere Process Server V6.0
  - Business Process editor in WebSphere Integration Developer V6.0
  - BPEL support built on top of BPEL support introduced in WebSphere Business Integration Server Foundation V5.1 and WebSphere Studio Application Developer Integration Developer V5.1
- Extensions to WS-BPEL provided and supported
  - Specific for using Java™ and working within an enterprise environment
  - Enabled only through WebSphere Integration Developer and supported only by WebSphere Process Server

(New and enhancements support from previous versions noted by specific icons) New V6 Enhanced in V6

Business processes defined in BPEL require robust runtimes that can interpret the business process and execute the appropriate steps while considering things such as long running business processes that span hours, days, or even longer and integration with transactional environments. WebSphere Process Server V6.0 includes the Business Flow Manager which is responsible for all aspects of BPEL business process execution. Support for designing BPEL business processes is provided with the Business Process editor in WebSphere Integration Developer.   Both WebSphere Process Server V6.0 and WebSphere Integration Developer V6.0 build on the support provided by WebSphere Business Integration Server Foundation V5.1.1 and WebSphere Studio Application Developer Integration Developer V5.1 for BPEL 1.1 business processes and now support BPEL.   With WebSphere Process Server and WebSphere Integration Developer, there are additional extensions and enhancements provided which allow for tighter integration with the underlying Java implementation and support for executing long-running business processes in an enterprise environment where transaction capabilities are an important consideration. Throughout this presentations, the areas of new support or enhanced support provided by WebSphere Process Server V6.0 and WebSphere Integration Developer V6.0 will be identified with special icons.



Run-Time Aspects of BPEL

- BPEL is defined to be independent of any software vendor's run-time implementation
- IBM defines specific attributes for executing business processes with Business Process Choreography

Process Type	Uninterruptible	Interruptible
Transactions	One transaction for the entire process	Transaction boundaries can be set between each activity
Persistence	None	Processes can be quiesced to a database and restarted
Crash Recovery	None, execution is completely transient. If the server crashes, process state is lost.	Process can resume from the last checkpoint
Parallelism	None, strictly single-threaded	Flow activities may execute in parallel
Interruptible	NO	YES
Asynchronous	NO	YES
Correlation	NO	YES

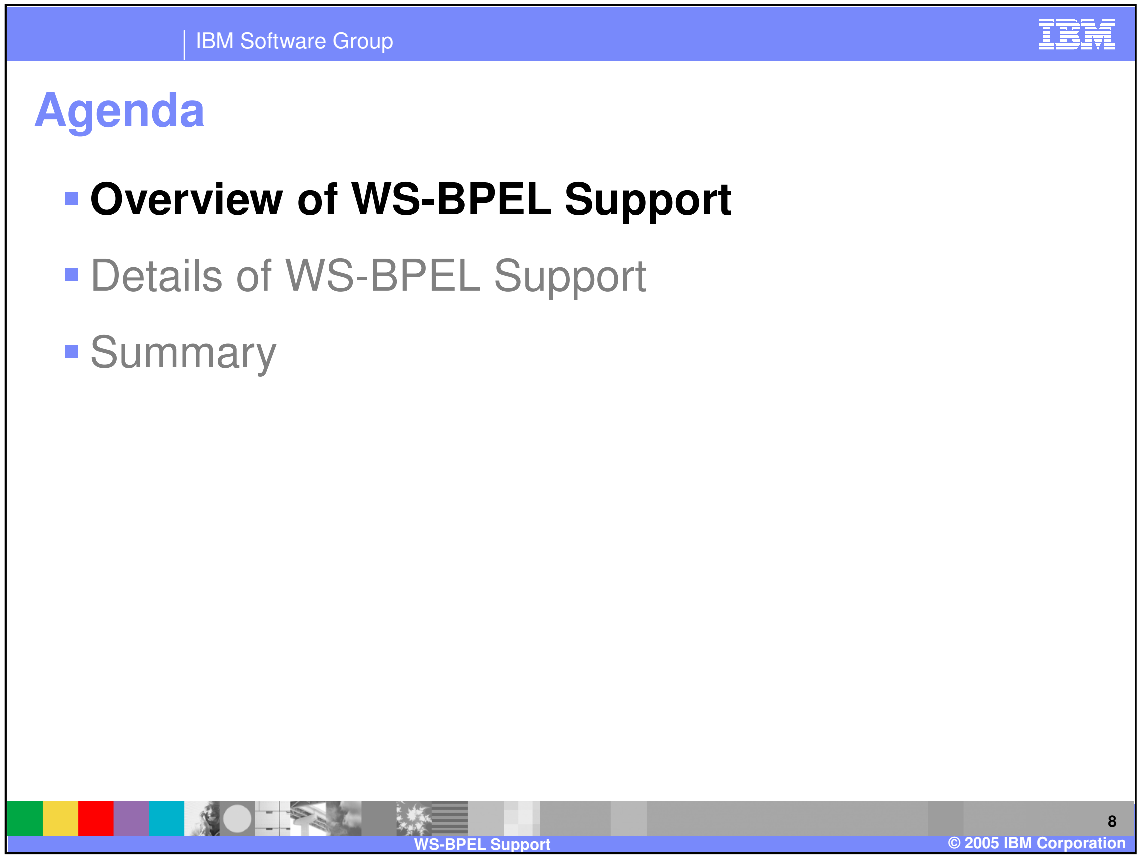
It is important to understand some of the runtime specific areas of support and BPEL extensions before going any further. Even though the goal of BPEL is to separate business logic from the underlying implementation, there are some implementation details that you need to be aware of in order to construct a business process. Certain extensions will restrict the capabilities and BPEL constructs available during design time. Many of these details are related to the IBM implementation of BPEL and are not outlined in the BPEL specification.

These runtime specific details are divided into the two primary areas of Uninterruptible, short-running business processes and Interruptible, long-running business processes.

The Business Process Choreography runtime supports executing business processes in different ways and provides different capabilities depending on the type of process. For example, a transaction is considered one uninterruptible process that is used to execute the entire business process. If the transaction consists of 5 steps, they will all run under a single J2EE transaction. A transaction can be either a new transaction initiated on the server or it can be a transaction passed in by a client. If it is passed in by a client, it is the responsibility of the client to commit the transaction when the business process is returned. If the business process engine starts a new transaction, the transaction will be committed when the business process completes.

With an interruptible, long-running business process, business process execution consists of multiple transactions. For example, a loan approval process could potentially take a week or longer to complete and the standard server transaction cannot be held open that long. For this reason, there will be multiple transactions at each step of the business process. For instance, one step of the loan approval process might be obtaining the credit rating for the applicant, which would be one separate transaction. There would be another transaction for the approval step. This separation of process steps provides crash recovery because if a specific part of the process fails, it does not cause the entire process to fail. The process can be resumed and completed at a later time. Additional processing can also be done to recover from the failure. Another feature of an interruptible process is persistent state that allows for resumption of the process in the event of failure or for exits to break out of the process for a task requiring human intervention such as a loan approval by a loan officer. Once the approval is given, the process can resume at the next step in the process. Multiple threads can be spawned, allowing flow activities to execute in parallel. Asynchronous processing provides better stability, crash recovery and persistence. A variety of messaging queues are used internally to distribute workload and establish transaction boundaries to indicate units of work. Correlation is provided, ensuring outside requests coming in to a business process are routed to the correct instance.

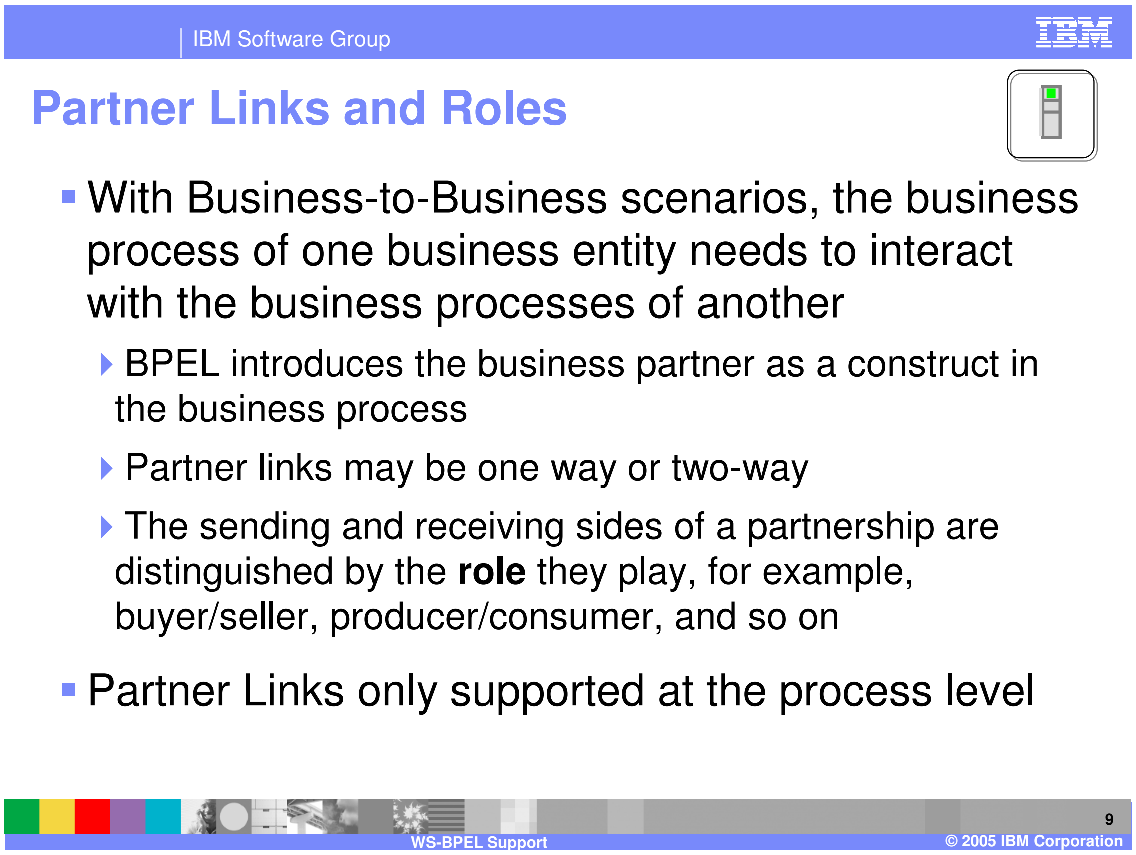
With an uninterruptible, short-running business process, a failed transaction will result in all the completed activities of the process being rolled back to the beginning of the process or back to the client that passed the transaction. Because all information is stored in memory, there is no persistence and all information would be lost in the event of a failure. Because all processing is done on a single thread, there is no capability for parallelism. Even if the process was modeled in a parallel manner, the business process engine will alternate between the different paths, simulating parallelism. However, this is not a true multi-threaded parallel type of runtime or execution. All processing is synchronous and no correlation is provided.



Agenda

- **Overview of WS-BPEL Support**
- Details of WS-BPEL Support
- Summary

This section will discuss the WS-BPEL specification in more detail and look at the extensions supported by WebSphere Process Server and WebSphere Integration Developer.

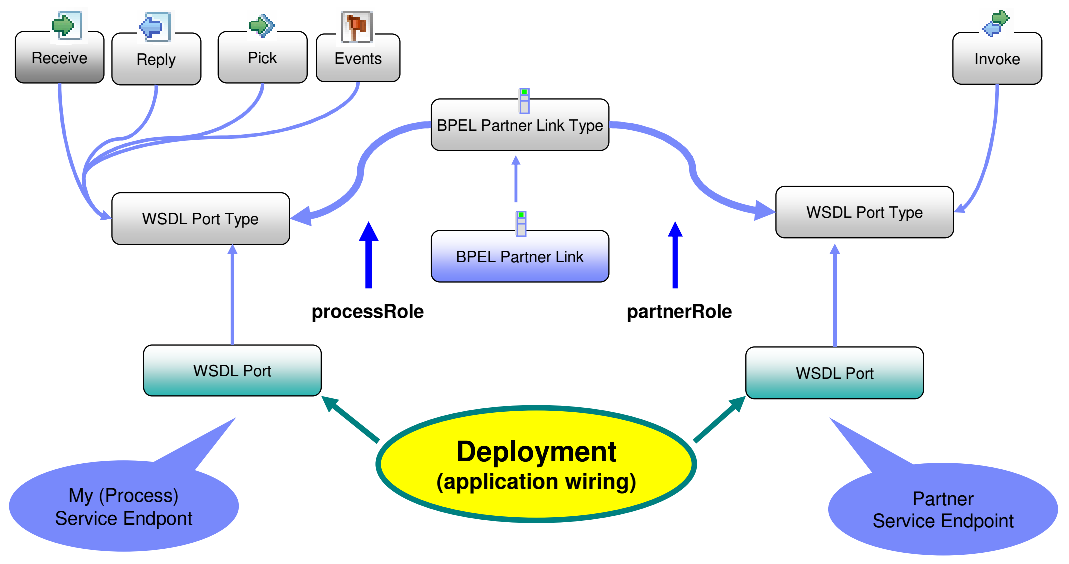


Partner Links and Roles

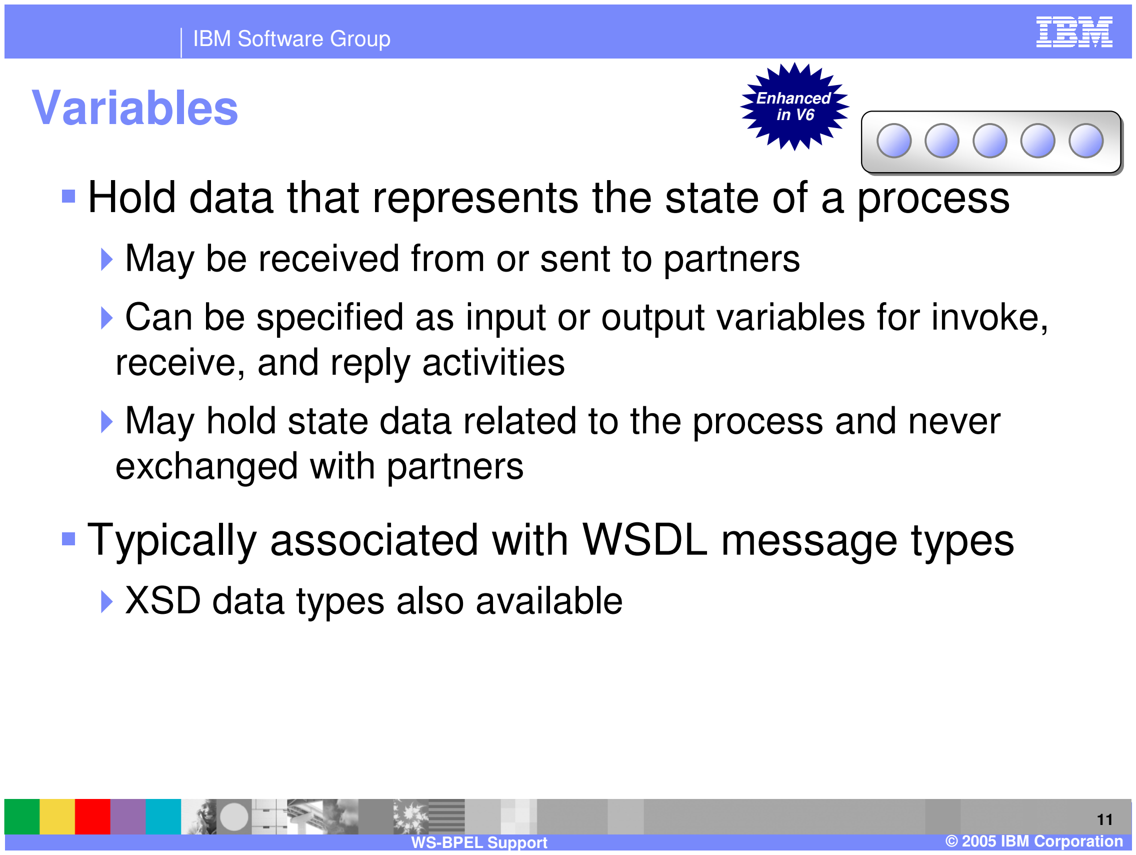
- With Business-to-Business scenarios, the business process of one business entity needs to interact with the business processes of another
  - BPEL introduces the business partner as a construct in the business process
  - Partner links may be one way or two-way
  - The sending and receiving sides of 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

It is easiest to discuss the various parts of the BPEL specification by starting with the Partner Links and Roles. 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 is 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



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) used by clients to send in data and receive a response. Pick and events are also used for sending in data and are also 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) used to call the business process is also defined at deployment. Implementation details are not needed or used within the definition of the BPEL process.

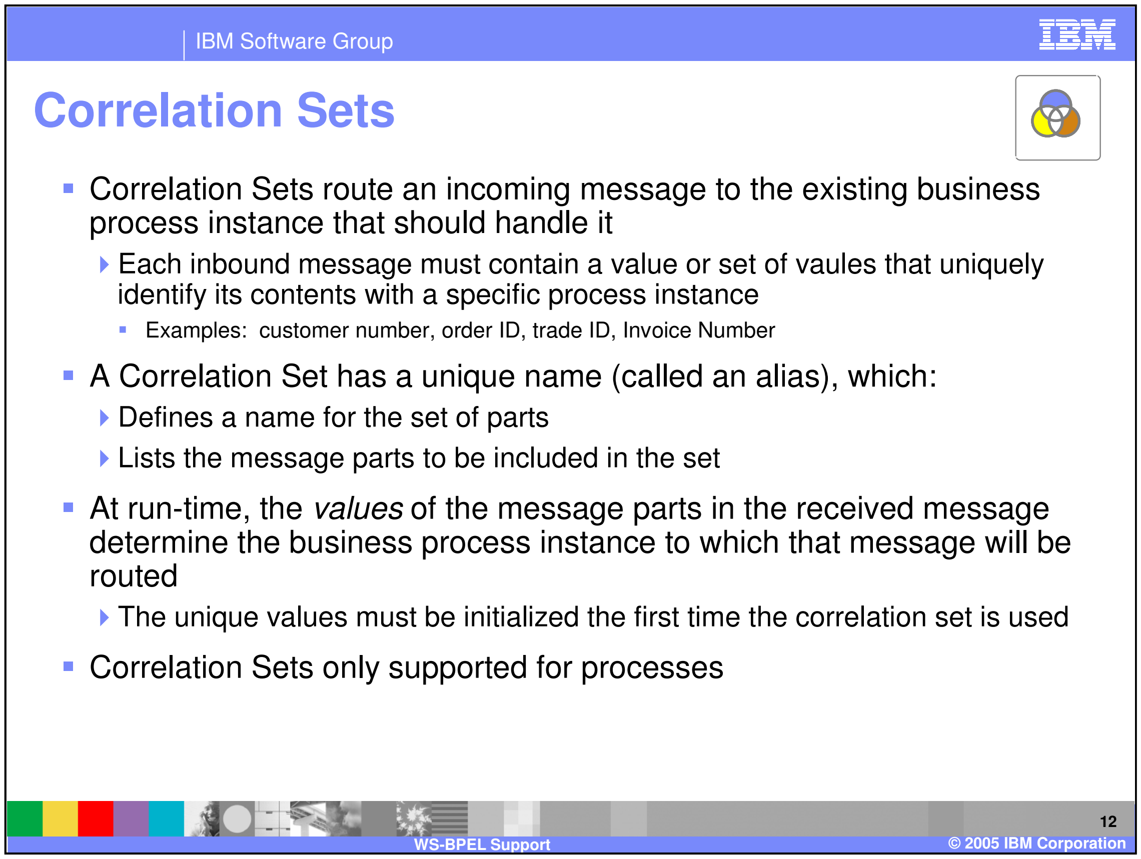


Variables

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

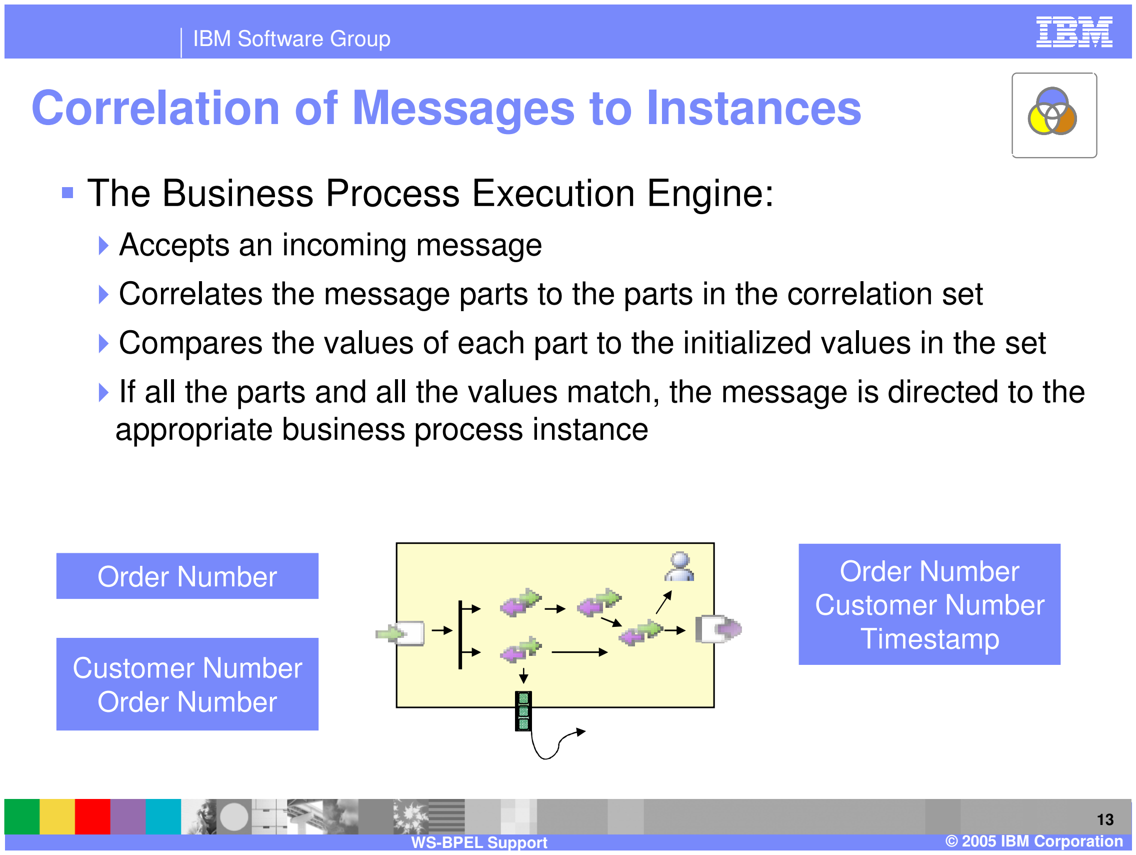
Variables are another very important part of a business process definition as they represent the state of the business process. Variables store information about data coming in to the business process that is then reused and sent to the various steps that are executed as part of the business process. Variables are also used to hold the data that is returned from a service back into the business process or to hold internal information such as counter values for iterative processes. The variable capability has been enhanced in line with the BPEL specification to allow the use of XSD data types. 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 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

Correlations consist of assisting with outside requests coming into the correct business process instance. For example, if there are 1000 loan applications in flight and the credit department is sending in credit report messages for multiple applications, correlation ensures that the message is sent to the correct instance. This is accomplished through the use of correlation sets. Within a correlation set, you will define the parts of an incoming message that indicate uniqueness. These will then map down to some type of value within the business process state. The messages can then be matched up with instance containing the same unique parts. It is important to note that correlation sets are provided from application data. There are some unique internal values used to keep track of multiple instances, but the business process must be designed using some unique value that is part of the enterprise. An example would be something like a social security number, order number, customer number or some 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 as the long-running business process executes over a period of time. As the business process state changes, there might be different unique identifiers that must be used to identify a unique 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 only supports correlation sets at the process level while the BPEL specification supports correlations sets at other levels.



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

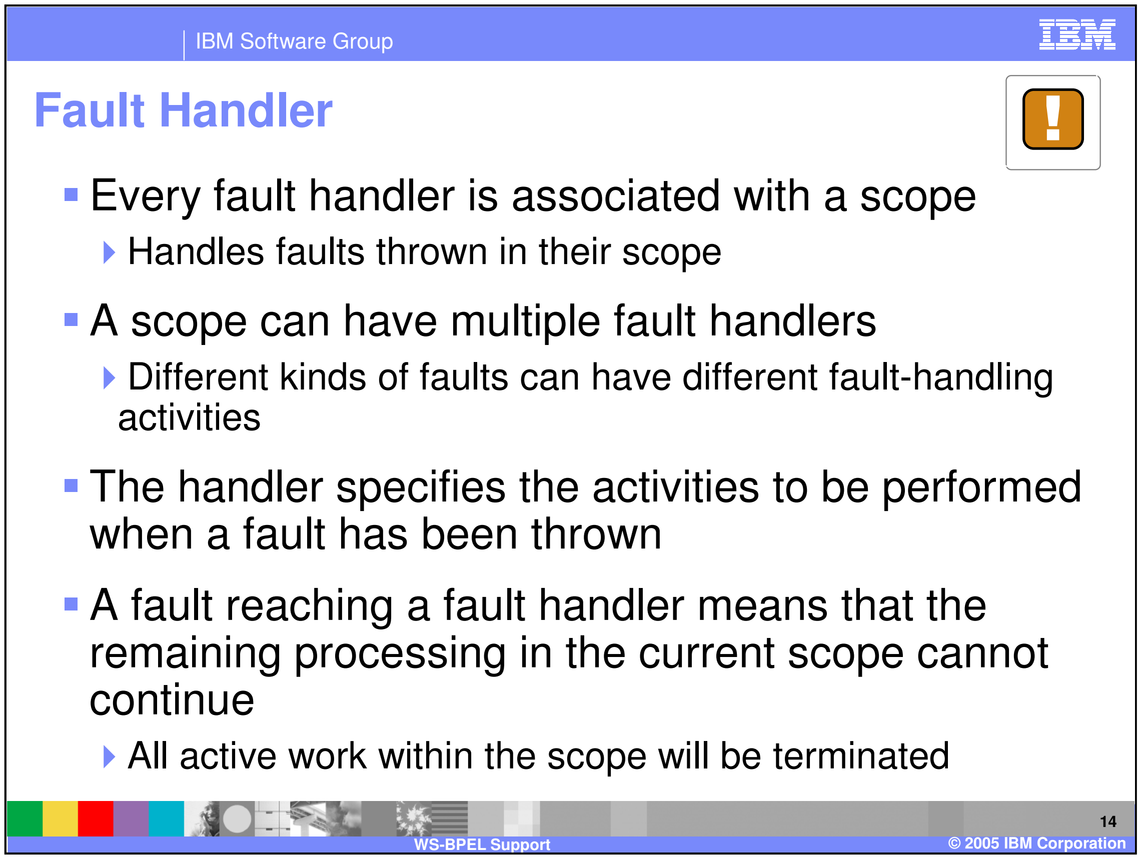
In a long running process there are points in the process where its state data are captured and the process goes into 'waiting' state, waiting for some sort of response, such as the manager to approve the expense. To make sure that the approval is associated with the right expense, some unique identifier 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, thus with BPEL there is the concept of correlation sets. … more on this later….

An important point to understand about the correlation ID is that it is something that is known by the outside activity with out 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 and returned to the starter 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.


IBM Software Group

Fault Handler

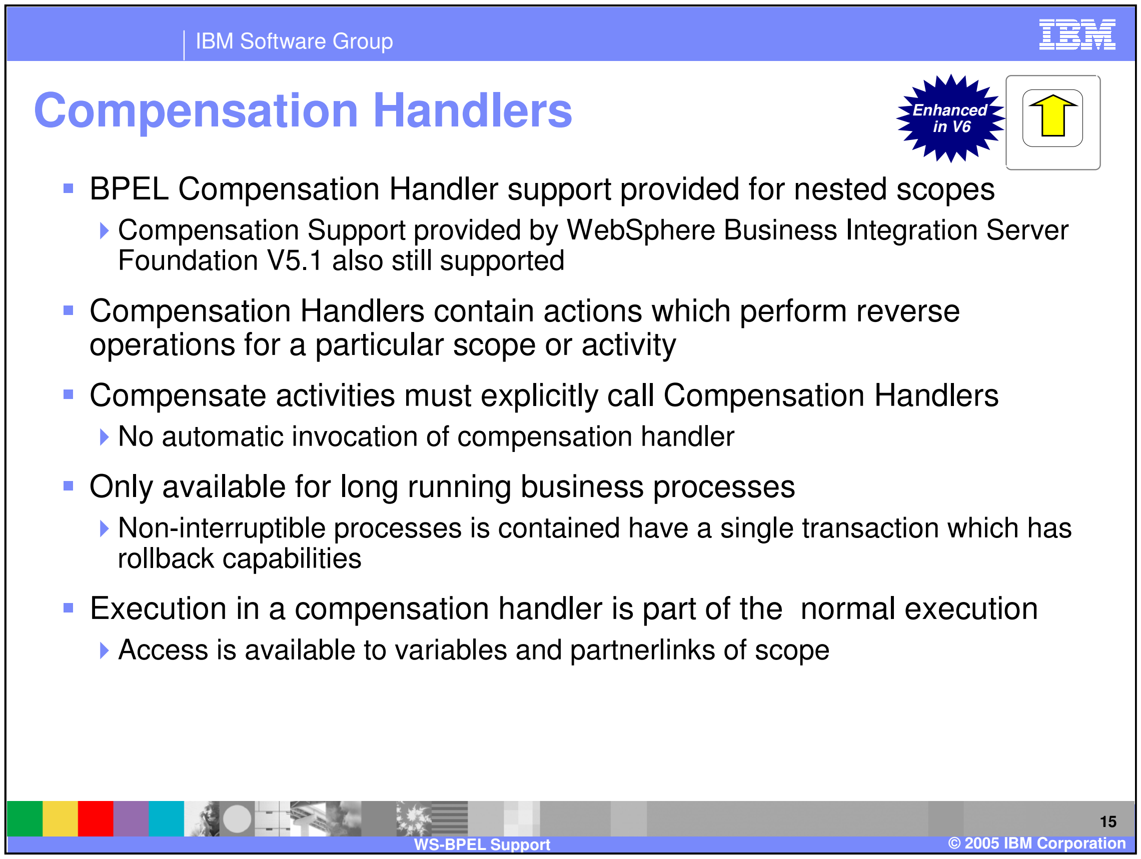
- Every fault handler is associated with a scope
  - Handles faults thrown in their scope
- A scope can have multiple fault handlers
  - Different kinds of faults can have different fault-handling activities
- The handler specifies the activities to be performed when a fault has been thrown
- A fault reaching a fault handler means that the remaining processing in the current scope cannot continue
  - All active work within the scope will be terminated

14

WS-BPEL Support

© 2005 IBM Corporation

Fault handlers are designed to detect and signal a failure in the execution of a business process. They are designed to catch primarily business exceptions such as inventory falling below a lower limit that is set in the business process 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 scope that encompasses a set of activities. When a fault is caught by a handler, any execution currently in progress on the activity would be terminated and execution would switch to the fault handler. Fault handlers can be nested in a similar manner to the way in which 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 would be expected to handle the fault or for some other fault monitoring tool to handle the fault. Fault Handler can also catch runtime failures. However, as a best practice it is better not to catch specific runtime problems as that exposes implementation details in the business process.

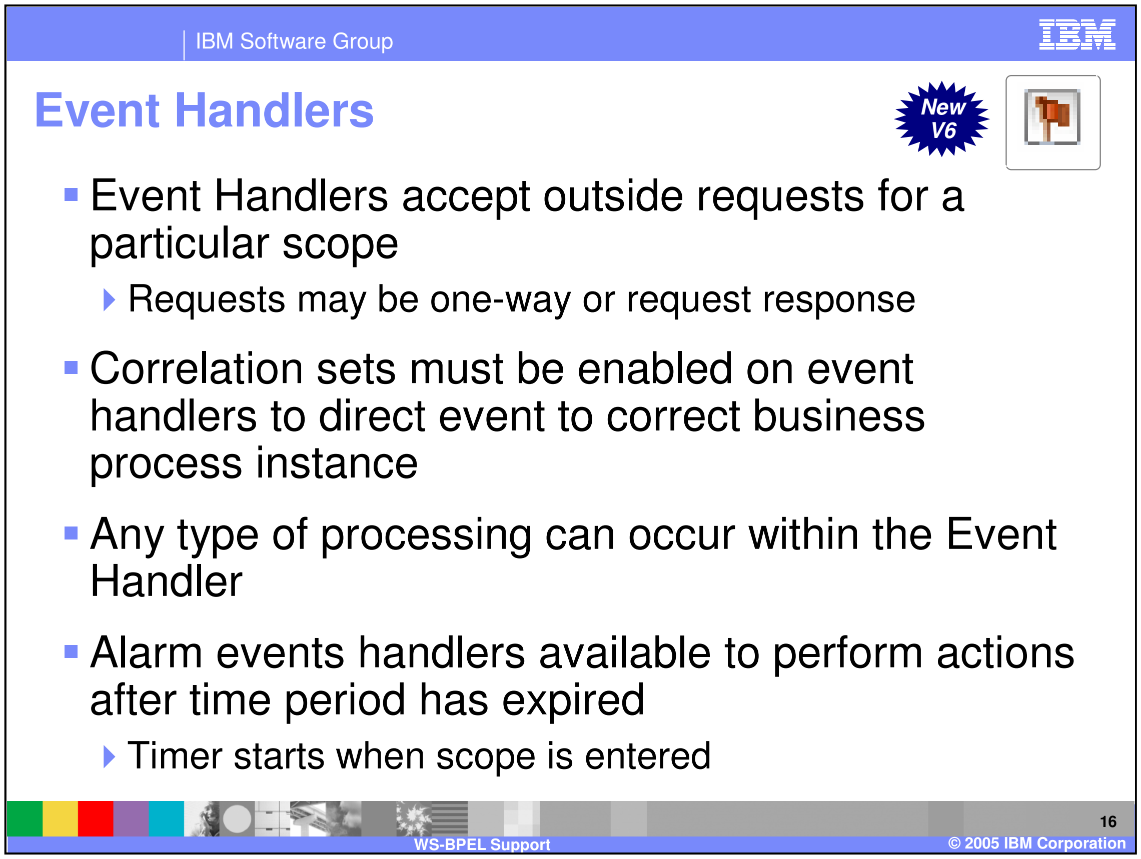


Compensation Handlers

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 is contained have a single transaction which has rollback capabilities
- Execution in a compensation handler is part of the normal execution
  - 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 it is recommended to used 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 consist of 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.



IBM Software Group

Event Handlers

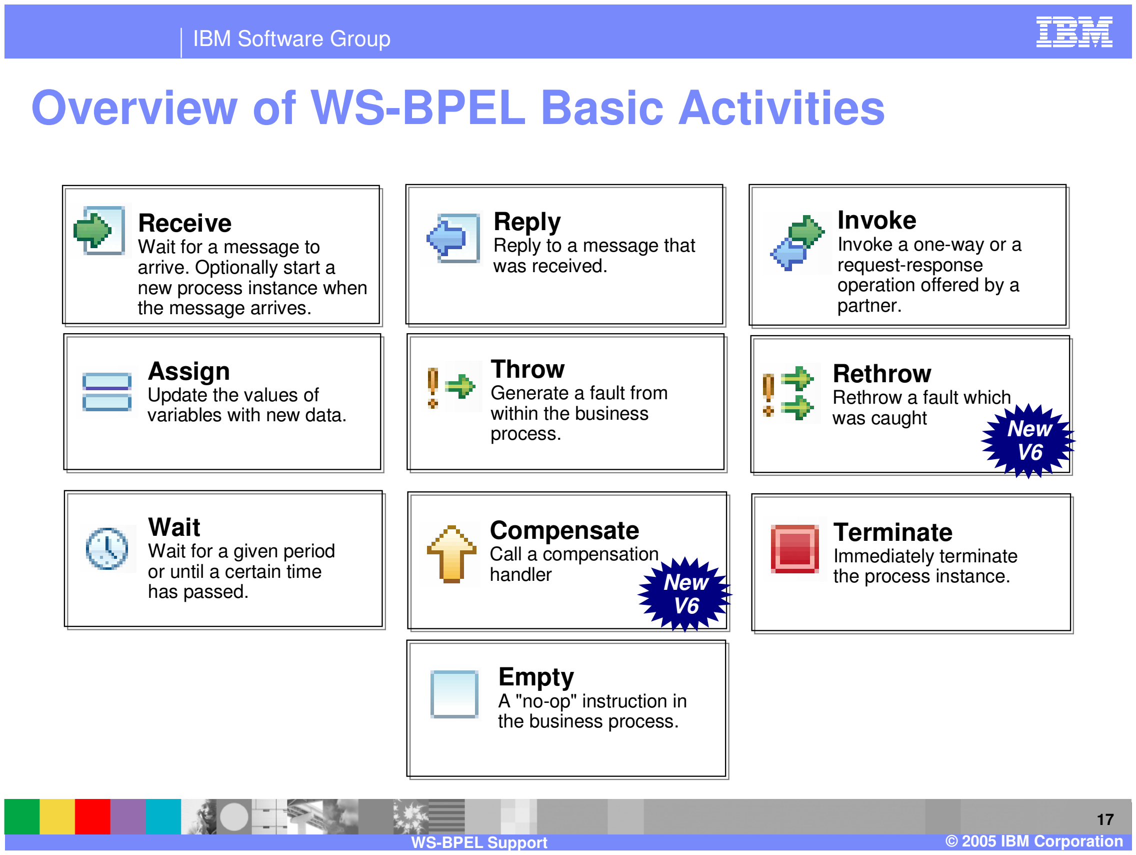
New V6

- 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

16

WS-BPEL Support © 2005 IBM Corporation

There is also new support for event handlers in WebSphere Process Server V6.0. Event handlers were originally defined under the BPEL 1.1 specification. Event handlers are only available for a long-running business process. They accept outside 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 limited to a specific scope. For example, during a customer order process there might be phases of the process, such as checking inventory or placing a product on the order, where it is not allowed to make changes. However, during other phases it might be allowed to modify the order in order to remove a product or even cancel the order. 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 executed on a separate thread from the business process itself. When the event handler activities are complete, the thread goes away. The event handler does have access to business process state information such as variables and partner links. 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.



Overview of WS-BPEL Basic Activities

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.

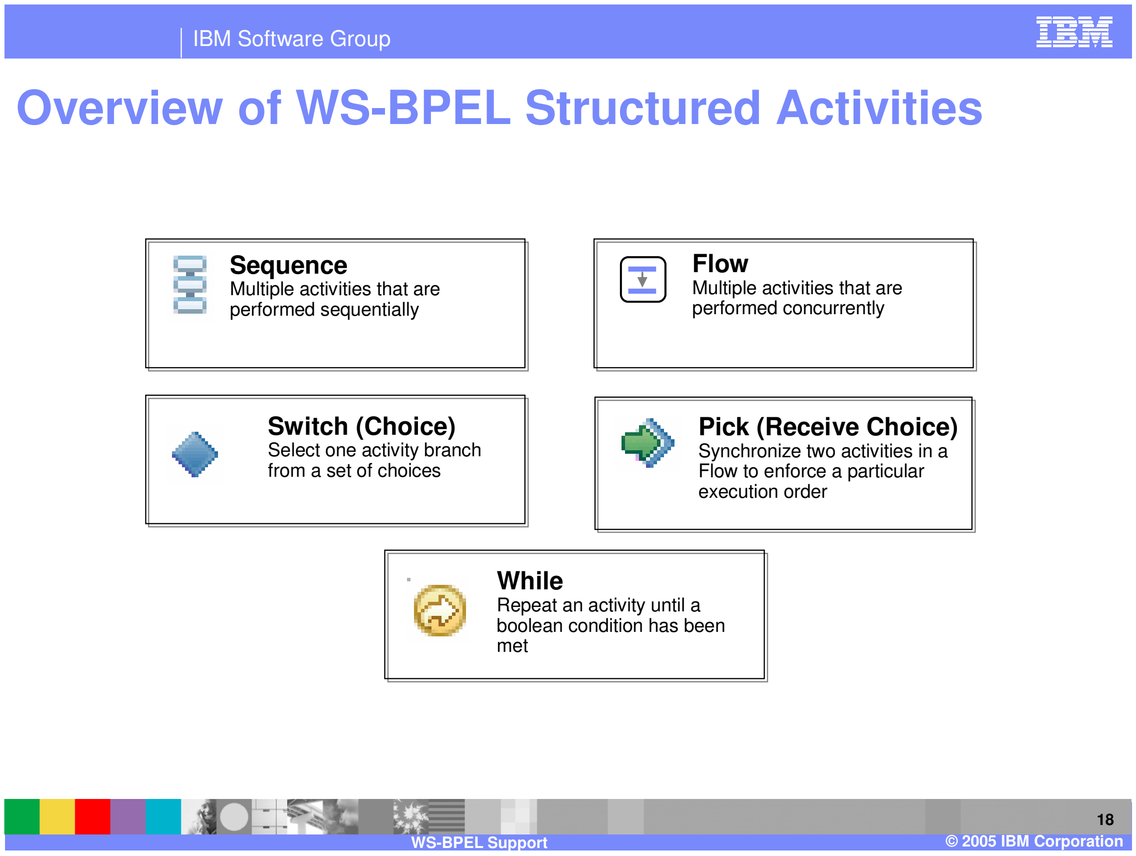
Compensate
Call a compensation handler
New V6

Terminate
Immediately terminate the process instance.

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

Here is an overview of the basic BPEL activities which are used primarily to perform and carry out the different steps in the business process. The Receive activity is an asynchronous activity which can be added to a business process. When a Receive activity is reached, the business process will halt and wait for a specific operation to be called from an outside location, in theory, synchronizing the business process with an outside action. When the operation is called, the business process will continue. The Receive activity is a blocking wait. 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 provides. Each Invoke activity will specify a specific Partner Link and operation on that partner. Depending on the type of interface of the partner, the Invoke activity will specify variables for the request and response message. Assign activities are the primarily way data can be transformed and moved from one variable to another. Business processes can fail at a business process level, which is something that is acceptable and can be part of the business process model. When a Fault occurs in a business process, the Fault Handler is invoked and the fault can be managed at the current scope or thrown to higher level scopes, signaling the business process has failed, or the business process can terminate. Throw activities are for indicating some type of failure has occurred within the business process and it needs to be handled either in the business process or passed back to the client as now further processing can occur. If a failure is caught a Rethrow activity can be used to throw the same failure which was caught 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. During this time, navigation will not occur on the path of the Wait activity but can continue on other parallel paths. The Compensate activity is used to call a Compensation Handler which contains business processing to undo or reverse processing for completed steps in a business process. 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 which may be implemented. When the Empty activity is reached, the business continues without stopping, treating the activity as a no op. The Empty activity can be morphed at a later time to any of the other activity types.

The Rethrow and Compensate activities are newly supported in WebSphere Process Server V6.0. These will be discussed in more detail later.



Overview of WS-BPEL Structured Activities

Sequence
Multiple activities that are performed sequentially

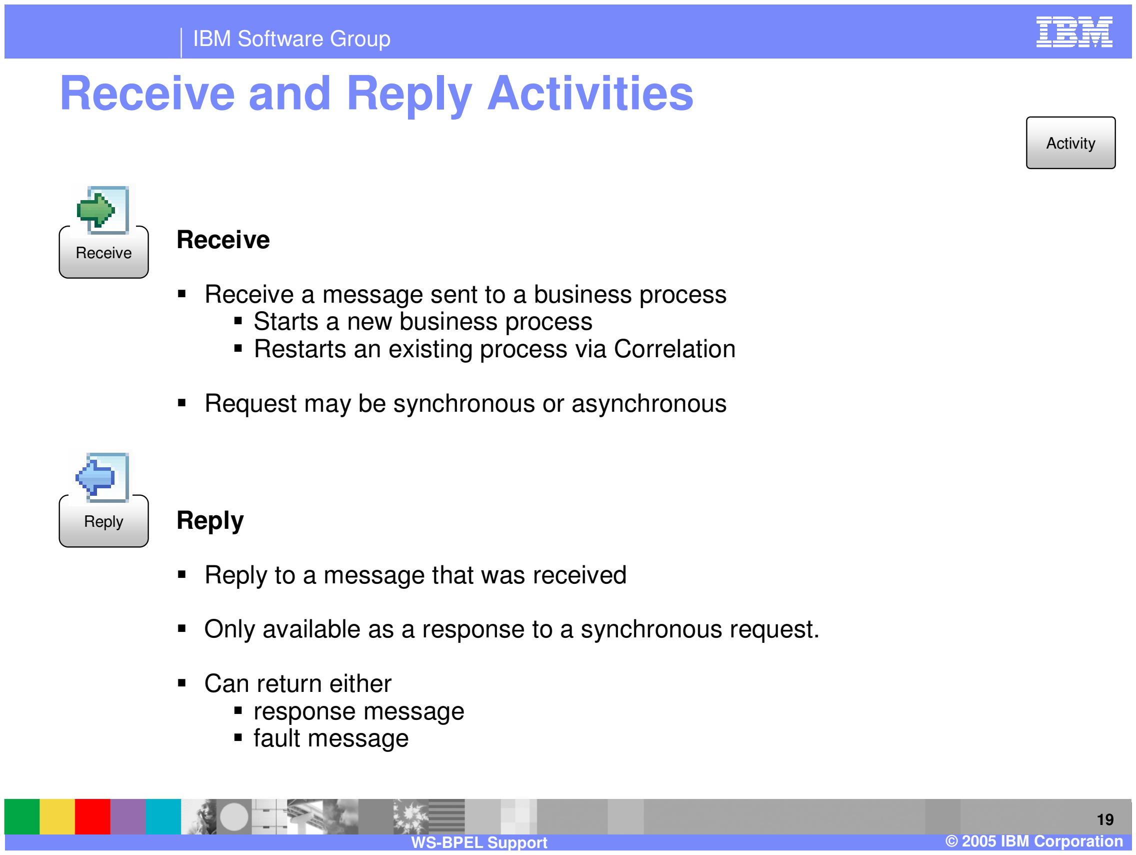
Flow
Multiple activities that are performed concurrently

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

Pick (Receive Choice)
Synchronize two activities in a Flow to enforce a particular execution 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 programing within a business process such as sequencing activities or placing them in a parallel type of execution 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 as 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 upon a boolean condition which is evaluated prior to execution and at each iteration.



Receive and Reply Activities

Activity

Receive

Receive

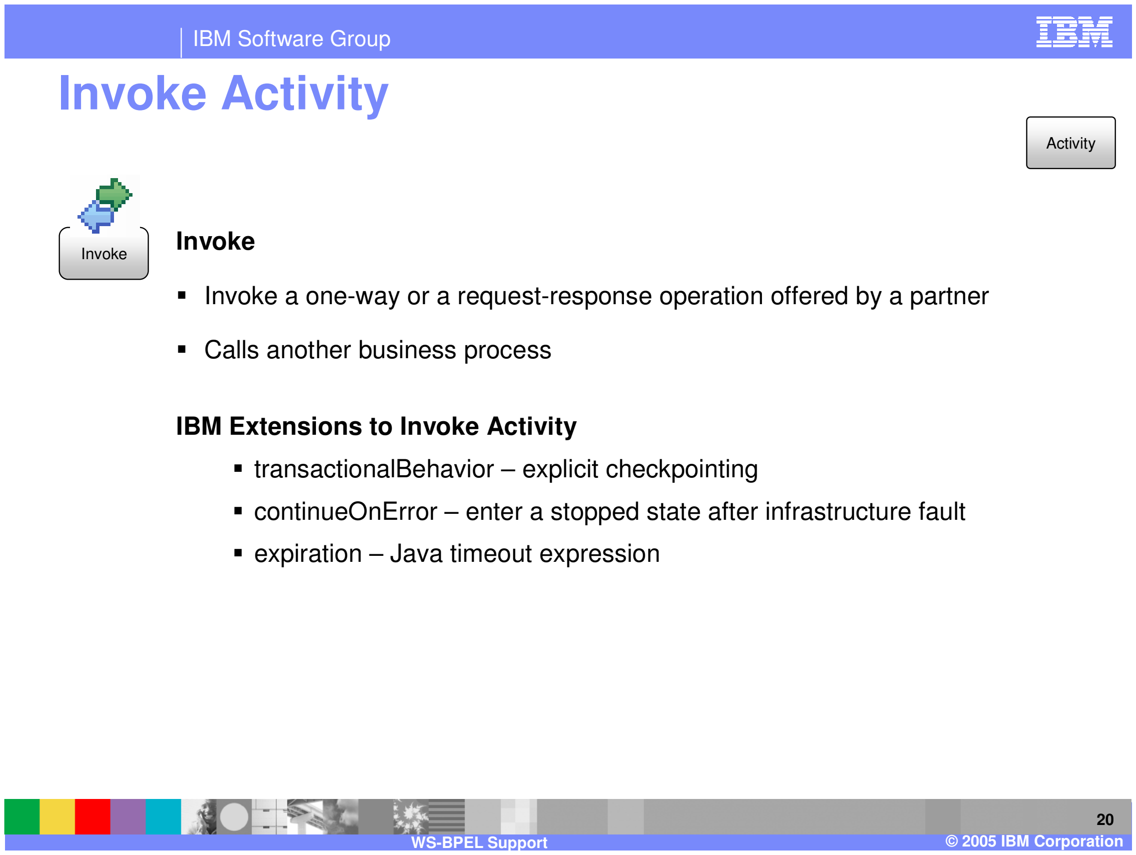
- Receive a message sent to a business process
  - Starts a new business process
  - Restarts an existing process via Correlation
- Request may be synchronous or asynchronous

Reply

Reply

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

The support for Receive and Reply activities has not change from WebSphere Business Integration Server Foundation V5.1. Receive is one of the primary 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 back a response message indicating success or any of a variety of fault messages in the event of a fault occuring. 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.


Activity
Invoke

Invoke Activity

Invoke

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

IBM Extensions to Invoke Activity

- transactionalBehavior – explicit checkpointing
- continueOnError – enter a stopped state after infrastructure fault
- expiration – Java timeout expression

The invoke activity is the primary activity for performing the various steps of a business process. This activity calls out to the different services by means of the partner links. Services can be one way 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 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 around the invoke such as the transaction behaviour of a long-running process. This includes details such as whether or not 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, better performance can be realized by grouping a number of invokes or other types of snippets with invokes together 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.

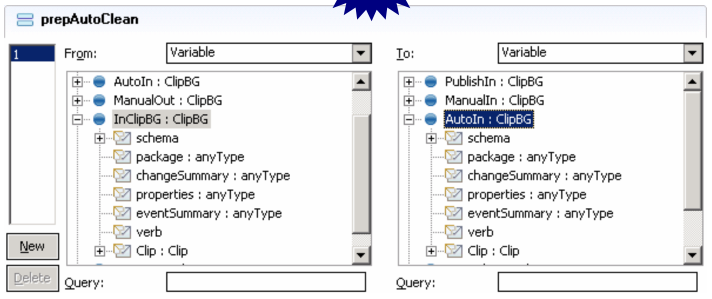
Assign Activity

Activity

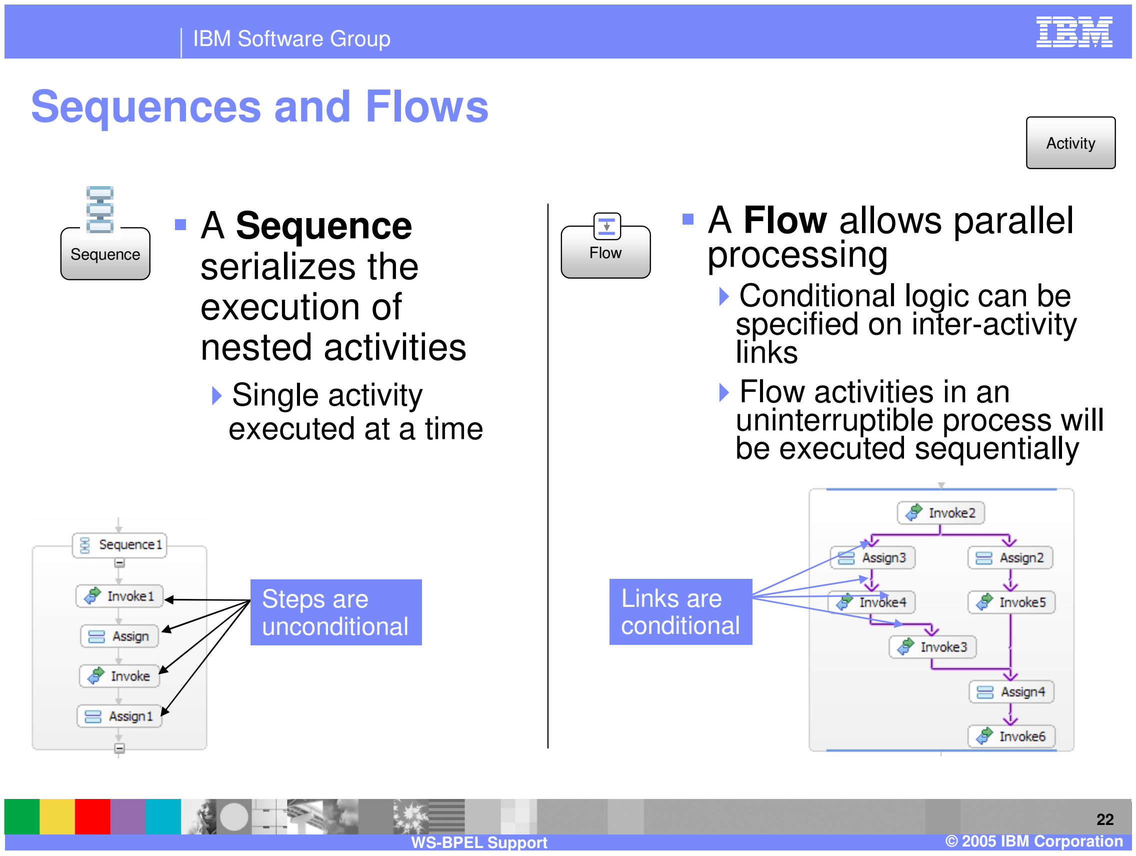
Assign

- 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, or when using custom properties for message parts
- Full XPath 1.0 support

Enhanced in V6



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.



Sequences and Flows

Activity
Sequence
Flow

- A **Sequence** serializes the execution of nested activities
  - Single activity executed at a time

- A **Flow** allows parallel processing
  - Conditional logic can be specified on inter-activity links
  - Flow activities in an uninterruptible process will be executed sequentially

Sequence1
Invoke1
Assign
Invoke
Assign1
Steps are unconditional
Links are conditional
Invoke2
Assign3
Assign2
Invoke4
Invoke5
Invoke3
Assign4
Invoke6

Shown here are the sequence and flow activities. A sequence consists of a serialized execution of activities occurring one right after another. Execution must complete successfully before continuing on to the next one. A flow allows a business process to have parallel execution of activities. Within the parallel path, there are links specified. These links can include logic to check whether or not execution should continue down a certain path. This logic can use variable state information from the business process to determine if execution 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 execution would be random sequential.

Switch (Choice) Activity

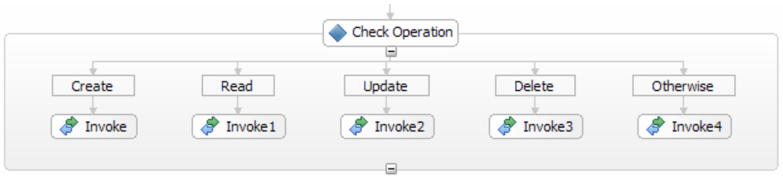
Activity

Switch

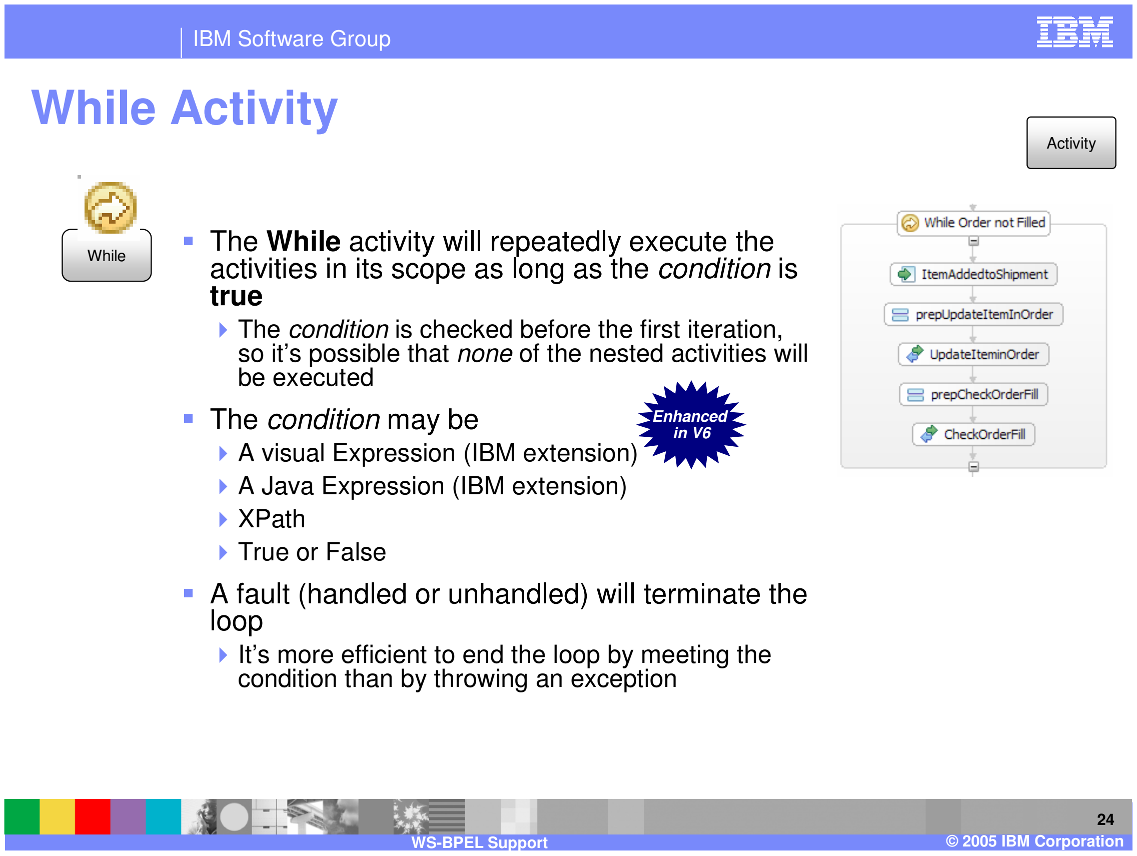
- A **Switch** implements a decision chosen from multiple conditions
  - Condition can be a Visual Expression, Java Expression (IBM Extension), XPath, True or False

Enhanced in V6

- The first case condition to evaluate to true is executed
- If no case resolves to true the Otherwise case is executed
- If the Otherwise is omitted, an implied Otherwise with condition true is used



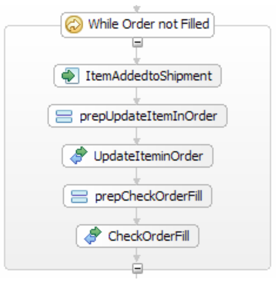
A switch or choice activity is similar to a programming case statement where conditions are defined. Whichever conditions resolves to true first determines what processing takes place. Conditions are checked from left to right and when one evaluates to true, the activities defined under that condition will be executed. 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.



While Activity

- The **While** activity will repeatedly execute 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 executed
- 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



The while activity is used to execute activities iteratively. The activities under the scope will be executed as long as a defined condition remains true. The condition is evaluated prior to activity execution, so it is possible that none of the while activities will be executed. The while activity has been enhanced to include support for evaluating data using XPath in V6. 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 occurr and reach the while scope, any execution in that loop that is executing in a parallel manner will halt and execution will continue in the fault handler. 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 as there is additional overhead associated with throwing a fault. The use of faults should be reserved for indicating failures in the business process.

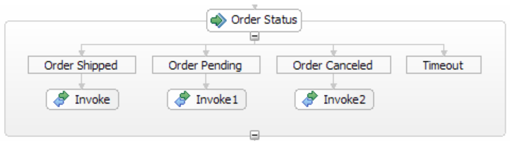
Pick (Receive Choice) Activity

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

Enhanced in V6



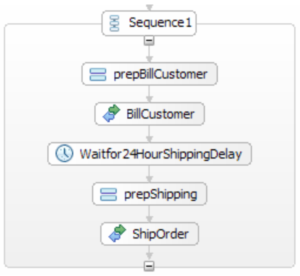
The pick or receive choice activity provides a point within the business process where information can be received from outside entities or from different clients. This points or interfaces can be different in different circumstances. For example, if you are waiting for a message to indicate order status, such as shipped, pending, or cancelled, each of these 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, execution 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 includes 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.

Wait Activity

- 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



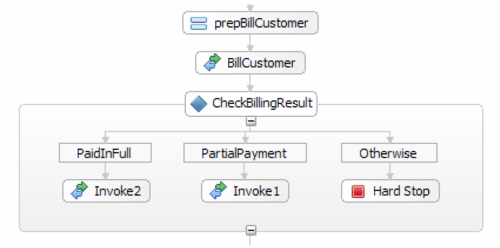
The wait activity can be used to stop a business process on the current execution path for a specific period of time or until a specific date has been reached. The duration or date can be either a hard coded value or calculated dynamically. XPath support which is new in WebSphere Process Server V6, can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

Terminate Activity

Activity

Terminate

- A **Terminate** activity ends processing of business process instance immediately (hard stop)
- All execution ends, including parallel execution paths
- Recovery of terminated processes (compensation) is unavailable



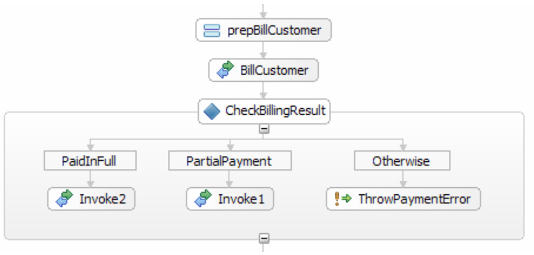
The terminate activity is the equivalent of pulling the plug on an electrical device. This is a hard stop that signifies a failure has occurred or the business process should be halted at this point immediately. If a process is terminated, no recovery is possible (or wanted) and no compensation is possible.

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 is sent to an outside entity or a higher level in the business process that there is a problem. 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 up to the next 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.

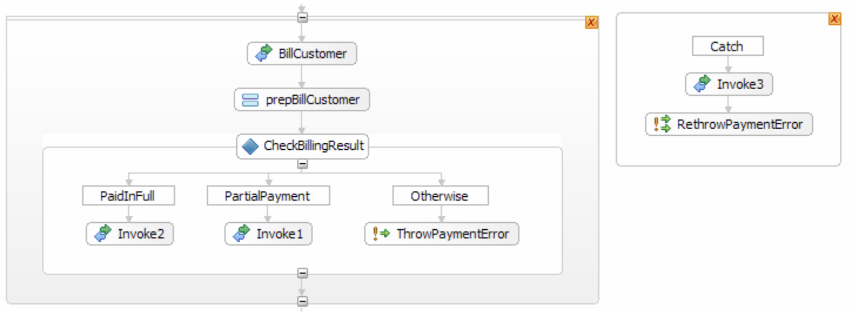
Rethrow

New V6

Activity

Rethrow

- Signals an error state still occurs
- Only available in Fault Handlers
- Raises new fault with identical name and message as caught fault



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 up to the next scope. In the BillCustomer example, if the CheckBIllingResult returns PaidInFull or PartialPayment, execution continues. However, if for instance 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 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.

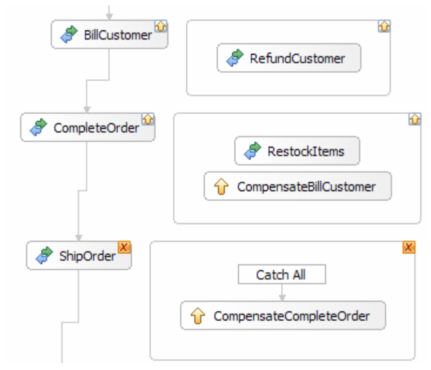
Compensate

New V6

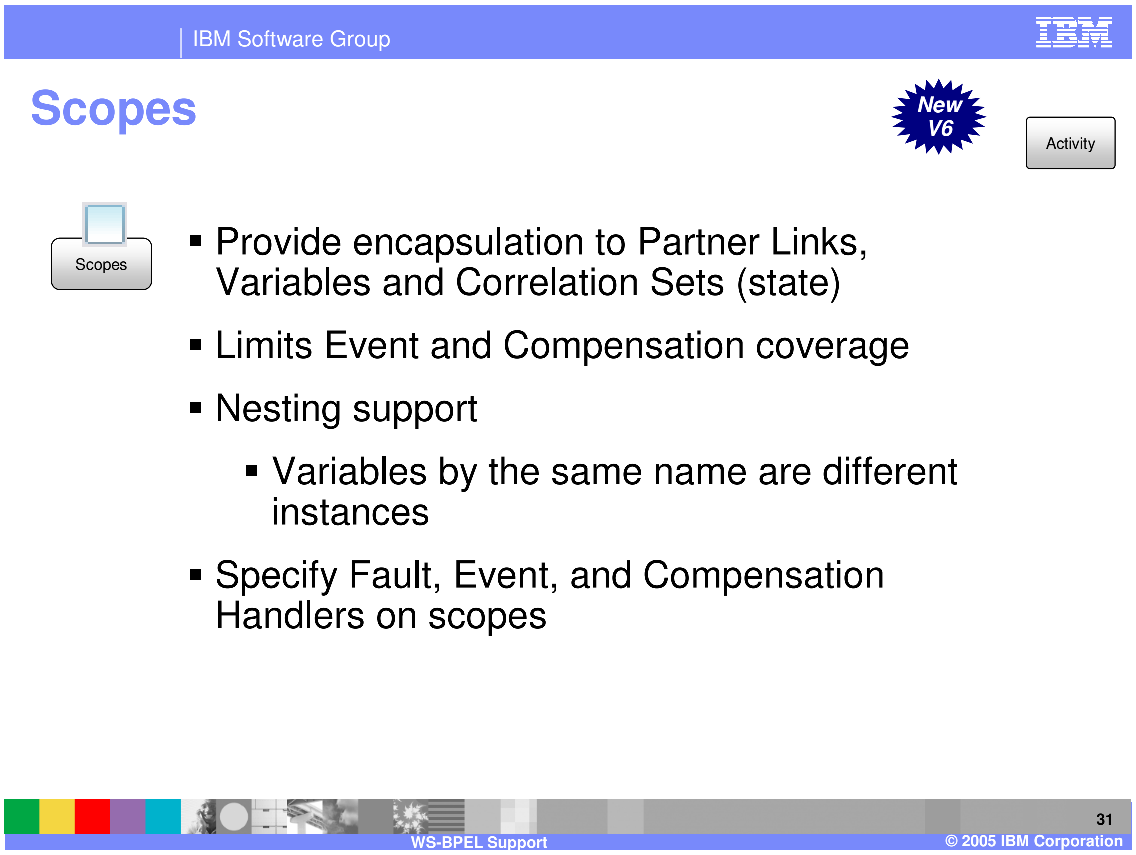
Activity

Compensate

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



The compensate activity is a newly supported in WebSphere Process Server V6.0 and is defined in the BPEL specification. This activity signifies that an activity or scope of activities that has completed must be undone. This occurs when a failure situation is reached and 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, executing 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 in a reverse situation and return it 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.



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

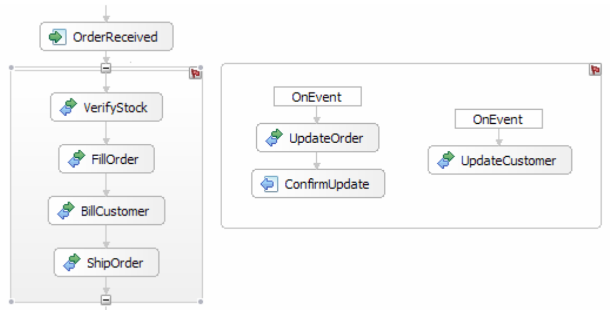
Scopes are supported in WebSphere Process Server V6.0 as defined in the BPEL specification. Scopes encapsulate correlation sets and the state of a business process in order to make particular values available to a specific set of activities. Scopes can be established to define event handlers as well 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 on the business process editor. The global variables are still available, even though only the current scope variables are shown in the editor.

Event Handlers

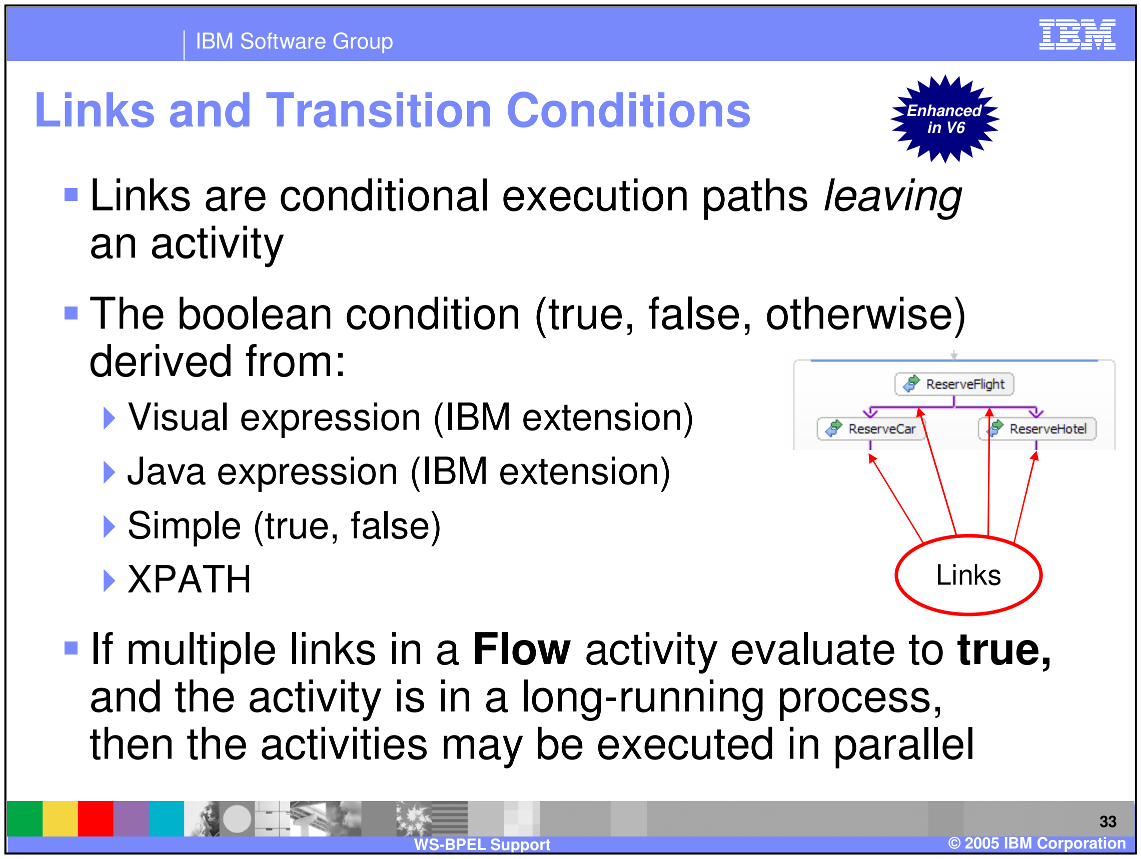
New V6

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



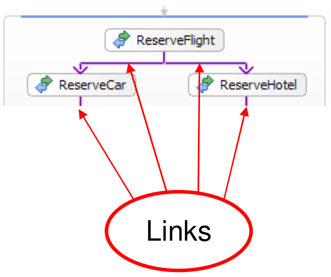
Events are associated with a scope. When the scope is started, events can be received until the scope is completed. This example lists four activities that are grouped into a scope with an event handler defined on the scope. As these four activities are executed at any time, different events can be received. There are two events that could come in while these four are executed. Each one of those events would initiate a different set of potential activities. The event can be one-way operation such as that on the right or a request/response as 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 thread of execution with access to partner links and variables within the scope is used for each event. Events can be secured differently, restricting 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.



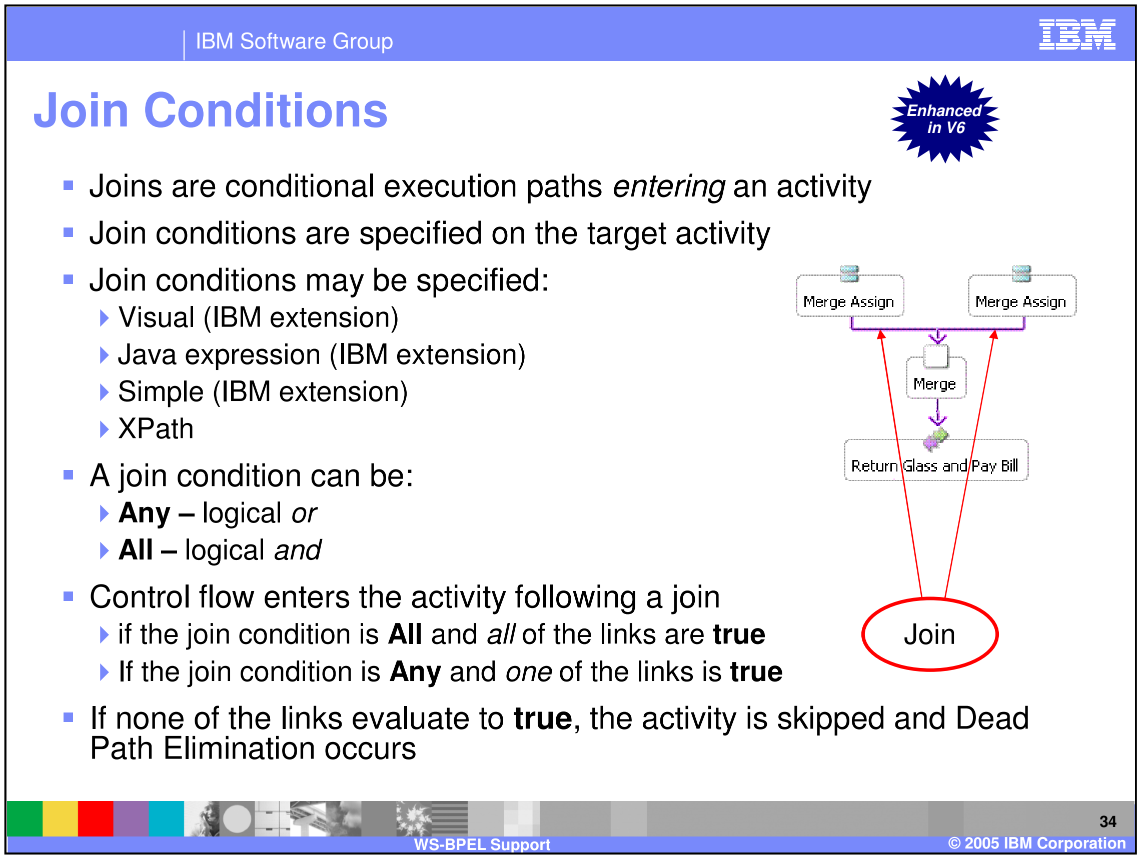
Links and Transition Conditions

Enhanced in V6

- Links are conditional execution 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 executed in parallel



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. Conditions can also be evaluated using Java programming, an IBM extension, or a value of true or false can be hard coded into the link. Parallel execution can be performed during a long-running business process if multiple conditions evaluate to true.

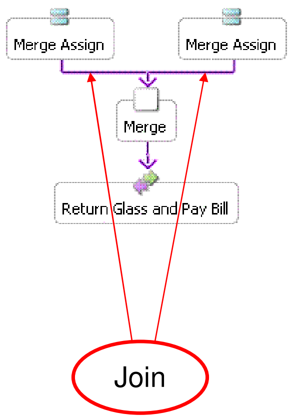


IBM Software Group

Join Conditions

Enhanced in V6

- Joins are conditional execution 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

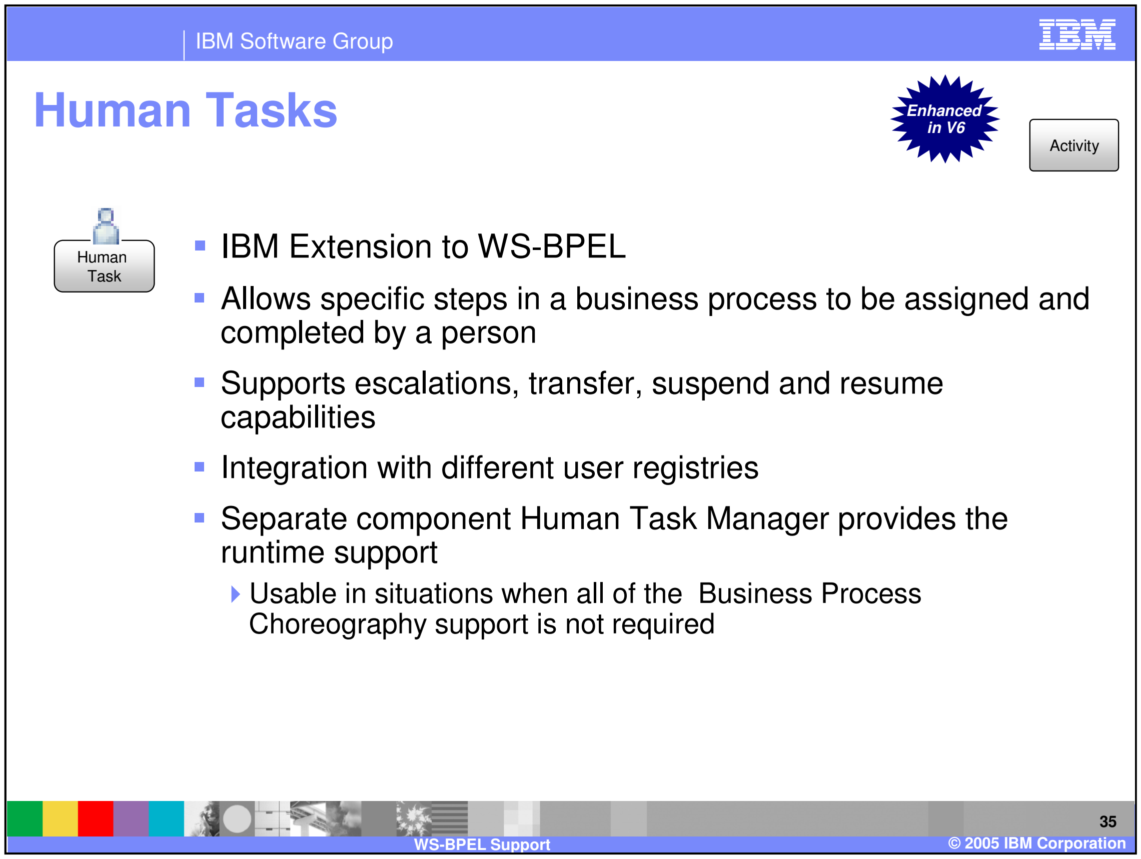


WS-BPEL Support

© 2005 IBM Corporation

34

Join conditions are logic to indicate if the process should wait for execution from all paths or just a single path before continuing on with the execution of 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 one to evaluate 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 execute incorrect activities. Join Conditions have been enhanced in WebSphere Process Server V6.0 to include support for XPath. Conditions can also be evaluated using Java programming, an IBM extension.



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

Human tasks are an IBM extension to the BPEL specification which allow 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 in a business process. The human task support is actually 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 a enterprise environment. There is integration with user registeries allowing for tasks to be assigned to different groups of individuals.

Java Snippets

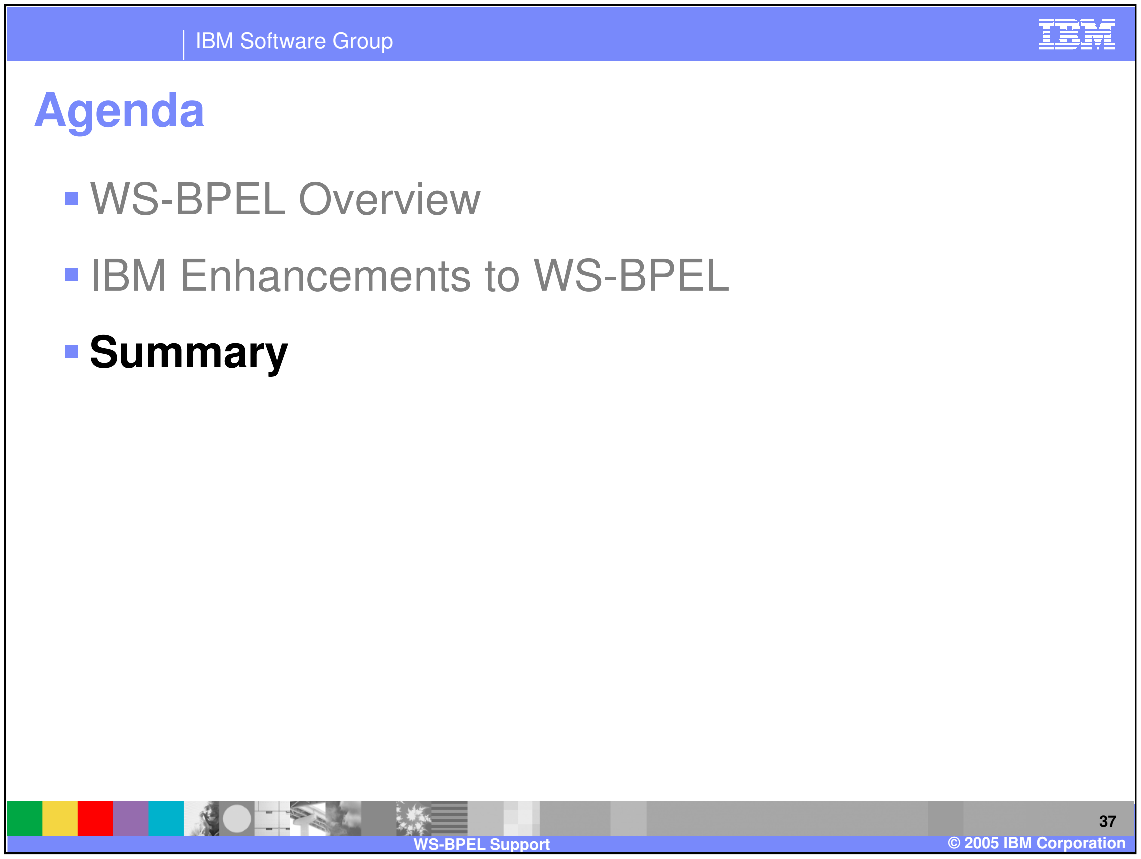
Enhanced in V6

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

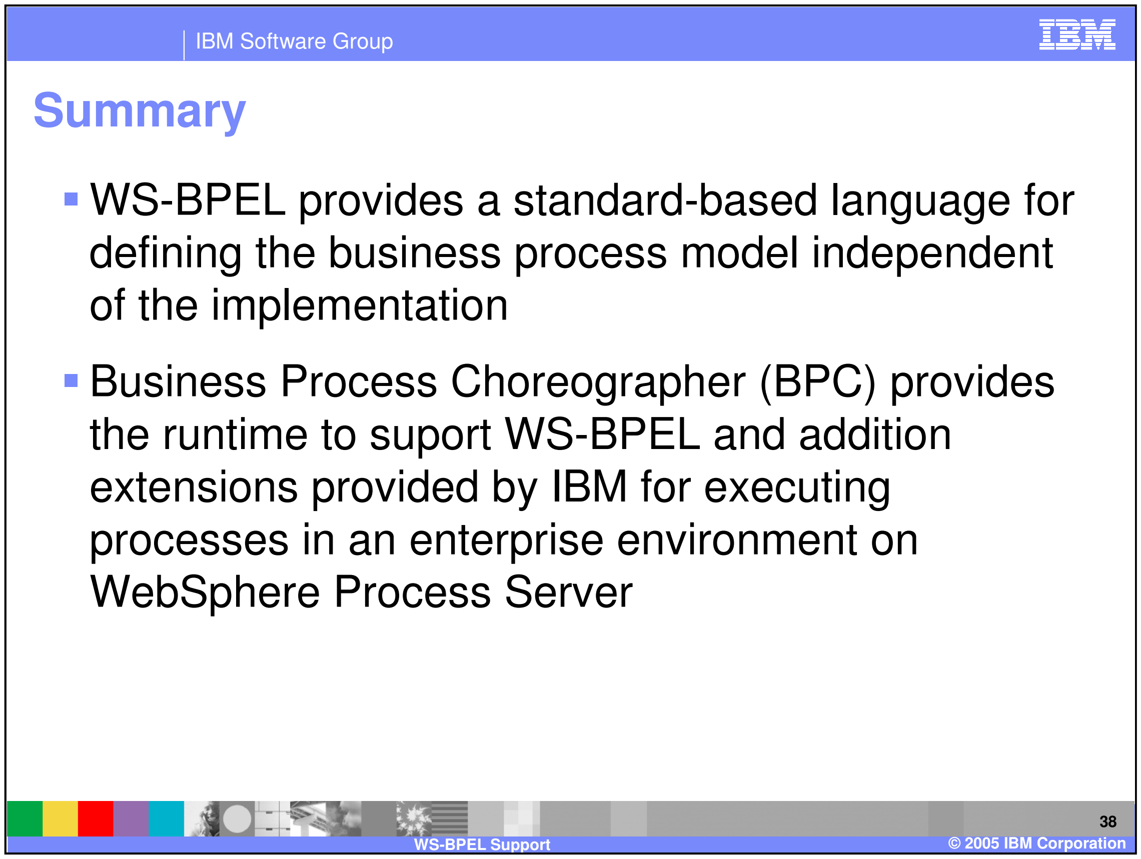
Snippets are an IBM extension to the BPEL specification. Formerly known as a Java snippet, this has been extended to include support for other types of snippets as well. WebSphere Process Server V6.0 includes a visual snippet editor, or you can program in Java directly. Snippets can be used to perform 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 persons name. Snippet activities should not be used for perfroming any type of external calls or processing.



Agenda

- WS-BPEL Overview
- IBM Enhancements to WS-BPEL
- **Summary**

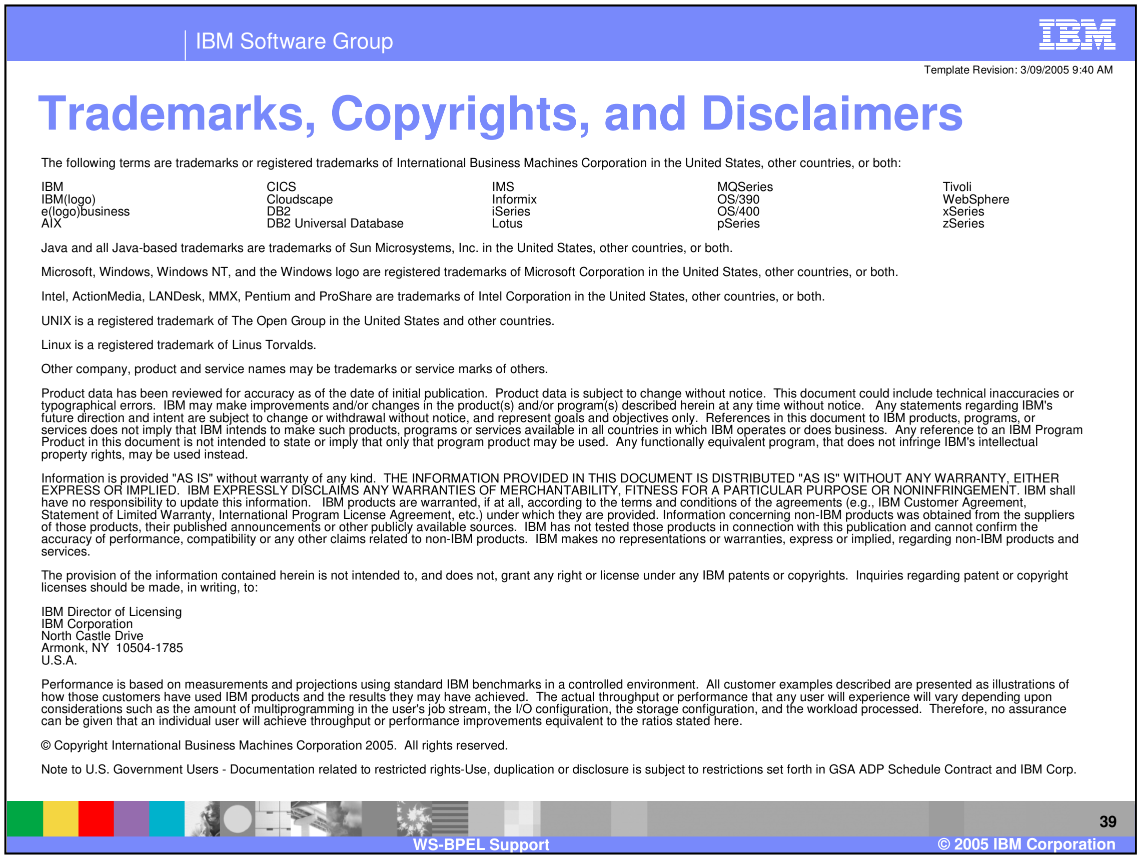
This section will provide a summary for the presentation.



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

In summary, Process Choreography provides a description language for defining business processes independent from the implementation. WebSphere Process Server and WebSphere Integration Developer V6.0 supports the proposed BPEL 2.0 specification and a number of issues that have been raised with it. There are number of enhancements over V5.1 such as event and compensation handlers, compensate activity, and rethrow, all built upon the core support provided in WebSphere Business Integration Server Foundation V5.1.



Template Revision: 3/09/2005 9:40 AM

Trademarks, Copyrights, and Disclaimers