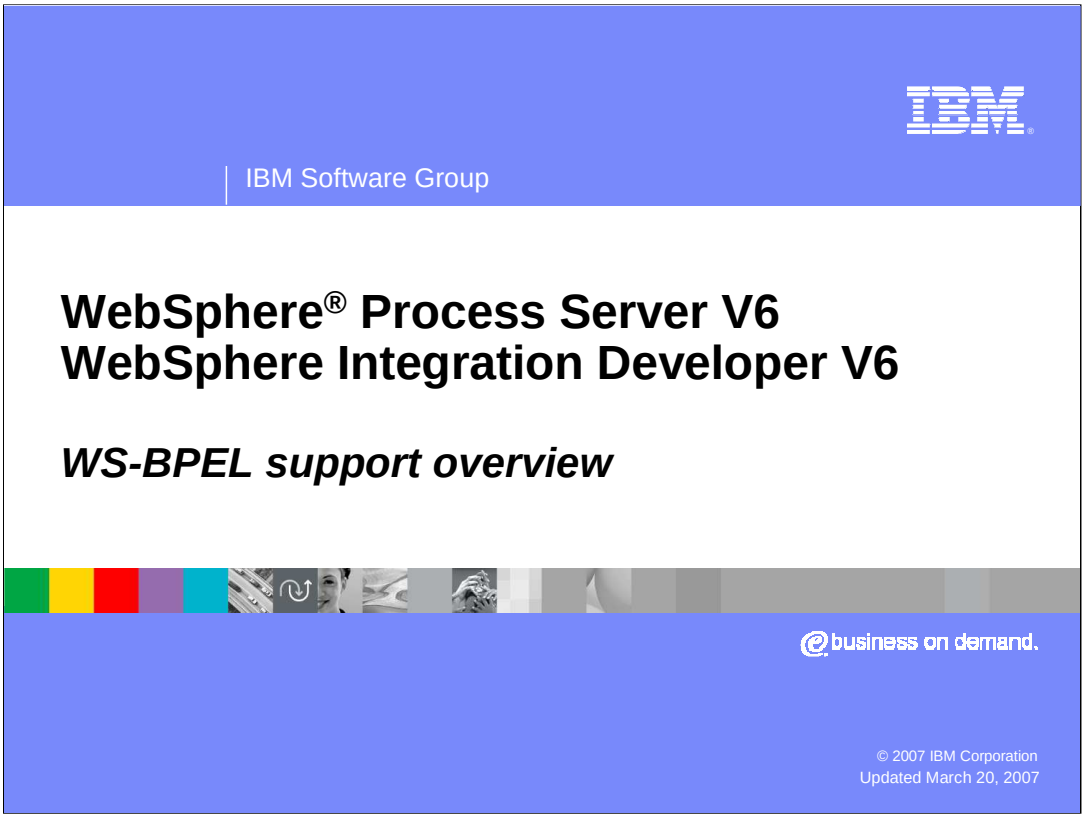


This presentation will provide an overview of the WS-BPEL specification and support provided for it in WebSphere Process Server V6 and WebSphere Integration Developer V6.

Goals

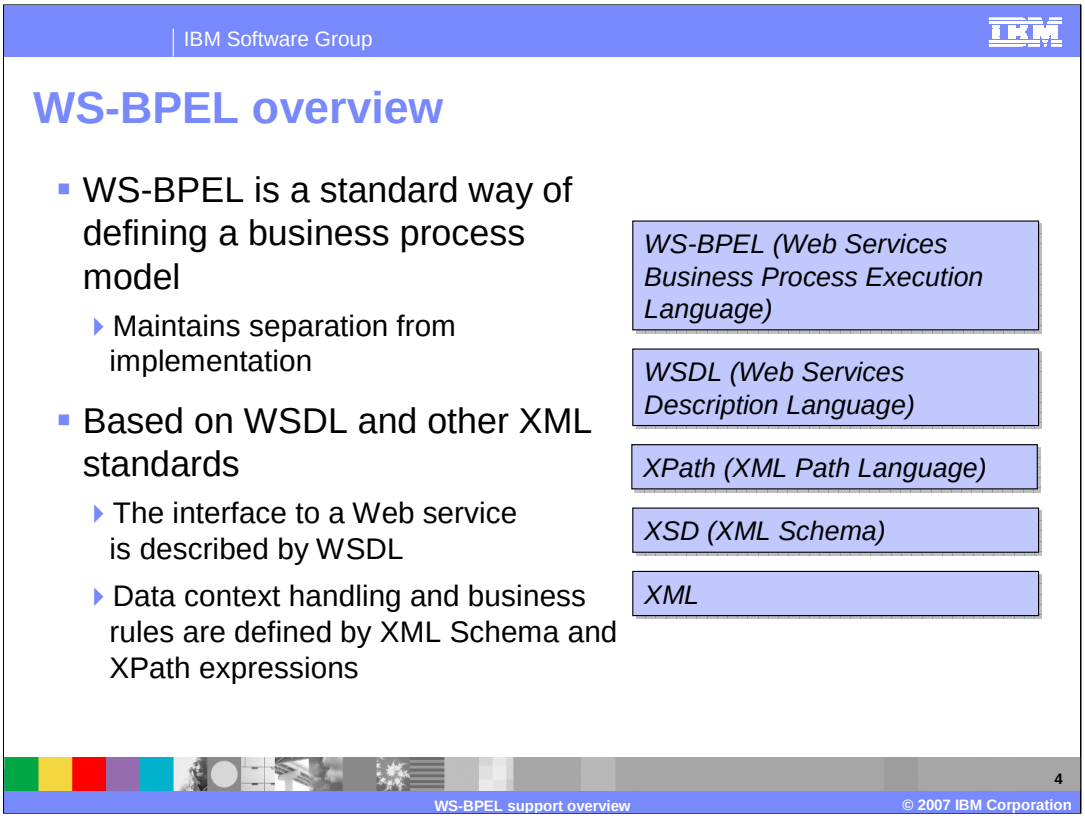
- Introduce Web Service Business Process Execution Language (WS-BPEL)
- Identify the enhancements to the support of the WS-BPEL specification

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 the enhancements to the specification that are supported by IBM.

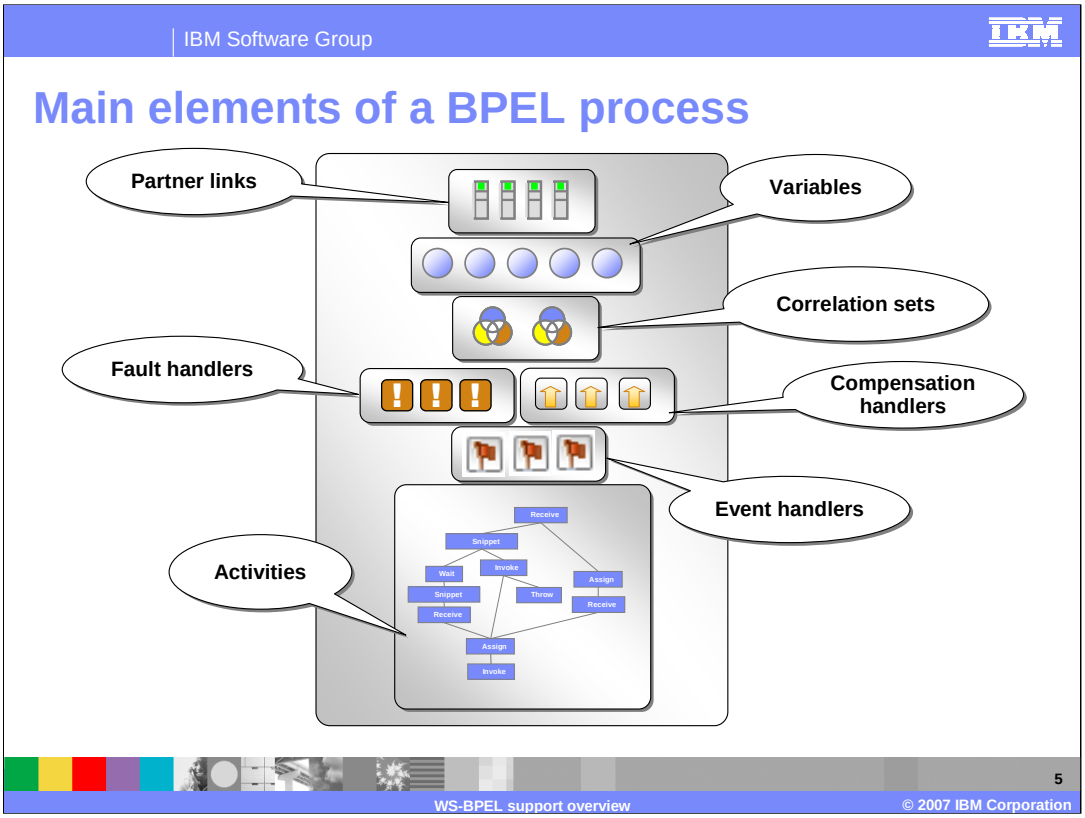
Agenda

- Overview of WS-BPEL Support
- Summary

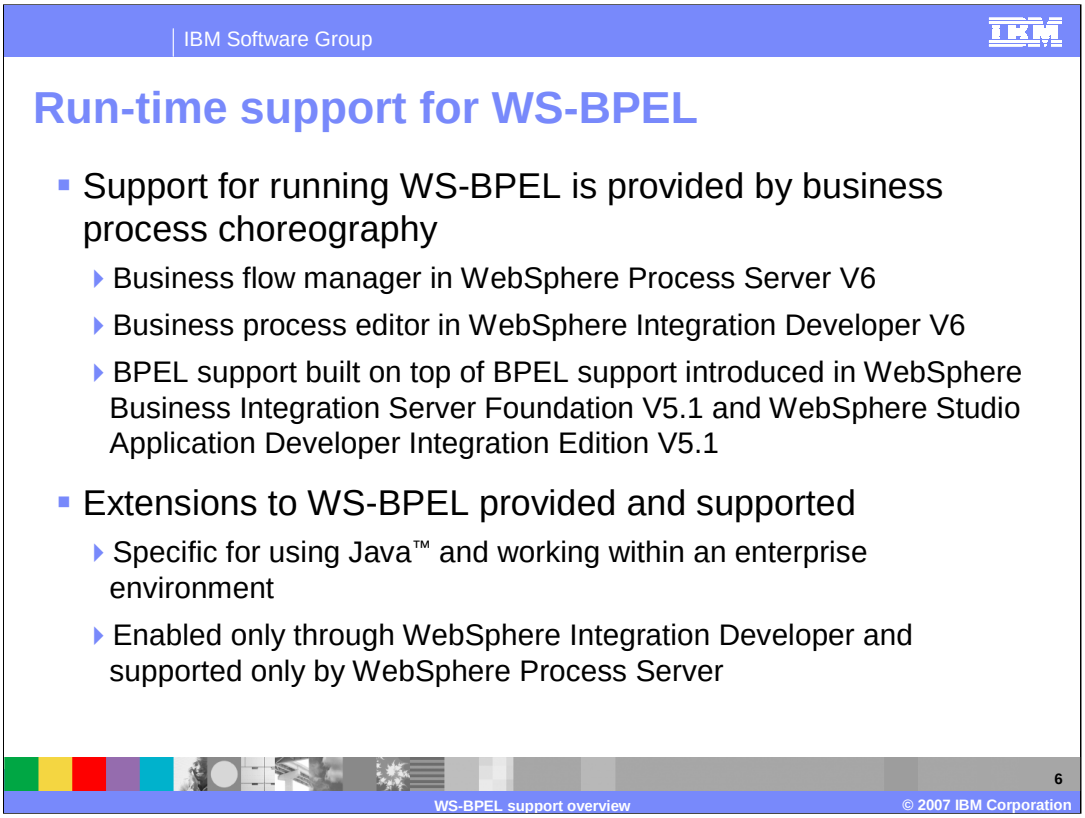
This section will provide an overview of the support for the WS-BPEL specification.



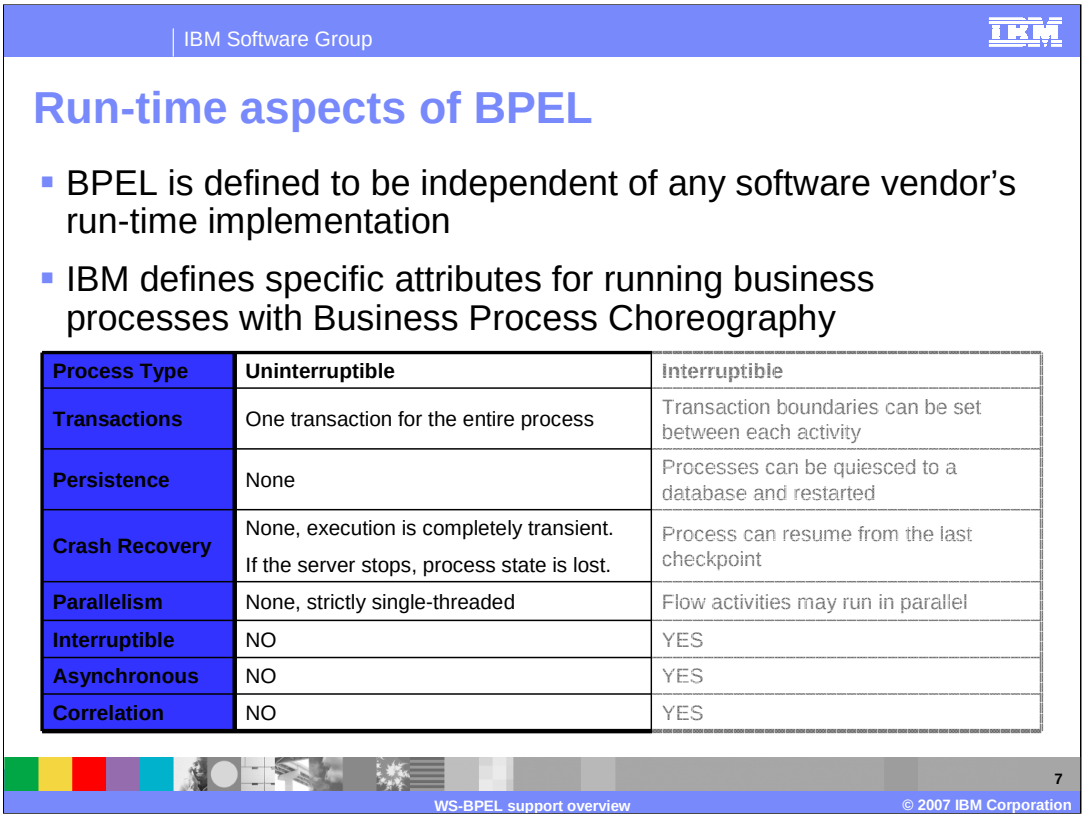
The WS-BPEL standard is finalizing at the version 2.0 level at the time of this recording. The standard allows business process models to be defined independent of the implementation, keeping the processes separate from the underlying infrastructure or technology. This fits nicely with the concept of 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.



The BPEL specification describes a number of areas for defining business processes. The primary elements or concepts are partner links, variables, correlation sets, fault handlers, compensation handlers, event handlers, and activities. These areas detail how a business process is organized and how it might be run.



Business processes defined in BPEL require robust runtimes that can interpret the business process and run the appropriate steps while considering things like integration with transactional environments and long running business processes that span hours, days, or even longer. WebSphere Process Server V6 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 and WebSphere Integration Developer V6 build on the support provided by WebSphere Business Integration Server Foundation V5.1.1 and WebSphere Studio Application Developer Integration Edition 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.

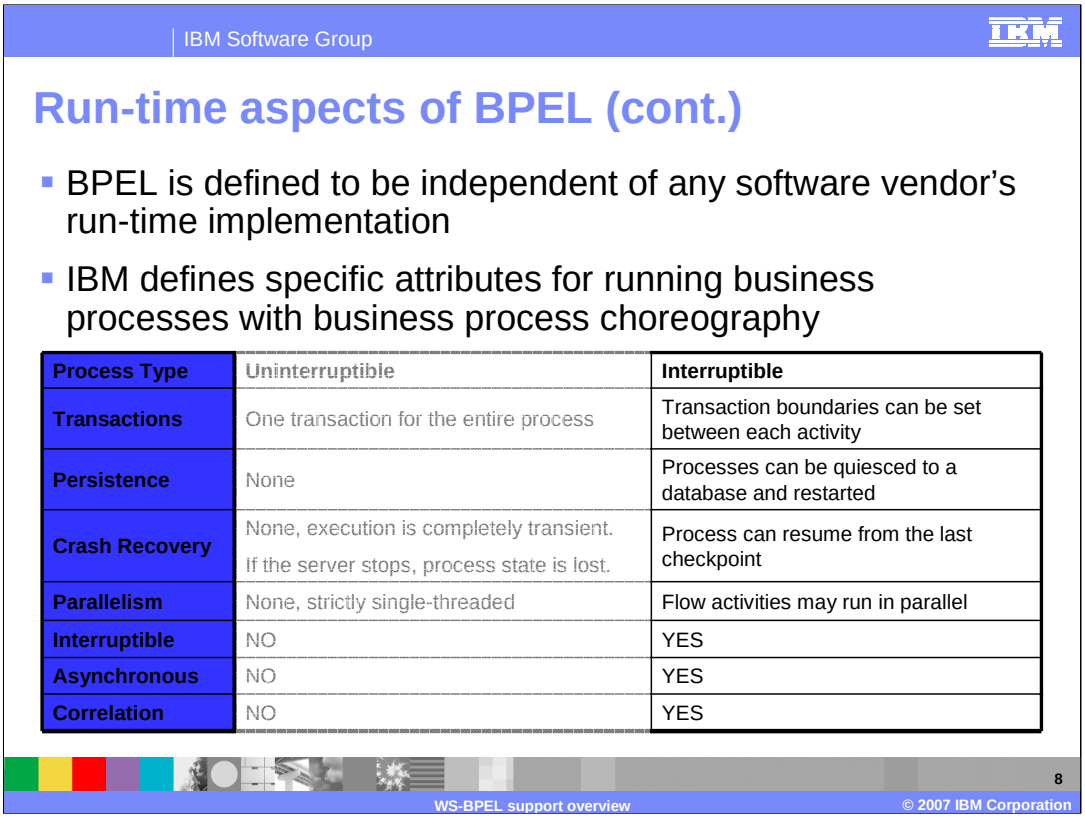


It is important to understand some of the runtime specific areas of support and BPEL extensions before going any further. Even though the goal of BPEL is to separate business logic from the underlying implementation, there are some implementation details that you need to be aware of in order to construct 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.

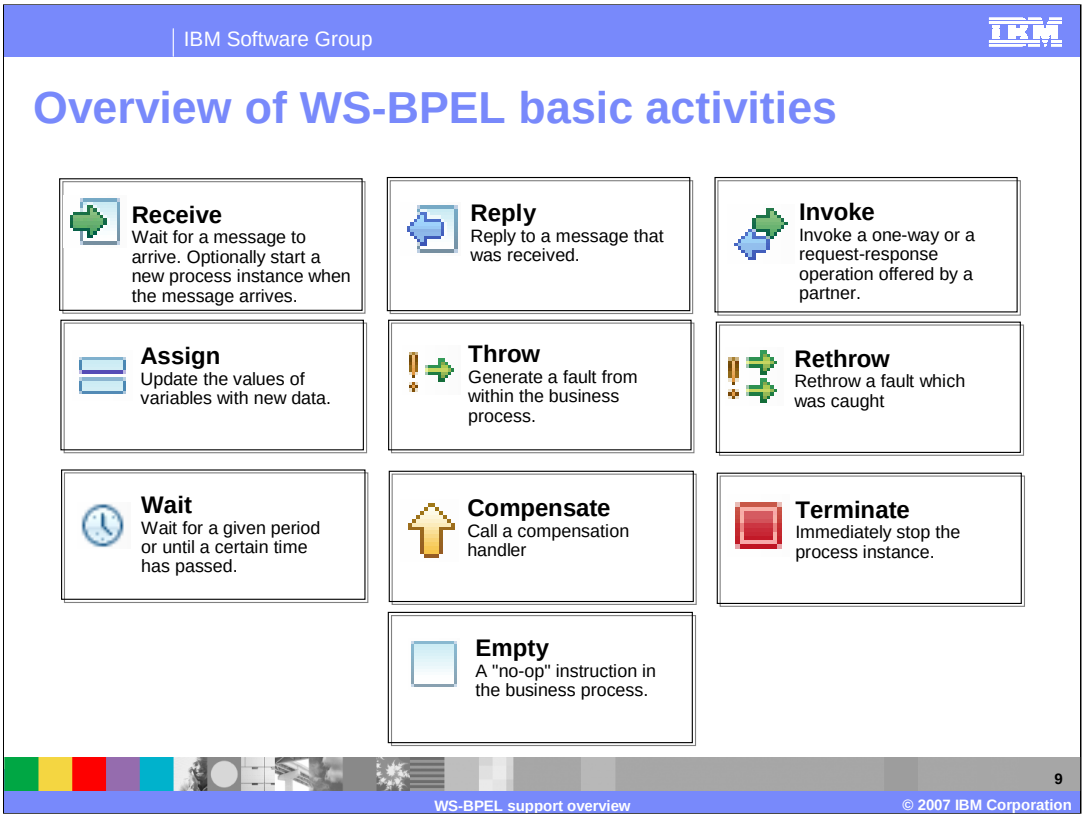
These runtime specific details are divided into the two primary areas of Uninterruptible, short-running business processes and Interruptible, long-running business processes.

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 run 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 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. All processing is synchronous and no correlation is provided.



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 run 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.



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. 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 providers. 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.

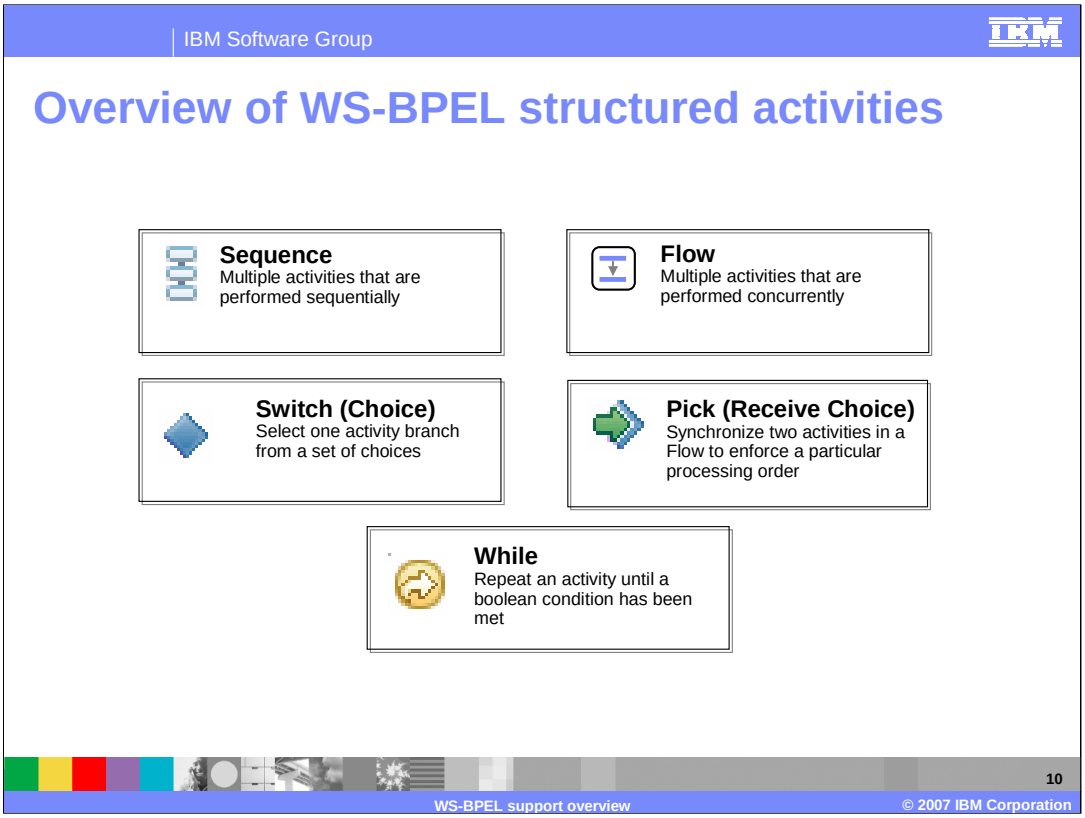
Assign activities are the primary way that data is transformed and moved from one variable to another.

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 because no further processing can occur.

If a failure is caught, a Rethrow activity can be used to throw the same 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. 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.



Here is an overview of structured activities, which are used to perform structured programing 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 running each iteration.

Section

Summary

This section will summarize 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 additional extensions provided by IBM for running 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 support the proposed BPEL 2.0 specification and address a number of issues that have been raised with it. There are number of enhancements, like event and compensation handlers, compensate activity, and rethrow, all built upon the core support provided in WebSphere Business Integration Server Foundation V5.1.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.