A standards-based approach to application integration

An introduction to IBM's WebSphere ESB product

Jim MacNair
Senior Consulting IT Specialist
Macnair@us.ibm.com

IBM WebSphere® Enterprise Service Bus (WebSphere ESB) V6.0.1 product is a component based integration platform built on a uniform programming model and a uniform data representation model. WebSphere ESB provides a flexible connectivity infrastructure designed to help you integrate applications and services as part of your service oriented architecture (SOA). An SOA separates implementation from interfaces. WebSphere ESB powers an SOA by separating the connectivity details from the interfaces, so that you can focus on your core business.

WebSphere® ESB V6.0.1 takes a new approach to application integration. It is based on industry standards and runs in a common J2EE environment, WebSphere® Application Server Network Deployment V6.

WebSphere® ESB is based on the Service Component Architecture (SCA), introduced in the WebSphere® Process Server V6.0 product. The SCA includes both the development of services and the connectivity of services into larger composite business processes, using an easy to use graphical tooling environment.

The WebSphere® Process Server product offers the infrastructure for the implementation and composition of services. The WebSphere® Application Server Network Deployment V6 product delivers the basic transport functions. The WebSphere ESB product offers capabilities to intelligently connect services. With WebSphere® ESB you can connect services faster and change existing connections more easily.

A flexible and manageable approach to an SOA

A Service Oriented Architecture (SOA) is a means of integrating an enterprise. It allows enterprises to increase their flexibility and supports the reuse of existing assets.

Services are composed into integrated processes, without the need to write or modify programs. By connecting through an ESB, the individual services can be loosely coupled. The services can be separately maintained, managed and enhanced without requiring two or more integrated applications to be changed at the same time. An ESB offers a flexible and manageable approach to support the implementation of a Service Oriented Architecture.

What is an Enterprise Service Bus?

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating application and services. There is a core set of functions that is included in most definitions. The core functions are:

- Route messages between service requestors and service providers
- Convert transport protocol between requestor and provider
- Transforms message formats between requestor and provider
- Distributes business events

The ESB must provide JMS messaging and Web services communications.

Industry standards

Industry standards are an important characteristic of any Enterprise Service Bus. These standards are popular because they can provide the wide interoperability necessary for services on a variety of platforms and environments to work together. Standards can provide application portability between different implementations of a standard.

Which standards are most relevant to an ESB?

There is a long list of standards that could apply to an Enterprise Service Bus (ESB). However, the following standards are likely to be the most relevant to an ESB.

- XML
- XSLT/XPath
- Web services (WS-*)
- JAX-RPC and JSR-109
- JMS
- SOAP
- WSDL
- UDDI

XML is a standard, platform independent method of representing data. The combination of XSLT and XPath provide a standard method for the identification and transformation of XML data.

If XML provides a standard representation of data, then the Web services standards provide a standard way of accessing the data. There are a variety of Web services standards that are relevant, including the WS-I specification and the WS-Security specification. The JAX-RPC and JSR-109 protocols provide a standard method of invoking a web service. WebSphere Application Server V6 provides strong support for relevant Web services standards. SOAP, Web Services Definition Language (WSDL) and UDDI provide important standards for the interoperability and platform independence of services.

JMS provides a common messaging application programming interface for the Java programming environment. A messaging infrastructure can provide the benefits of loose coupling and reliable communications.

While industry standards are an important part of an ESB, it is possible to draw some erroneous conclusions based on misunderstandings of what industry standards do and do not cover.

Do standards mean that my choice of vendor is not important?

The best way to keep up with current and emerging standards is to choose a vendor like IBM who is committed to standards. You cannot buy industry standards. Rather, you must buy a solution (product) that implements a standard from a vendor. It is the vendor who implements the standards and who must stay current with new and evolving standards. A vendor's commitment to invest in standards now and in the future is more important than a typical point in time comparison.

It is easy to think that the choice of vendor is not important, since applications can easily be moved to a different vendor's product in the future. Standards are often aimed at interoperability rather than portability. A robust and complete product offering requires more than the implementation of a list of standards. Many applications that use industry standards are not easily portable.

Standards do not cover non-functional requirements such as performance, scalability and availability. These can vary widely among products that support a particular standard.

An architected approach to integration

The WebSphere ESB product uses an architected approach to integrate applications. IBM developed this architecture to allow tooling that simplifies and quickens development and that shortens the associated learning curves of all of the underlying technologies generated by the tool.

The Service Component Architecture (SCA) is a programming model that simplifies the process of developing composite business applications by providing a common component invocation method and a common data representation. It includes support for both the implementation of services ("programming in the small") and the composing of services ("programming in the large"). Services can be developed in a variety of programming models, including stateless session EJBs, Web services, EIS services, BPEL4WS and database access.

The SCA architecture is designed for use with a graphical assembly editor. This function, provided in WebSphere® Integration Developer V6, allows programmers to create services and less-skilled developers to assemble the services into processes. Components are generally independent of the communications protocols, and an administrator can change the bindings without changing the component. The function contained in the WebSphere® Integration Developer allows lower-skilled developers to connect services and change existing connections.

Business objects are used to present all data in a standardized format, based on the Service Data Object standard, with extensions to support business integration. The Service Data Object is a hierarchy of typed data, with the necessary programming interfaces to access the data. The Service Message Object extends business objects to provide access to message header information as well as context areas.

The SCA and BO technologies provide a uniform service-oriented view over existing component models and APIs. This encourages best-practice implementations and simplifies the developer's view of the application modules and middleware APIs. As a result, the developer can focus on the business logic and data rather than the technology.

Standards Based Messaging

WebSphere ESB provides full support for JMS 1.1 applications. Services can be accessed and can access other services using JMS bindings. Messaging provides the benefits of loose-coupling between applications. Applications can be managed and developed independently, without low-level integration concerns causing complicated interdependencies.

A number of access patterns are supported, including point to point, request/reply, store and forward and publish/subscribe. A correlation context provides explicit support for the request/reply pattern.

Support for Web services

The WebSphere ESB product is built on a WebSphere Application Server Network Deployment V6 base. It has very strong support for advanced Web services, including SOAP, UDDI, WS-I, WS-Security, WS-Atomic Transactions, JSR 109 and JSR 101. Full support is provided for Web services bindings, including SOAP over HTTP.

Extending the reach of the ESB

Access to the ESB can be extended with support for JMS and Web services clients, interoperation with WebSphere MQ backbone networks and support for WebSphere Adapters.

A number of client platforms, including C and C++, .NET, J2EE and J2SE Java environments, are provided. The support includes access to both the JMS messaging and JAX/RPC Web services communications.

In addition to the client support, the WebSphere ESB product offers strong connectivity into WebSphere MQ backbone networks. This support allows the WebSphere Messaging bus to connect to a WebSphere MQ network using channel connections. The WebSphere MQ network appears as an extension of the WebSphere Messaging network.

Popular packaged applications can be accessed through the new WebSphere Adapters. This includes popular ERP systems such as SAP®, CRM systems such as PeopleSoft® and Siebel®. WebSphere ESB also supports most of the existing WBI Adapters.

A common runtime and development environment saves money

A common run-time environment for both the ESB and application server environments allows for common administration and operational procedures and reuse of skills. SCA modules are packaged as standard EAR files and deployed to a WebSphere ESB runtime server. The WebSphere ESB runtime server is capable of running WebSphere Application Server V6 applications as well as mediation modules. Administration is performed using the standard WebSphere administration console and scripting capabilities. Additional panels are added to the administration console to simplify the administration of SCA modules.

The WebSphere ESB product is built on a standard WebSphere Application Server (WAS) Network Deployment V6 server. It inherits all the capabilities of the WAS platform, including clustering, fail-over, scalability and security. WAS V6 includes strong support for standards, including JMS 1.1 messaging and the many WS-* and other Web services standards.

WebSphere ESB shares a common development tool with the IBM WebSphere Process Server V6.0.1® offering, namely WebSphere Integration Developer V6.0.1, which is available separately. By using a common Eclipse-based development environment, all developers can use a common toolset. Developers who perform different roles, such as writing java programs and developing mediation flows have common ways of performing common tasks.

The WebSphere Integration Developer V6.0.1® product can be integrated with other IBM development environments, including Rational Application Developer V6.0.1®.

Summary

The new IBM WebSphere ESB product offers a standards-based approach to providing the functions of an Enterprise Service Bus. It supports new service-oriented applications, including the use of standard Web services for communications. WebSphere Application Server V6 offers strong Web services support, including support for all relevant standards. WebSphere ESB takes advantage of a common runtime environment to reuse skills and reduce costs.

The IBM WebSphere ESB product is ideally suited when a departmental or line of business ESB solution is required and when ESB functions are required for a Web services environment.

Appendix A. Technical Details

This appendix provides technical details of the new WebSphere Enterprise Service Bus (ESB) V6.0.1 product, currently scheduled to be delivered by the end of 2005.

The deployable artifact is a mediation module, packaged as an EAR file. The mediation module contains the mediation component as well as other SCA components such as an Export and one or more Imports.

Modules and Libraries in WebSphere Integration Developer V6.0.1

A new module type of Mediation Module is introduced in WebSphere ESB V6.0.1. Mediation modules can execute in either WebSphere ESB V6.0.1 or WebSphere Process Server V6.0.1. The existing business module projects introduced in WebSphere Integration Developer V6.0 will only run on WebSphere Process Server environments. Library projects, also introduced in WebSphere Integration Developer V6.0, can hold shared resources such as business objects and interfaces for either mediation or process (business) modules.

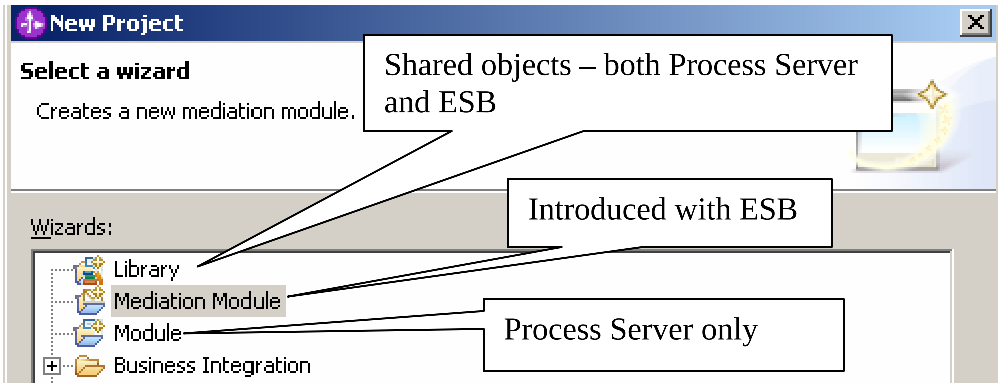


Figure 1- New Project Wizard

SCA Export and Import

SCA Import and Exports enable WebSphere ESB to:

- Provide SCA components with a uniform view of services external to the module

- Allow components to communicate with EIS systems using a consistent SCA programming model
- Support multiple types of bindings including J2C and JMS

SCA supports both synchronous and asynchronous invocation patterns. In the case of synchronous calls, the caller is blocked until a reply is received. In the case of asynchronous calls, control is returned immediately to the caller. The SCA supports the following asynchronous invocation patterns.

- One Way: The request is sent. No reply is expected or received.
- Deferred Response: A ticket is returned to the caller. The caller can use the ticket to retrieve the response. The caller has a choice of *no wait*, *wait forever* or *wait for a limited time*.
- Request with callback: A ticket is returned immediately and a callback is registered. SCA will invoke the callback method to return the response.

Mediation components

The WebSphere ESB product adds a mediation component type to the SCA architecture. A mediation component is constructed using a visual editor, the mediation flow editor. A mediation flow consists of a collection of mediation primitives wired together. The developer of the mediation flow uses an easy drag and drop paradigm to create the mediation component. IBM provides a number of pre-built mediation primitives. Each mediation primitive has an input terminal and usually one or more output terminals.

The mediation primitives are wired together to produce a mediation flow component. The mediation flow component is then connected to other SCA components using the assembly editor. The assembly editor is also used to provide inputs and outputs, specify interfaces and select the type of bindings to be used.

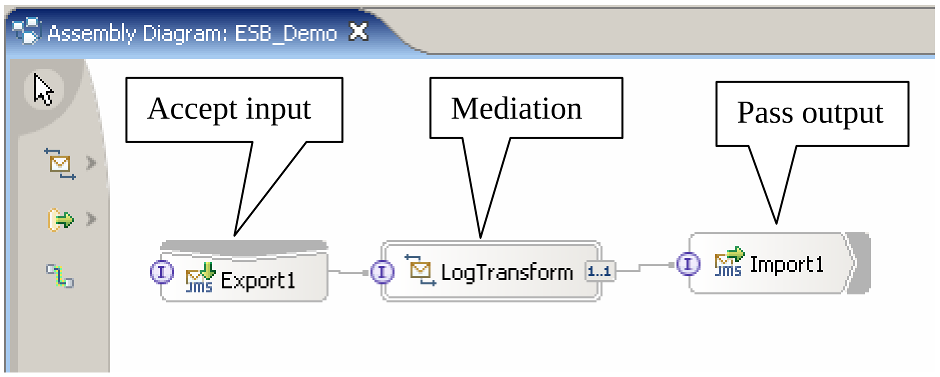
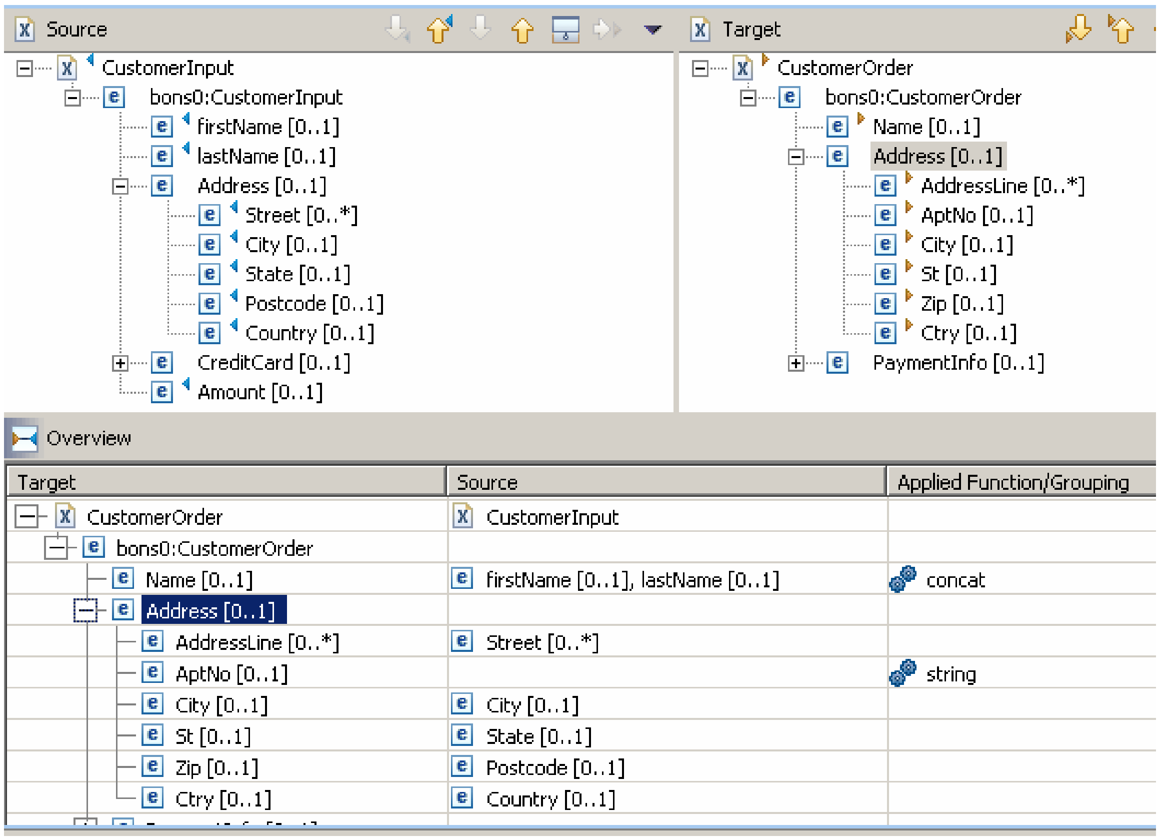


Figure 2 Assembly Editor example

Mediation primitives

The supplied mediation primitives include XSLT transformation, message logging, database access and a filter or routing function. A programmer can also create mediation primitives. Custom primitives can be either Java routines or complete nodes that can be added to the developer's palette and reused in multiple mediation flows.

The XSLT transformation mediation primitive transforms XML documents from one XML format to another. The transformation can also access any JMS and/or SOAP message headers and properties, as well as two context areas. The XSLT transformation can be developed with a graphical drag and drop mapping editor.



Source

- CustomerInput
  - bons0:CustomerInput
    - firstName [0..1]
    - lastName [0..1]
    - Address [0..1]
      - Street [0..*]
      - City [0..1]
      - State [0..1]
      - Postcode [0..1]
      - Country [0..1]
    - CreditCard [0..1]
    - Amount [0..1]

Target

- CustomerOrder
  - bons0:CustomerOrder
    - Name [0..1]
    - Address [0..1]
      - AddressLine [0..*]
      - AptNo [0..1]
      - City [0..1]
      - St [0..1]
      - Zip [0..1]
      - Ctry [0..1]
    - PaymentInfo [0..1]

Overview

Target	Source	Applied Function/Grouping
CustomerOrder	CustomerInput	
bons0:CustomerOrder		
Name [0..1]	firstName [0..1], lastName [0..1]	concat
Address [0..1]		
AddressLine [0..*]	Street [0..*]	
AptNo [0..1]		string
City [0..1]	City [0..1]	
St [0..1]	State [0..1]	
Zip [0..1]	Postcode [0..1]	
Ctry [0..1]	Country [0..1]	

Figure 3 XSLT Mapping Editor example

The message logging mediation primitive will write the contents of a request to a database. This could be used for auditing and recovery purposes, for example.

The database mediation primitive can retrieve a value from a database and insert it into the user data. This can be used for message augmentation for example.

The filter mediation primitive provides conditional and branching capabilities within a mediation flow. This could be used for message routing that depends on the contents of the message for example. The filter mediation primitive supports a variable number of output terminals. A list of XPath expressions is used to determine which terminal or terminals the message should be propagated to.

A developer can create custom mediation primitives using Java. There are two types of custom mediation primitives. In the first case, a developer creates a reusable mediation primitive that can is added to the mediation flow editor palette. The custom nodes can be used in multiple mediation flows. In the second case, a developer creates unique function in Java for a particular node in a particular flow. In this case, the developer implements a defined method, creating a plain java object. The custom node is not used in other mediation flows.

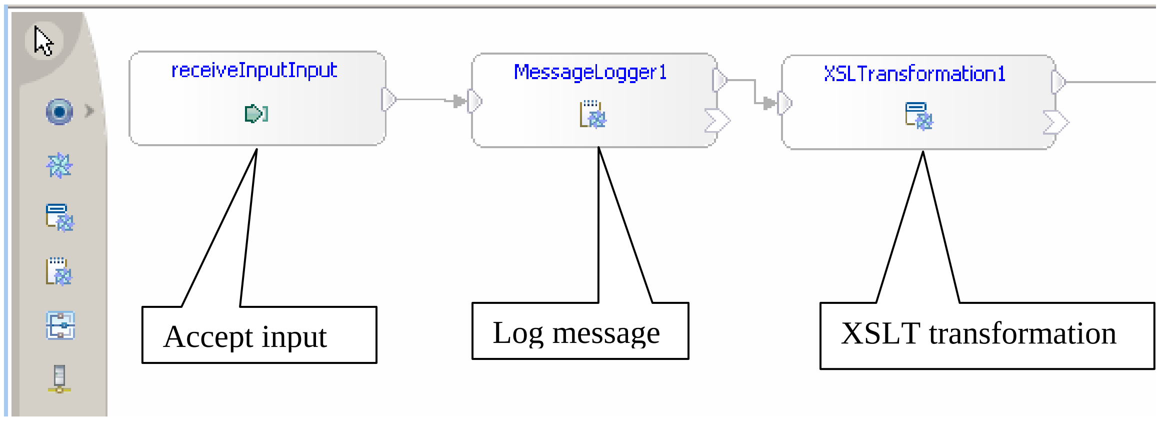


Figure 4 Mediation Flow Editor example

Interfaces

In a Services Oriented Architecture (SOA), services must expose interfaces so they can be called. The Service Component Architecture supports two types of interfaces, namely those that can be described using Web Services Description Language (WSDL) and Java interfaces. A graphical interface editor is provided for the creation and maintenance of WSDL interfaces.

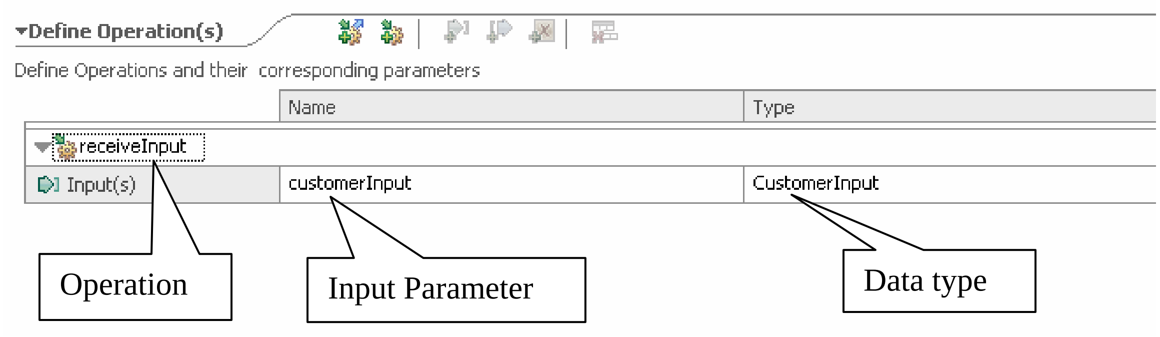


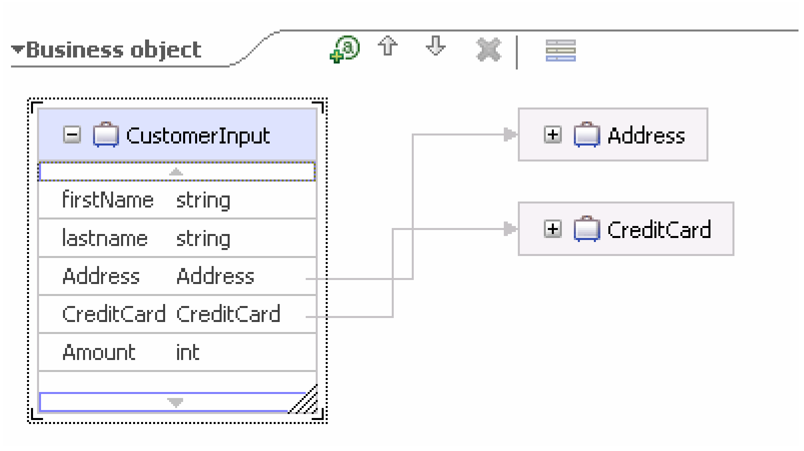
Figure 5 Interface Editor example

The Service Message Object

A mediation flow has full access to not only the message contents (user data) but to any message headers and properties (e.g. JMS and/or SOAP) and to two message context areas. The transient context is used as a temporary work area during the processing of a single message. A second context area can be used in request and reply scenarios. Data can be saved in the correlation context area that is made available when a reply message is processed by the mediation.

The message contents, message headers and context areas are children of the service message object. The service message object is itself a service data object (SDO).

The user data is represented as a business object. A business object is stored using a subset of the XML schema file syntax. An easy to use business object editor simplifies the creation and manipulation of business objects.



firstName	string
lastname	string
Address	Address
CreditCard	CreditCard
Amount	int

Figure 6 Business Object Editor

A full function XML schema editor is also available for more complex data formats.

Visual Debugger

A visual debugger is provided in the WebSphere Integration Developer V6.0.1 environment. It allows Mediation flows to be debugged. The debugger provides the standard debugging functions, including breakpoints, single stepping, display of variables, etc.

Appendix B. What is the relationship of WebSphere Process Server and WebSphere ESB?

WebSphere ESB uses the Service Component Architecture (SCA) introduced in IBM's WebSphere Process Server V6.0 product. WebSphere ESB V6.0.1 is a proper subset of WebSphere Process Server V6.0.1. Both products use the same WebSphere Integration Developer V6.0.1 (sold separately) to create mediation flows. In particular, WebSphere ESB adds the following additional functions:

- A new Mediation Flow component type
- Introduces a Service Message Object to extend the Business Objects in WebSphere Process Server
    - Makes message headers and context visible to the flow
- Adds additional administration panels for mediation flows

The following functions are present in WebSphere Process Server but not in the WebSphere ESB runtime.

- Business Process (BPEL4WS) support
- Human task manager
- Business Rules
- State machine
- Data and Interface maps
- Selectors

The WebSphere Process Server product offers the infrastructure for the implementation and composition of services. The WebSphere ESB product offers capabilities to intelligently connect services.