WebSphere® Process Server for z/OS

Version 6.0.2

Product Overview

Note
Before using this information, be sure to read the general information in "Notices" on page 57.

30 March 2007

This edition applies to version 6, release 0, modification 2 of WebSphere Process Server for z/OS (product number 5655-N53) and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Introduction to WebSphere Process Server

IBM WebSphere Process Server is the next generation business process integration server that has evolved from proven business integration concepts, application server technologies, and the latest open standards.

IBM® WebSphere® Process Server, which supports a service-oriented architecture (SOA), is the ideal platform for business applications that require business integration using different technologies. Using the WebSphere Integration Developer tool set, business integration solutions can be created using simplified integration mechanisms, such as the Service Component Architecture (SCA) programming model and the Service Data Objects (SDO) data model. SDO business objects can be defined, transformed, routed, and mapped using SCA components. WebSphere Adapters supply connectivity to back-end Enterprise Information Systems (EIS). With WebSphere Process Server, business integration applications can define business logic and processes based on Web Services - Business Process Execution Language (BPEL), human tasks, and business rules. For the runtime monitoring of the business integration solutions, WebSphere Process Server provides Common Event Infrastructure (CEI), which centralizes the monitoring of the various events that can occur in these applications.

WebSphere Process Server enables deployment of standards-based process integration solutions in an SOA. SOA is a discipline and a framework that takes everyday business applications and breaks them down into individual business functions and processes, rendering them as services. SOA is a conceptual description of the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services and connections between components. This means that a well defined set of business-level interfaces for the components can be created and maintained, shielded from lower-level technology changes. Loosely coupled integration applications that are based on SOA provide flexibility and agility. You can implement integration solutions independent of platform, protocols and products. For more information about SOA, refer to the Service-Oriented Architecture (SOA) from IBM Web site.

Hardware and software requirements

To view the official statement of supported hardware and software for WebSphere Process Server, go to the WebSphere Process Server system requirements Web site.

Information roadmaps

WebSphere Process Server can be used by a variety of users including *solution deployers*, who deploy solutions developed in WebSphere Integration Developer to WebSphere Process Server, *solution administrators*, who use the administrative console to administer solutions, and *operators*, who use the administrative console to operate a solution.

To help you to navigate through the available information sources, both within and beyond the product information centers, WebSphere Process Server roadmaps are available online from IBM developerWorks®. These roadmaps list high level goals based on your user role and point to documentation resources that are useful to accomplishing your goals in WebSphere Process Server.

To access information roadmaps for WebSphere Process Server, go to the WebSphere Business Integration information roadmaps page on IBM developerWorks.

Chapter 2. What is new in this release

IBM WebSphere Process Server for z/OS, version 6.0.2, includes several new features.

Welcome to IBM WebSphere Process Server for z/OS, version 6.0.2, which includes the following new features:

- Easier and fast integration between WebSphere Process Server, WebSphere ESB, and WebSphere Application Server and WebSphere Message Broker and WebSphere MQ with new WebSphere MQ JMS binding.
- Service governance with dynamic runtime lookup and invocation of services (integration with WebSphere Service Registry and Repository).
- Easy-to-use, comprehensive human-centric business process management scenarios:
  - Web client generation tool for business users to generate user interfaces and task lists.
  - Customizable out-of-the-box administration and configuration client for administrators.
  - Enhanced support to build clients for business users from powerful JSF components for both Web-based and WebSphere Portal clients.
  - Secure Web services interface for generic workflow client applications on any platform, including .NET and J2EE (easier access for remote clients).
  - Remote client install option, which provides an environment for custom remote clients that use the WebSphere Process Server APIs (not just on the same system as WebSphere Process Server).
  - Ability to assign work to a group, or team, of individuals who all share the same job or responsibility, in shifts or in parallel.
  - Line of business users can create additional tasks (insert additional tasks into a task-list), schedule follow-up work for the same user, or follow-on tasks for co-workers, and handle events that were not planned for in advance.
  - Customized e-mail message for human task escalations (for example, to alert a manager that a task has not been processed).
  - Post-processing of staff query results to add customer-specific workforce management policies (such as staff workload balancing, and substitute when absent) and integration with additional custom staff repositories.
  - Business processes queries and filters based on client-specific process data (such as order ID or client name) that can be saved as private views.
  - Task redistribution based on organizational staff and responsibility changes, using a timer-controlled daemon.
  - Server-controlled page flow capability to automatically present users the next task at hand within the same business process.
  - Graphical view of processes to track status and drill down into each individual activity.
- Runtime administration improvements (dynamic reconfiguration, with no need to rebuild or redeploy):
  - Administration configuration of mediation properties and end points.

  - Dynamic end-point selection: the administrator can intervene to get part of the process to interface to a different system (for example, to change from Oracle to Siebel dynamically for future process instances).
  - Ability to handle unmodeled faults from Web service invocations.
  - Ability to add mediation modules after deployment without going into WebSphere Integration Developer.
  - Event sequencing: Preserve the processing sequence of events in which they were generated remotely.
  - Configurable clean-up service to automatically delete business process instances from the database.
  - Graphical charts and flexible drill-down capabilities for historical and accumulated data such as average process duration or actual work time.
- Tight integration between information services and business processes:
  - Information Service Activity that provides direct access to relational database systems with the support of full SQL and interaction with other information management services (for example, Extract, Transform, Load) and federated access to heterogeneous information sources such as ECM systems.
  - Support for WebSphere Integration Developer plug-in for information management activities.
- Business rules improvements:
  - Support for more common business rule logic scenarios, including specifying initialization logic for a decision table, specifying an ″otherwise″ clause on decision table conditions and using ″return″ option available in rule sets to force running the rule list to end.
  - Simplified rule programming model to lower development time and cost.
  - Improved visibility to business rule changes with new audit capabilities, including support for an ″approval″ scenario as rules are promoted from one environment to another.
  - New business rule import and export capability to simplify maintaining consistency between server or environment instances.
- Support for IBM DB2 on z/OS as a remote data base management system.
- Simplified server configuration using the WebSphere Application Server Network Deployment console for clusters and multiple cells.
- Additional platform support for Solaris 10 (SPARC and x86-64), HP-UX 11i2 (PA-RISC), SuSE Linux Enterprise Server 10, Red Hat Enterprise Linux 4, and Linux on zSeries (64-bit).

Chapter 3. Product family overview

IBM WebSphere Process Server works with several other IBM products including WebSphere Integration Developer, WebSphere Application Server, WebSphere Adapters, WebSphere Business Integration Adapters, WebSphere Application Server Toolkit, WebSphere Business Modeler, WebSphere Business Monitor, Rational Application Developer, Rational Software Architect, WebSphere Partner Gateway, and WebSphere Portal.

IBM WebSphere Process Server is based on the robust J2EE 1.4 infrastructure and associated platform services provided by WebSphere Application Server, version 6.0.2. WebSphere Process Server is built on WebSphere Application Server, Network Deployment, version 6.0.2. WebSphere Process Server also works with infrastructure and platform services from WebSphere Application Server, version 6.0.2. For more information about WebSphere Application Server, see the WebSphere Application Server for z/OS documentation.

IBM WebSphere Integration Developer is the development environment for WebSphere Process Server. For more information about WebSphere Integration Developer, see the IBM WebSphere Business Process Management Version 6.0 information center.

WebSphere Process Server and WebSphere Integration Developer include additional capabilities that make it possible to model, build, deploy, install, configure, run, monitor, and manage integration applications. WebSphere Integration Developer complements IBM WebSphere Business Modeler, version 6.0, and IBM WebSphere Business Monitor, version 6.0, and can be used in conjunction with IBM Rational® Application Developer, version 6.0.x, or IBM Rational Software Architect, version 6.0.x, to create a unique, integrated and powerful integration development platform. For more information about these products, see the WebSphere Business Modeler information center, the WebSphere Business Monitor information center, the Rational Application Developer Information Center, and the Rational Software Architect Information Center.

WebSphere Process Server is powered by the same technology available with IBM WebSphere Enterprise Service Bus. Enterprise service bus capability is part of the underlying functionality of WebSphere Process Server, and no additional license for WebSphere Enterprise Service Bus is required to take advantage of these capabilities. However, you can deploy additional stand-alone licenses of WebSphere Enterprise Service Bus around your enterprise to extend the connectivity reach of the process integration solutions powered by WebSphere Process Server. For example, WebSphere Enterprise Service Bus can be installed closer to an SAP application to host an IBM WebSphere Adapter for SAP and to transform SAP messages before sending that information across the network to a business process choreographed by WebSphere Process Server.

IBM WebSphere Adapters, version 6.0.x, and IBM WebSphere Business Integration Adapters (based on IBM WebSphere Business Integration Framework, version 2.6) allow for integration of existing Enterprise Information System infrastructure and applications that are deployed on WebSphere Process Server. For more information about these products, see the IBM WebSphere Business Integration information center.

In addition, you can extend existing applications for reuse in enterprise processes with an IBM enterprise modernization portfolio that includes IBM CICS® Transaction Gateway and IBM WebSphere Host Access Transformation Services. For more information about these products, see the CICS Transaction Gateway Library and the WebSphere Host Access Transformation Services (HATS) Information Center.

The IBM WebSphere Application Server Toolkit is a set of basic tools that help you to assemble, test, and deploy Web services in WebSphere Process Server. For more information, see the WebSphere Application Server Toolkit documentation on the WebSphere Application Server, Version 6.0 Information Center.

IBM WebSphere Partner Gateway used with WebSphere Process Server supports business-to-business applications. A limited license of WebSphere Partner Gateway is included with WebSphere Process Server. For more information about WebSphere Partner Gateway, see the WebSphere Partner Gateway documentation.

IBM WebSphere Portal provides access to various administrative functions and allows portlets to have access to business processes and other Service Component Architecture services in WebSphere Process Server. For more information about WebSphere Portal, see the WebSphere Portal Documentation Library.

Chapter 4. Architectural overview of WebSphere Process Server

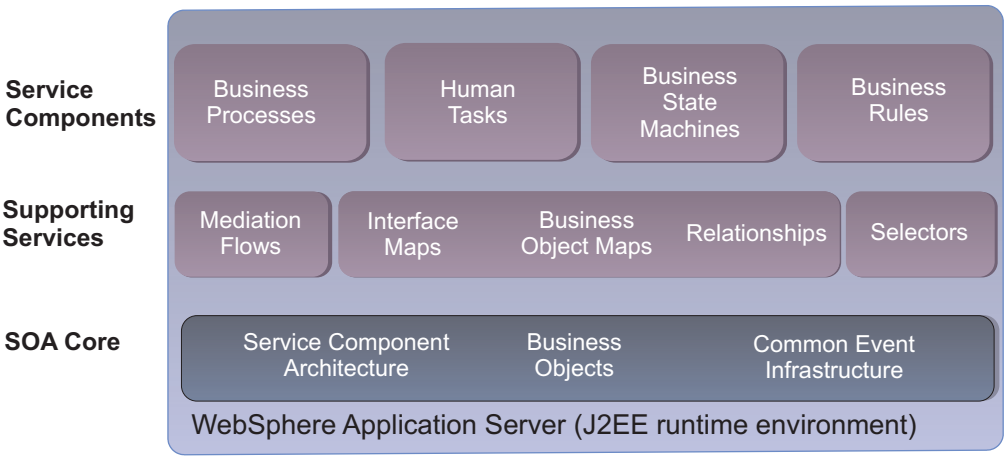
IBM WebSphere Process Server combines integration capabilities with a composite application platform to deliver an integration platform with a fully converged, standards-based business process engine, using the full power of WebSphere Application Server.

IBM WebSphere Process Server is a service-oriented architecture (SOA) integration platform built on a uniform invocation programming model and a uniform data representation model.

The base runtime infrastructure for WebSphere Process Server is WebSphere Application Server. The Service Component Architecture and business objects that are part of the SOA core provide uniform invocation and data-representation programming models. The SOA core includes the Common Event Infrastructure for generating events for the monitoring and management of applications running on WebSphere Process Server. Supporting services provide the foundational business object and transformation framework for WebSphere Process Server. Service components represent the functional components required for composite applications.

The combination of a powerful foundation (WebSphere Application Server and the SOA Core) and service components in WebSphere Process Server allows quick development and deployment of sophisticated composite applications that run on WebSphere Process Server.

One component-based framework addresses all styles of integration.

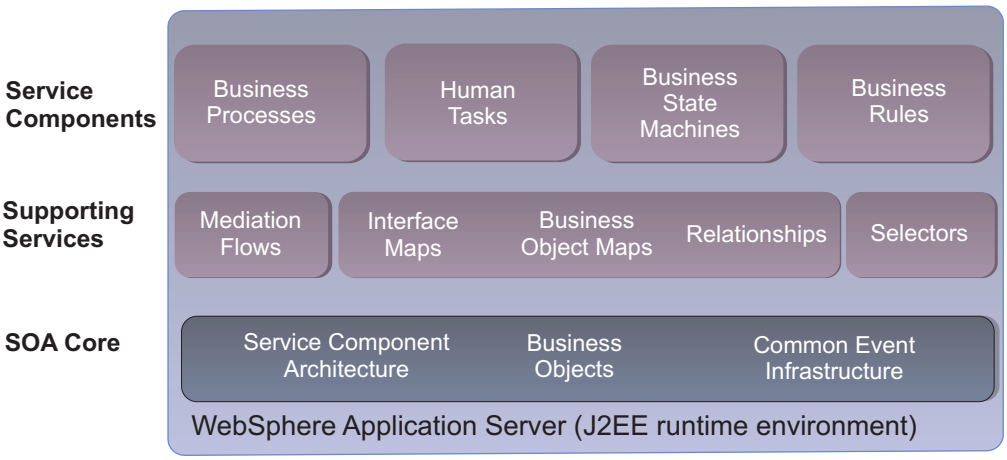


Service-oriented architecture core

The service-oriented architecture (SOA) core of WebSphere Process Server provides both uniform invocation and data-representation programming models and monitoring and management capabilities for applications running on WebSphere Process Server.

SOA is a conceptual description of the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services and connections between components. The Service Component Architecture and business objects that are part of the SOA core provide uniform invocation and data-representation programming models for applications deployed on IBM WebSphere Process Server. The SOA core also includes the Common Event Infrastructure for generating events for the monitoring and management of applications on WebSphere Process Server.

One component-based framework addresses all styles of integration.



Service Component Architecture

Service Component Architecture presents all elements of business transactions – access to web services, Enterprise Information System (EIS) service assets, business rules, workflows, databases and so on – in a service-oriented way.

Service Component Architecture separates business logic from implementation, so that you can focus on assembling an integrated application without knowing implementation details. The implementation of business processes is contained in service components.

Service components can be assembled graphically in the IBM WebSphere Integration Developer tools, and the implementation can be added later. The Service Component Architecture programming model narrows what developers must know about Java™ and J2EE or other implementation in particular scenarios to a core set of language concepts that are familiar to all who develop business applications in other programming languages today. This allows developers to quickly and easily integrate technologies.

Developers switching from classical application development environments face a much smaller learning curve; they can quickly become productive with this programming model. The Service Component Architecture programming model also helps experienced J2EE developers be more productive.

Service Component Architecture supports several standard service implementation types:

- Java objects, which implement a Java class. As in the Java programming language, instances of Java components at run time are referred to as Java objects.

- Business process components, which implement a business process. The implementation language is the Business Process Execution Language (BPEL) and its IBM extensions.
- Human task components, which represent and implement a task typically performed by a person in a business process or an integration application.
- Business state machine components, which are used when applications work with artifacts that have a set of states. A state machine defines what the artifacts can do at a point in time.
- Business rule components, which determine the outcome of a business process based on a context and can be designed as if-then rules, decision tables, or decision trees. Business rules within a business process allow applications to respond quickly to changing business conditions. The rules are independent of the business process itself, and you can change them at any time without having to redo your process.

Service qualifiers govern the interaction between a service client and a service on the WebSphere Process Server runtime environment. Service qualifiers are quality of service specifications that define a set of communication characteristics required by an application for transmission priority, level of route reliability, transaction management, and security level. An application communicates its quality of service needs to a runtime environment by specifying service qualifiers. Quality of service qualifiers can be specified when wiring components in the assembly editor in WebSphere Integration Developer. These specifications, when running on WebSphere Process Server, determine how the clients interact with the target components. Depending on the qualifiers specified, the run time can supply additional required processing.

Importing and exporting capabilities within the Service Component Architecture define a service module's external interfaces or access points for WebSphere Process Server. Imports and exports can be either to other modules within a same application, or to other applications on enterprise information systems (EIS). This allows working with IBM WebSphere Adapters and IBM WebSphere Business Integration Adapters. For more information, see Imports, exports, and adapters.

WebSphere Process Server solutions rely upon the underlying WebSphere Application Server capabilities for transaction, security, and workload management to provide a scalable integration environment.

For business processes, WebSphere Process Server offers support for transactions involving multiple resource managers using the two-phase commit process to ensure atomic, consistent, isolated, and durable (ACID) properties. This capability is available for both short-running flows (single transaction) and long-running flows (multiple transactions). You can group multiple steps in a business process into one transaction by modifying transaction boundaries in WebSphere Integration Developer.

Because not all service invocations support two-phase-commit transactions, WebSphere Process Server also includes recovery capabilities. If a failure occurs in the middle of running an integration application, the server detects it and allows an administrator to manage the failed event from the failed event manager.

Service Data Objects and business objects

Service Data Objects and business objects define the data flowing between components that are defined in Service Component Architecture.

Service Data Objects (SDOs), part of WebSphere Application Server capabilities that are built into WebSphere Process Server, provide a framework for data application development that simplifies the J2EE data programming model.

WebSphere Process Server includes business objects, which are enhanced SDOs. Business objects are based on a data-access technology called Service Data Objects. SDOs provide a universal means of describing disparate data (like JDBC ResultSet, XML Schema described data, for example). Business objects include some extensions that are important for integration solutions and are used to further describe the data that is being exchanged between Service Component Architecture services. Business objects are part of the Service-oriented architecture (SOA) core of WebSphere Process Server.

A business object is a set of attributes that represent a business entity (such as Employee), an action on the data (such as a create or update operation), and instructions for processing the data. Components of the integration application use business objects to exchange information and trigger actions. Business objects are flexible because they can represent many kinds of data. For example, in addition to supporting the data canonicalization model of traditional integration servers, they also can represent data returned from a synchronous EJB Session Bean facade or a synchronous business process, and then they can be bound to IBM WebSphere Portal portlets and JSF components.

Business objects are the primary mechanism for representing business entities, or documenting literal message definitions, enabling everything from a simple basic object with scalar properties to a large, complex hierarchy or graph of objects.

In WebSphere Process Server, business object framework is made up of these elements:

- Business object definition
- Business graph definition
- Business object metadata definition
- Business object services (service APIs)

A business object definition is the name, set of ordered attributes, properties, version number, and application-specific text that specify a type of business object. A business graph definition is the wrapper added around a simple business object or a hierarchy of business objects to provide additional capabilities, such as carrying change summary and event summary information related to the business objects in the business graph. A business object metadata definition is the metadata that can be added to business object definitions to enhance their value when running on WebSphere Process Server. This metadata is added to the business object's XML schema definition as well known xs:annotation and xs:appinfo elements. Business object services are a set of capabilities provided on top of the basic capabilities provided by WebSphere Application Server Service Data Objects. Examples are services such as create, copy, equality, and serialization.

For more information about WebSphere Application Server Service Data Objects, see the WebSphere Application Server, Version 6.0 Information Center.

Common Event Infrastructure

The Common Event Infrastructure is an embedded technology within WebSphere Process Server to provide basic event management services.

The Common Event Infrastructure (CEI) provides functionality for generation, propagation, persistence, and consumption of events representing service component processes. A standard, XML-based format, the Common Base Event model, defines the structure of these events. Each type of event used by the server contains a number of standard fields specific to a given type of event. In some cases, it contains an encapsulation of the business object data that is being used by the service component at a particular event point.

IBM WebSphere Process Server uses events in the CEI almost exclusively to enable service component monitoring. You must configure the CEI server if you want to use event-related functions, but after that, you should not use CEI directly. Instead, use the existing services in WebSphere Process Server.

In WebSphere Process Server, a specially configured CEI Server -- which can be part of an existing process server or another server -- is used for all event-related services. You must first create and deploy several facilities that are used by the CEI Server, including an event database, a messaging engine, one or more enterprise applications, and a database driver.

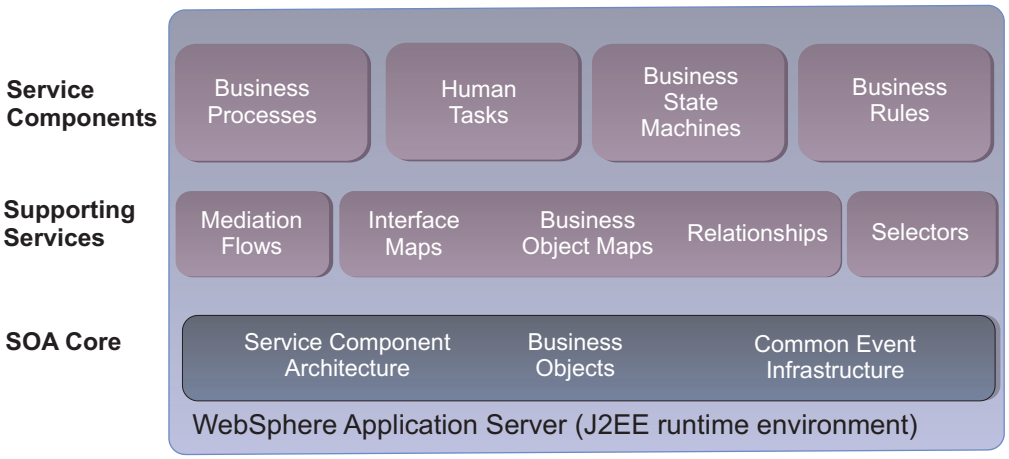
For more information about Common Event Infrastructure, see the *Administering WebSphere Process Server* PDF file.

Supporting services

Supporting services in IBM WebSphere Process Server address a number of transformation challenges for connecting components and external artifacts.

You can use mediation flows, interface maps, business object maps, relationships, and selectors to integrate applications running on IBM WebSphere Process Server.

One component-based framework addresses all styles of integration.



Mediation flows

Mediation flows intercept and modify messages that are passed between existing services (providers) and clients (requesters) that want to use those services.

A mediation flow mediates or intervenes between an export and import to provide functions such as message logging, data transformation and routing. Mediation flows are created in IBM WebSphere Integration Developer and deployed as part of a mediation module in WebSphere Process Server.

Related concepts

Chapter 7, “Enterprise service bus messaging infrastructure,” on page 21
WebSphere Process Server includes enterprise service bus capabilities. WebSphere Process Server supports the integration of service-oriented, message-oriented and event-driven technologies to provide a standards-based, messaging infrastructure in an integrated enterprise service bus.

Interface maps

Interface maps reconcile the differences between components that have different interfaces.

Interface maps are supporting service components in IBM WebSphere Process Server that resolve and reconcile differences between interfaces in other Service Component Architecture (SCA) components to enable them to communicate. The interface map captures a first-class pattern that allows module designers in IBM WebSphere Integration Developer to reconcile differences across multiple interfaces using transforms and other rudimentary operations. Interface maps are deployed on WebSphere Process Server as part of modules, also called SCA modules.

Business object maps

Business object maps support mappings between the source and target business objects.

Business object maps are supporting service components in IBM WebSphere Process Server that assign values to the target business objects service components based on the values in the source business objects service components.

Developers create the business object maps in IBM WebSphere Integration Developer.

Relationships

Relationships are supporting services in WebSphere Process Server applications that establish an association between data from two or more data types.

In IBM WebSphere Process Server, relationship manager is a tool for manually manipulating relationship data to correct errors found in automated relationship management or provide more complete relationship information. In particular, it provides a facility for retrieving as well as modifying relationship instance data. Relationship manager allows you to configure, query, view, and perform operations on relationship runtime data, including participants and their data. You create relationship definitions with relationship designer. At run time, instances of the relationships are populated with the data that associates information from different applications.

For more information about relationship manager, see the *Administering WebSphere Process Server* PDF file.

Selectors

Selectors provide flexibility at points in the processing of components in an application running on IBM WebSphere Process Server.

Selectors are supporting services that provide flexibility at points in the processing of service components during run time. A selector takes one invocation and allows different targets to be called based on the selection criteria.

Selectors add additional flexibility beyond business rules. Business rules are a fundamental part of businesses. Business rules drive the general processing of an application, invoking certain services to get the data through the application. For example, a rule may be: Two weeks before school starts, offer a back-to-school special price on our school-related merchandise. A selector takes one invocation and allows different targets to be called based on the selection criteria. For example, if the time is just before school starts, then the previous back-to-school offer would be called. However, if the season is the just as school ends, then a get-your-kids-ready-for-summer offer would be called.

The application is portable because it calls the same thing all the time. The business rule never changes. The actual processing differs (and calls different service components) because of the selector.

For more information about selectors, also called selector components, see the *Administering WebSphere Process Server* PDF file.

Service components

All integration artifacts running on IBM WebSphere Process Server (for example, business processes, business rules, and human tasks) are represented as components with well defined interfaces. Within the Service Component Architecture (SCA), a service component defines a service implementation.

Because all integration artifacts are represented as service components, also called SCA components, IBM WebSphere Process Server creates an environment with unparalleled flexibility. SCA components each have an interface and can be wired together to form a module deployed to WebSphere Process Server. This enables changing any part of an application without affecting the other parts. It is possible, for example, to replace a human task for an approval with a business rule for automatic approval simply by replacing the components in the assembly diagram without changing either a business process or the caller of the business process.

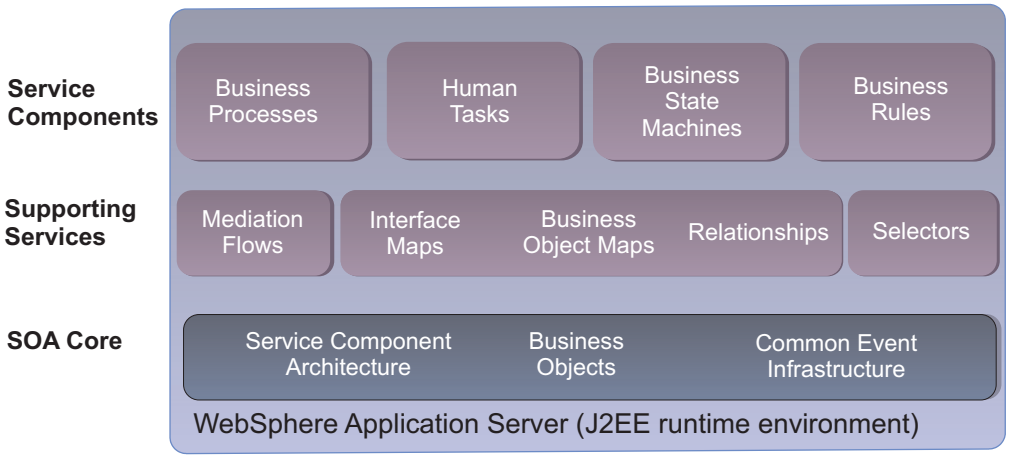
Components can interact with existing applications, using the following programming constructs:

- Java Beans
- Enterprise Java Beans
- Web services
- JMS messages

In addition, components can interact with other applications on enterprise information systems (EIS) with IBM WebSphere Adapters, version 6.0.x and WebSphere Business Integration Adapters, based on WebSphere Business Integration Framework, version 2.6.

On top of the runtime infrastructure supporting services and the service-oriented architecture core, WebSphere Process Server offers a variety of ready-to-use SCA service components that can be used in integration applications.

One component-based framework addresses all styles of integration.



Business processes

Business processes are service components that provide the primary means through which enterprise services are integrated.

A business process is any system or procedure that an organization uses to achieve a larger business goal. When you break it down, you see that a business process is actually a series of individual tasks, and each task is executed in a specific order. As an integral part of applications running on IBM WebSphere Process Server, business processes provide the primary means through which enterprise services are integrated.

Business process components implement a fully supported Web Services Business Process Execution Language (BPEL) engine. WebSphere Process Server includes a business process choreography engine on top of the WebSphere Application Server. You can develop and deploy complex business processes in a simple development model with sophisticated support for long and short running business processes in a highly scalable infrastructure. You can either create BPEL models in WebSphere Integration Developer, version 6.0.x, or import them from a business model you created in WebSphere Business Modeler, version 6.0.x.

Web Services Business Process Execution Language (BPEL) is used to choreograph the flow of business processes. Business process integration services build on BPEL4WS version 1.1 and add major capabilities of the upcoming WS-BPEL version 2.0 specification.

For more information about business processes, see the WebSphere Process Server *Installing* PDF file.

Human tasks

Human tasks are stand-alone service components that can be used to either assign work to employees or to invoke other services.

The Human Task Manager, available in IBM WebSphere Process Server, supports ad-hoc creation and tracking of tasks. Existing LDAP directories (as well as operating system repositories and the WebSphere user registry) can be used to access user and group information. WebSphere Process Server supports multi-level

escalation for human tasks including e-mail notification. It also includes a Web client to manage human tasks, and a set of Java Server Faces (JSF) components that can be used to create custom clients or to embed human task functionality into other Web applications.

Human task services allow role-based task assignment, invocation and escalation.

For more information about human tasks, see the WebSphere Process Server *Installing* PDF file.

Business state machines

Business state machines are service components that specify the sequences of states, responses, and actions that an object or an interaction goes through during its life in response to events.

A business state machine provides another way of modeling a business process. This gives you the choice of representing business processes based on states and events rather then a sequential business process model.

For more information about monitoring business state machines in WebSphere Process Server, refer to the WebSphere Process Server *Monitoring* PDF file.

Business rules

Business rules are service components that declare policy or conditions that must be satisfied within your business.

Business rules make business processes more flexible. Because business rules determine the outcome of a process based on a context, using business rules within a business process allows applications to respond quickly to changing business conditions.

Business rule authoring is supported with IBM WebSphere Integration Developer. IBM WebSphere Process Server includes the business rules manager, a Web-based runtime tool for business analysts to update business rules as business needs dictate, without affecting other components or Service Component Architecture (SCA) services.

For more information about administering business rules in WebSphere Process Server, see the *Administering WebSphere Process Server* PDF file.

Chapter 5. Imports, exports and adapters

Importing and exporting capabilities within the Service Component Architecture define a service module's external interfaces or access points for IBM WebSphere Process Server. Imports and exports can be either to other modules within a same application, or to other applications on enterprise information systems (EIS).

Imports identify services outside of a module, making them callable from within the module. *Exports* allow components in a module to provide their services to external clients.

An import or export lets modules access other modules and lets your application access applications on EIS systems as if they were local components. This allows working with IBM WebSphere Adapters and IBM WebSphere Business Integration Adapters.

WebSphere Adapters, version 6.0.x, and WebSphere Business Integration Adapters (based on WebSphere Business Integration Framework, version 2.6) provide a service-oriented approach to EIS integration.

WebSphere Adapters are compliant with J2EE Connector Architecture (JCA 1.5). JCA is the J2EE standard for EIS connectivity. EIS Import and EIS Export provide SCA components with the uniform view of the services external to the module. This allows components to communicate with the variety of external EIS systems using the consistent SCA programming model. WebSphere Adapters are assembled in WebSphere Integration Developer from imported RAR files and then exported as an enterprise archive (EAR) file and deployed on WebSphere Process Server.

WebSphere Adapters include the following:

- IBM WebSphere Adapter for Email, version 6.0.2
- IBM WebSphere Adapter for Flat Files, version 6.0.2
- IBM WebSphere Adapter for FTP, version 6.0.2
- IBM WebSphere Adapter for JD Edwards EnterpriseOne, version 6.0.2
- IBM WebSphere Adapter for JDBC, version 6.0.2
- IBM WebSphere Adapter for Oracle Ebusiness Suite, version 6.0.2
- IBM WebSphere Adapter for Siebel Business Applications, version 6.0.2
- IBM WebSphere Adapter for SAP Software, version 6.0.2

For more information about these products, see the IBM WebSphere Business Integration information center.

WebSphere Business Integration Adapters consist of a collection of software, Application Programming Interfaces, and tools to enable applications to exchange business data through an integration broker. Each business application requires its own application-specific adapter to participate in the business integration process. You can install, configure, and test the adapter using current WebSphere Business Integration Adapter Framework and Development Kit System Manager tools. You can use WebSphere Integration Developer to import existing business objects and connector configuration files, to generate artifacts, and to assemble the solution for WebSphere Process Server. Operational commands for the WebSphere Business Integration Adapters are part of the WebSphere Process Server administrative

console. For more information about working with these adapters and WebSphere Process Server, see the IBM WebSphere Business Integration information center.

Imports and exports require binding information, which specifies the means of transporting the data from the modules. The assembly editor in WebSphere Integration Developer sets up imports and exports, lists the bindings supported and simplifies the creation of them. A properties view displays the binding information.

Chapter 6. Clusters

Clusters are sets of managed servers that provide high availability and workload balancing for applications. Members of a cluster can be servers located on different host machines or servers located on the same host machine (the same node).

Clusters give your applications more capacity and availability than a single server. A clustered environment provides the following benefits:

- Workload balancing: By running application images on multiple servers, a cluster balances an application workload across the servers in the cluster.
- Processing power for the application: You can add processing power to your application by configuring additional server hardware as cluster members supporting the application.
- Application availability: When a server fails, the application continues to process work on the other servers in the cluster thereby allowing recovery efforts to proceed without affecting the application users.
- Maintainability: You can stop a server for planned maintenance without stopping application processing.
- Flexibility: You can add or remove capacity as needed by using the administrative console.

By nature, the z/OS environment is clustered and can provide a highly scalable environment without all the complexities of separate clustered servers. Separate clustered servers can be utilized for multisystem nodes to provide additional availability.

For more information about clusters, see Introduction: Clusters in the WebSphere Application Server for z/OS documentation.

Chapter 7. Enterprise service bus messaging infrastructure

WebSphere Process Server includes enterprise service bus capabilities. WebSphere Process Server supports the integration of service-oriented, message-oriented and event-driven technologies to provide a standards-based, messaging infrastructure in an integrated enterprise service bus.

The enterprise service capabilities that you can use for your enterprise applications provide not only a transport layer but mediation support to facilitate service interactions. The enterprise service bus is built around open standards and service-oriented architecture (SOA). It is based on the robust J2EE 1.4 infrastructure and associated platform services provided by IBM WebSphere Application Server Network Deployment, version 6.0.2.

IBM WebSphere Process Server is powered by the same technology available with IBM WebSphere Enterprise Service Bus. This capability is part of the underlying functionality of WebSphere Process Server, and no additional license for WebSphere Enterprise Service Bus is required to take advantage of these capabilities. However, you can deploy additional stand-alone licenses of WebSphere Enterprise Service Bus around your enterprise to extend the connectivity reach of the process integration solutions powered by WebSphere Process Server. For example, WebSphere Enterprise Service Bus can be installed closer to an SAP application to host an IBM WebSphere Adapter for SAP and to transform SAP messages before sending that information across the network to a business process choreographed by WebSphere Process Server.

Related concepts

"Mediation flows" on page 11
Mediation flows intercept and modify messages that are passed between existing services (providers) and clients (requesters) that want to use those services.

Overview of the bus environment

The bus environment comprises one or more service integration buses, ESB servers, and their resources, organized into logical administrative domains of cells and nodes.

Although a service integrator can deploy an SCA module without knowing anything about the bus environment, as an administrator you might want to manage servers and buses, and therefore need to understand what those tasks involve:

- The SCA runtime (exploited by mediation modules) uses queues on an SCA.SYSTEM service integration bus as a robust infrastructure to support asynchronous interactions between components and modules. The queues are hosted by the server as a member of the SCA.SYSTEM bus.
- The ESB server provides the integration technologies, infrastructure services, configuration, and runtime administration needed to run mediation modules and service applications in WebSphere Process Server. As a bus member, the server has a messaging engine that provides the core messaging functionality of the SCA.SYSTEM bus.

Both the server and SCA.SYSTEM bus are configured with default properties that might be suitable for you to deploy and run your SCA modules.

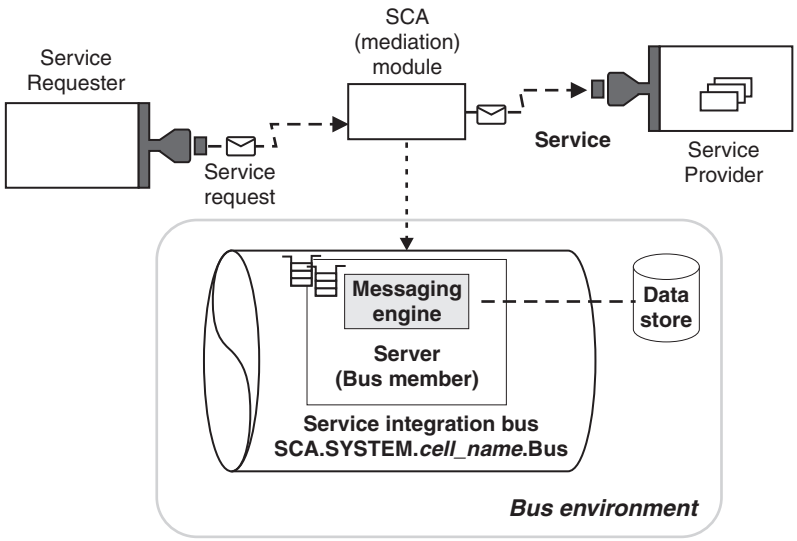


Figure 1. A bus environment with one server assigned to the SCA.SYSTEM service integration bus. As a bus member, the server is assigned one messaging engine, which has a data store for storing state data and messages. This figure also shows a mediation module deployed into the bus environment and assigned to the bus member.

For more advanced usage, you might want to change the configuration of the bus environment for a stand-alone server, or create a bus environment for a network deployment cell. For example:

- You can configure a variety of quality of service from secure, assured delivery (where messages are guaranteed not to get lost and are transported securely) to best-effort (where messages might get lost in case of a system failure).
- You might want to set up a network deployment cell to provide several servers to host mediation modules. This provides advantages of scalability, the ability to handle more client connections, and greater message throughput. You can also create server clusters, which enables you to manage a group of servers together and enables those servers to participate in workload management.
- Your complete bus environment might be made up of several stand-alone and deployment manager profiles, to provide separate administrative domains for different departments or to separate test and production facilities. Each profile has its own SCA.SYSTEM service integration bus.

Besides the SCA.SYSTEM bus used for SCA modules, you can also create other service integration buses that you can use to support the service integration logic provided by the modules. For example, the SCA.APPLICATION bus is provided and used to define JMS queue destinations and other JMS resources for modules deployed with JMS bindings.

You can create other buses for use as in WebSphere Application Server; for example, for applications acting as service requesters and providers within WebSphere Process Server, or to link to WebSphere MQ. You can also use a WebSphere Process Server deployment manager to manage separate application servers for use with applications and modules deployed onto WebSphere Application Server.

Related concepts

"Management of the server and bus environment" on page 31
With WebSphere Process Server, administrators can create an environment of ESB servers and service integration buses that support the deployment of service applications.

Clients

WebSphere Process Server provides Message Service clients that extend the connectivity of the enterprise service bus.

- Message Service Clients for C/C++ and .NET enable non-Java applications to connect to WebSphere Process Server.

  Message Service Clients for C/C++ and .NET provide an API called XMS that has the same set of interfaces as the Java Message Service (JMS) API. Message Service Client for C/C++ contains two implementations of XMS, one for use by C applications and another for use by C++ applications. Message Service Client for .NET contains a fully managed implementation of XMS, which can be used by any .NET compliant language.
- Web Services Client for C++ provides a set of libraries and Java tools that enable you to build ANSI C++ web service client applications from existing Web Service Description Language (WSDL) files.

  The ANSI C++ web service client applications that you build from existing WSDL files, using the Web Services Client for C++ libraries and Java tools, are able to communicate with other similarly configured applications over HTTP using TCP/IP with SOAP protocols.

You can also install and use the J2EE client support from WebSphere Application Server Network Deployment, version 6, including Web services Client, EJB Client, and JMS Client.

Mediation modules

Mediation modules are Service Component Architecture (SCA) modules that can change the format, content or target of service requests.

Mediation modules operate on messages that are in flight between service requesters and service providers. They allow you to route messages to different service providers. They also let you transform messages: you can amend message content or form. In addition, mediation modules can provide functions such as message logging, and error processing that is tailored to your requirements.

You can change certain aspects of mediation modules dynamically, from the WebSphere Process Server administrative console, without having to redeploy the module.

Components of mediation modules

Among the items that mediation modules contain are the following:

- Imports
  - Imports define interactions between SCA modules and service providers.
  - Imports allow SCA modules to call external services as if they were local.
  - Mediation module imports can be viewed from WebSphere Process Server, and if the import binding is a Web service or SCA binding then it can be modified.
- Exports

- Exports define interactions between SCA modules and service requesters.
- Exports allow an SCA module to offer a service. Exports define the external interfaces (access points) of an SCA module.
- Mediation module exports can be viewed from WebSphere Process Server.

- SCA components
  - SCA components, or service components, are SCA building blocks. You build SCA modules such as mediation modules, using SCA components. You can create and customize SCA modules and components graphically, using WebSphere Integration Developer. In addition, you can customize some of the properties of SCA modules dynamically, from WebSphere Process Server administrative console, without having to redeploy the module.
  - Typically, mediation modules contain a specific type of SCA component called a mediation flow component. Mediation flow components define mediation flows. A mediation module can contain, at most, one mediation flow component.
  - A mediation flow component can contain one mediation primitive, a number of mediation primitives or no mediation primitives. WebSphere Process Server supports a supplied set of mediation primitives that provide functionality for message routing and transformation. One of the mediation primitives that WebSphere Process Server supports allows you to invoke custom logic.
  - A mediation module does not have to contain a mediation flow component. The purpose of a mediation module that does not contain a mediation flow component is to transform service requests from one protocol to another. For example, a service request might be made using SOAP/JMS but need transforming to SOAP/HTTP before sending on.

Note: You can view mediation modules from WebSphere Process Server. You can also make certain changes to mediation modules from WebSphere Process Server. However, you cannot view or change the SCA components from inside a WebSphere Process Server module. Use WebSphere Integration Developer to customize SCA components.

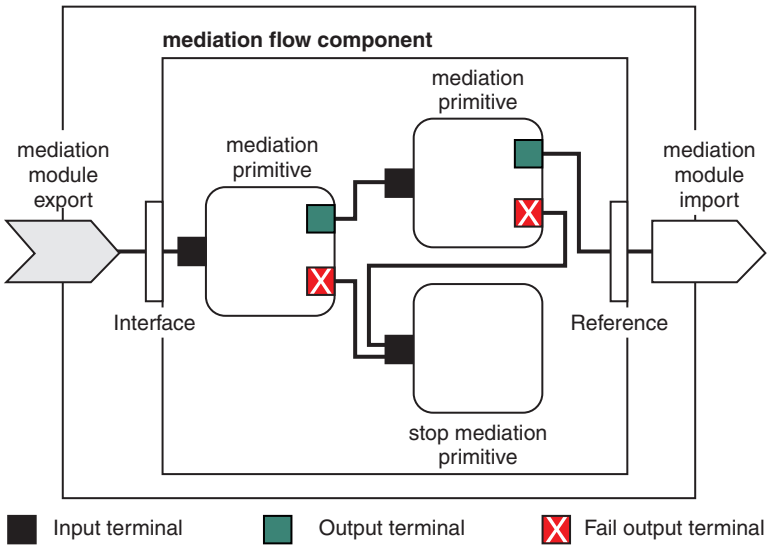


Figure 2. Simplified example of a mediation module. The mediation module contains one mediation flow component. The mediation flow component contains mediation primitives.

- Properties

- Mediation primitives have properties, some of which can be displayed in the administrative console as additional properties of an SCA module.
- In order for mediation primitive properties to be visible from the WebSphere Process Server administrative console the integration developer must flag the properties as promoted. Certain properties lend themselves to being administratively configured and WebSphere Integration Developer describes these as promotable properties, because they can be promoted from the integration cycle to the administrative cycle. Other properties are not suitable for administrative configuration, typically because modifying them affects the mediation flow in such a way that you need to redeploy the mediation module. WebSphere Integration Developer lists the properties that you can choose to promote under the `Promoted Properties` of a mediation primitive.
- You can use the WebSphere Process Server administrative console to change the value of promoted properties without having to redeploy a mediation module, or restart the server or module. New invocations of mediation flows use property changes immediately, unless the changes occur in a deployment manager cell. If changes occur in a deployment manager cell then they take effect after all nodes in the cell have been synchronized. Inflight invocations of mediation flows continue to use previous values.

  Note: If you want to change the property names and types of mediation primitives, and not the property values, you should use WebSphere Integration Developer.

Deploying mediation modules

Mediation modules are created using WebSphere Integration Developer, and are generally deployed to WebSphere Process Server inside an EAR (enterprise archive) file.

You can change the value of promoted properties at deployment time.

You can export a mediation module from WebSphere Integration Developer, and cause WebSphere Integration Developer to package the mediation module inside a JAR (Java archive) file, and the JAR file inside an EAR file. You can then deploy the EAR file, by installing a new application from the administrative console.

Mediation modules can be thought of as one entity. However, SCA modules are defined by a number of XML files stored in a JAR file.

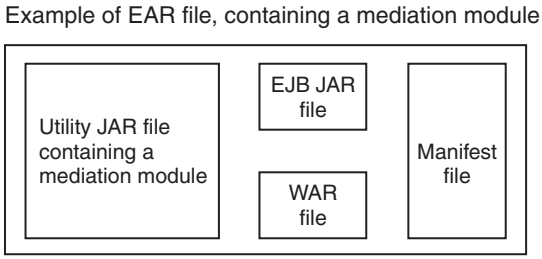


Figure 3. Simplified example of an EAR file containing a mediation module. The EAR file contains JARs. The utility JAR file contains a mediation module.

Mediation primitives

Mediation components operate on message flows between service components. The capabilities of a mediation component are implemented by *mediation primitives*, which implement standard service implementation types.

A mediation component has one or more flows. For example, one for request and one for reply.

WebSphere Process Server supports a supplied set of mediation primitives, which implement standard mediation capabilities for mediation modules deployed into WebSphere Process Server. If you need special mediation capabilities, you can develop your own custom mediation primitives.

A mediation primitive defines one "in" operation that processes or handles messages that are represented by service message objects (SMO). A mediation primitives can also define "out" operations that send messages to another component or module.

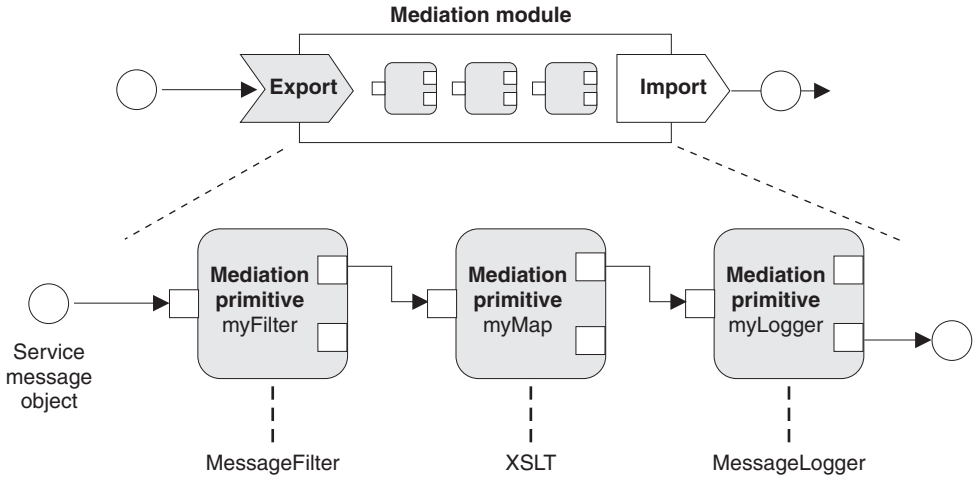


Figure 4. Mediation module containing three mediation primitives

Mediation primitives typically function at the level of a single operation, with possible mediation of the request and response. In some cases, you can specify mediation primitives down to the level of a single parameter on an operation. For example, selectors can operate at the operation level or parameter level.

You can use WebSphere Integration Developer to graphically model and assemble mediation components from mediation primitives, and assemble mediation modules from mediation components.

Supported mediation primitives

The following set of mediation primitives are supported by WebSphere Process Server:

Custom Mediation
Runs custom logic. The Custom Mediation primitive can also call an external Service Component Architecture (SCA) component that you provide.

- The operation that is called must be a two-way operation.
- The target SCA component must exist in the same mediation module as the Custom Mediation primitive.

Database Lookup
Modifies messages, using information from a user-supplied database.

- You must set up a database, datasource and any server authentication settings for the Database Lookup mediation primitive to use.
- The Database Lookup mediation primitive can read from only one table.
- The specified key column must contain a unique value.
- The data in the value columns must be either a Java primitive or a Java String (or be able to be cast to a Java primitive or a Java String).

Endpoint Lookup
Allows for dynamic routing of requests, by searching for service endpoints in a repository.

- Service endpoint information is retrieved from a WebSphere Service Registry and Repository (WSRR), which can be local or remote.
- You can make registry changes from the Registry administrative console.

Event Emitter
Enhances monitoring, by emitting events from within a mediation flow.

- The events are sent in the form of Common Base Events (CBE) and are sent to a Common Event Infrastructure (CEI) server.
- To fully utilize Event Emitter information, event consumers need to understand the structure of the CBE. The CBE has an overall schema but this does not model the application specific data, contained in the extended data elements. To model the extended data elements, the WebSphere Integration Developer tools generate a CEI event catalog definition file for each of the configured Event Emitter mediation primitives. Event catalog definition files are export artifacts that are provided to help you, they are not used by WebSphere Integration Developer or by the WebSphere Process Server runtime. You should refer to the event catalog definition files when you create applications to consume Event Emitter events.
- You can specify other monitoring from WebSphere Process Server. For example, you can monitor events to be emitted from imports and exports. What the Event Emitter mediation primitive does is let you send events from inside a mediation flow component. You can then view Event Emitter events using the CBE browser on WebSphere Process Server.

Fail Generates a failure in the flow.

Message Element Setter
Provides a simple mechanism for setting the content of message headers or bodies, it does not change the type of the message.

Message Filter
Routes messages down different paths, based on the message content.

Message Logger
Logs messages in a database. The messages are stored as XML, therefore data can be post-processed by XML-aware applications.

- The database schema is defined by IBM.

- On z/OS®, the installation of WebSphere Process Server creates an application server, and a sample database and datasource. The Sample Message Logger mediation primitive can be configured to use either a Cloudscape™ or a DB2® database
- If you want to create your own database and datasource, using the administrative console, then WebSphere Process Server provides data definition language (ddl) files that describe the table schema. The Table.ddl files are stored in: *install_root*/util/EsbLoggerMediation/*database_ type*/Table.ddl. Where *database_ type* refers to the type of database, for example, CLOUDSCAPE_V50. If you create your own database and want to use the default JNDI name for your datasource, then you must remove the default datasource.

Stop Stops a particular path in the flow, without generating an exception.

XSLT Transforms messages.

- The XSLT mediation primitive can change the headers or the body of your messages.
- You transform messages using an XSLT (Extensible Stylesheet Transformations) 1.0 transformation. The transformation operates on an XML serialization of the message.

SplitPath
Select the target service (or another mediation), route to a specific target, modify the routing path ...

BOMapper
Select the target service (or another mediation), route to a specific target, modify the routing path ...

Service message objects

Service Message Objects (SMOs) are enhanced Service Data Objects (SDOs). SMO provides an abstraction layer for processing and manipulating messages exchanged between services.

SMO model

The SMO model is a pattern for using SDO DataObjects to represent messages. The SMO contains a representation of the following groups of data:

- The business payload of the message. The payload is the application data exchanged between service endpoints.
- Header information associated with the message. For example, Java Message Service (JMS) headers if a message has been conveyed using the JMS API, or MQ headers if the messages has come from WebSphere MQ.
- Context information (data other than the message payload).

All of this information is accessed as SDO DataObjects, and there is a schema declaration that specifies the overall structure of the SMO. The schema is generated by the WebSphere Integration Developer.

All SMOs have the same basic structure. The structure consists of a root data object called a ServiceMessageObject, which contains other data objects representing header, body and context data. The SMO body contains the message payload. The headers contain information that originates from a specific import or export binding. For example, a JMS Binding.

SMO provides an interface to access and modify message headers and message payloads. SMO can represent the logical content of many different types of message.

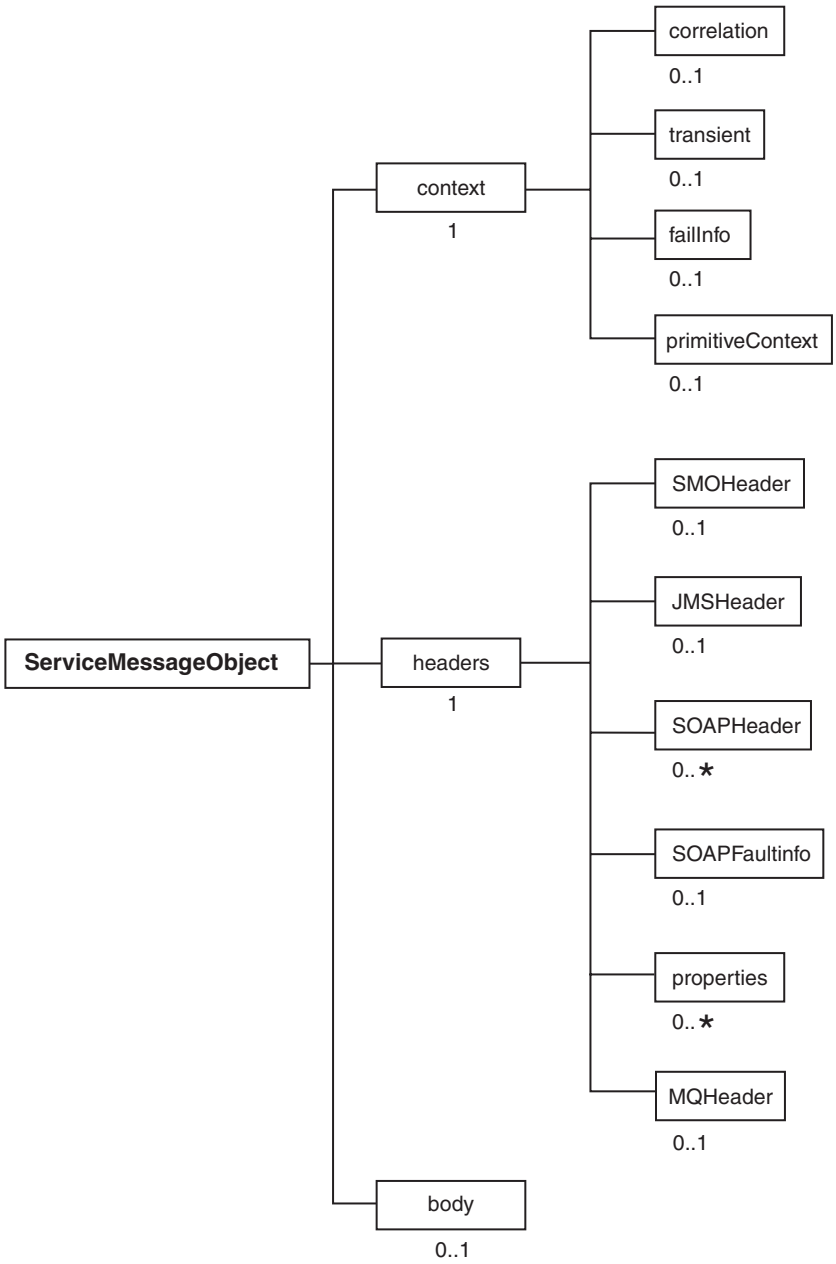


Figure 5. Overview of SMO structure. The headers, context and body of a ServiceMessageObject

WebSphere Process Server and SMO

WebSphere Process Server operates on messages that are in flight between interaction endpoints. Within WebSphere Process Server, mediation flows process messages as SMOs.

To provide dynamic routing, the interaction endpoints can be looked up using WebSphere Service Registry and Repository (WSRR): the result of the WSRR query is stored in the SMO.

Messages can come from a number of sources, so the SMO has to be able carry different kinds of message header. The kinds of message headers handled by WebSphere Process Server are:

- Web Services message headers.
- Service Component Architecture (SCA) message headers.
- Java Message Service (JMS) message headers.
- WebSphere Adapter message headers.
- WebSphere MQ message headers

WebSphere Process Server SMO runtime

WebSphere Process Server creates SMO objects, which are then available to mediation flows.

When creating mediation flows, WebSphere Integration Developer specifies the type of message body for each terminal (input, output or fail) and, optionally, the type of context information. WebSphere Process Server uses this information to convert messages into SMO objects of the specified type.

MQ header data in the SMO

The SMO structure stores MQ header information at /headers/MQHeader. This contains three elements:

- `md`, which represents the message's MQ message descriptor (MQMD)
- `control`, which contains format information relating to the message body
- `header`, which optionally repeats, and represents header structures contained within the WebSphere MQ message

The `md` elements contains all the fields from the MQMD definition (see the WebSphere MQ documentation), except for certain control fields which carry no useful data (such as `StrucId` and `Version`) and message format fields (`Encoding`, `CodedCharSetId` and `Format`).

The `control` element carries the `Encoding`, `CodedCharSetId` and `Format` fields, which describe the message body. If the WebSphere MQ message contains any message headers (for example, MQRFH2), the `Encoding`, `CodedCharSetId` and `Format` fields that describe the header are carried within the `header` element.

A `header` element represents a message header, such as an MQRFH2. Each header is one of the following:

- A header explicitly modelled in the SMO: an RFH version 1 or 2
- A header following the standard MQ header structure, but not explicitly modeled. These have MQ format identifiers beginning `MQH`. These headers are represented as unstructured binary data in the SMO.
- A header handled by a user-supplied MQ header data binding.

The `header` element contains `Encoding`, `CodedCharSetId` and `Format` fields, which describe the header. The `Format` field, in particular, must be set correctly (for example, `MQHRF2` for an MQRFH2 header); and the `CodedCharSetId` and `Encoding` fields are important for opaque data. When rendered as a WebSphere MQ message,

this format information is written into the previous MQ header (or into the MQMD if there is no previous header).

Header subelements

The `header` element also contains subelements (`rfh` and `rfh2`) for each of the modeled headers; a subelement (`opaque`) for unmodeled standard MQ headers; and a subelement (`value`) for user-supplied MQ headers. Precisely one of these four elements must be set: it is an error to have more than one of these set in any header element. The `value` subelement stores the structure used by the user-supplied header data binding; the other three elements (`rfh`, `rfh2` and `opaque`) are described in the following sections.

RFH headers

A WebSphere MQ `RFH` header contains a string of name-value pairs, where each name and value is simply a text string. This is represented, in SMO, as a repeating property element, containing a name and a value element.

RFH2 headers

A WebSphere MQ `RFH2` header contains zero or more named folders, each of which contains a sequence of properties and groups. A property has a name, optional type and value (all represented as `string`). A group has a name and itself contains a sequence of properties and groups. The SMO representation of an RFH2 header also contains a `NameValueCCSID` element, which determines the CCSID used to encode the folders in the WebSphere MQ message.

Unmodeled standard WebSphere MQ headers

The `opaque` element represents any WebSphere MQ header of the standard structure. Fields common to all such headers (`StrucId`, `Version` and `Flags`) are represented as elements. There is also a `data` element which contains the unmodeled portion of the header as `hexBinary` data. When using the `opaque` element, it is usually important to keep the correct `Encoding` and `CodedCharSetId` values associated with the header to avoid data corruption.

Types for WebSphere MQ header information

The WebSphere MQ header fields are defined using the same set of types used by WebSphere MQ itself. `MQLONG` fields are represented as `int`; `MQBYTEnn` fields as `hexBinary` data limited to *nn* in length; and `MQCHARnn` fields as `string` data limited to *nn* characters in length.

Management of the server and bus environment

With WebSphere Process Server, administrators can create an environment of ESB servers and service integration buses that support the deployment of service applications.

Managing the bus environment includes setting up WebSphere Process Server, sometimes as part of a larger system, typically as a production environment or realistic test environment. This includes some installation and customization activities, bus topology planning, and creating product configurations. The focus is on the administration of service integration buses, servers, and how to balance workload through clustering and high availability configurations.

If you create a Complete (default) installation for WebSphere Process Server, you get a default profile which includes a stand-alone server on which you can deploy Service Component Architecture (SCA) modules without having to do any configuration of the server.

You can choose to create a WebSphere Process Server deployment manager profile to enable the use of multiple servers and server clusters. You can start with one server in the network deployment cell and optionally add capacity and enhanced availability by build up to multiple servers or server clusters.

Use the administrative console or line commands to manage the server and bus environment. In the administrative console, each task is supported by one or more pages. To restrict the pages to those that support managing the bus environment, select **Server and Bus** in the**Task filtering selector** on the Welcome page of the administrative console. This displays pages suitable for the following tasks:

- Listing the service integration buses, servers, server clusters, messaging engines, and local topologies needed to support the deployment of mediation modules and service applications
- Enabling and disabling infrastructure services
- Installing applications and mediation modules
- Creating resources (for example, JMS connection factories and Common Event Infrastructure profiles) needed by deployed service applications and mediation modules
- Operational control of the ESB runtime

This is most appropriate for an administrator interested in managing the server and bus environment needed to support the deployment of service applications and mediation modules. This includes defining the network and bus topology, defining appropriate resources and monitoring the runtime system and troubleshooting any runtime errors.

To display all the administrative console panels, select **All** in the **Task filtering selector** of the Welcome page. This displays all administrative console panels. This is most appropriate for an administrator interested in managing all parts of WebSphere Process Server and the underlying WebSphere Application Server.

Besides the administrative console, you can use commands and scripts and administrative programs to manage the server and bus environment.

Related concepts

"Overview of the bus environment" on page 21
The bus environment comprises one or more service integration buses, ESB servers, and their resources, organized into logical administrative domains of cells and nodes.

Developing and deploying service applications

You can develop service applications that, when deployed onto WebSphere Process Server, enable message-based communication between services and can operate on and manipulate messages in flight between interaction endpoints.

Developing service applications

WebSphere Integration Developer is the separate development environment for WebSphere Process Server. You can use WebSphere Integration Developer to

graphically model and assemble mediation components from mediation primitives, and assemble mediation modules from mediation components.

- If the interface for SCA mediation components are not imported, you can use the Simplified Interface Editor to create the interface. You can use this editor to specify and edit interfaces (operations and parameters) of mediation modules.
- You can use the Mediation Flow Editor to map between operations on the end points of a mediation, to define the set of mediation flows needed for this application. You can use a set of predefined mediation primitives to visually compose a mediation flow.
- You can use the Business Object Editor to construct the messages that are used in mediations.
- You can use other editors to extend the development environment to meet your business needs; for example:
  - Create and edit custom mediation primitives, and add them to the Mediation Flow Editor.
  - Create and edit message descriptors.

You can develop some service components using other application development tools then import them into WebSphere Integration Developer for modelling, editing, testing, and packaging for deployment into WebSphere Process Server.

Deploying service applications

Deploying is the act of enabling your applications in either a test or a production environment. While the concept of deploying is the same for both environments, there are a few differences between the deployment task in each environment.

After developing a service application, best practices state that you should deploy the application onto a test server for testing before committing it to the production environment. Use WebSphere Integration Developer to deploy the applications into a test environment, and to package a service application as a standard enterprise application package, for deployment into WebSphere Process Server.

Use WebSphere Process Server to install and deploy the applications into a production environment. In WebSphere Process Server, you can use the standard WebSphere administrative console, with role-based administration views that simplify the experience for solution administrators to deploy and manage the components of service integration packages.

Related concepts

Chapter 8, “Development and deployment of integrated applications,” on page 35

Options for development and deployment of integrated applications on WebSphere Process Server include working in the WebSphere Integration Developer development environment, working with Service Component Architecture APIs, and enabling the applications in a test or production server environment using WebSphere Process Server.

Chapter 8. Development and deployment of integrated applications

Options for development and deployment of integrated applications on WebSphere Process Server include working in the WebSphere Integration Developer development environment, working with Service Component Architecture APIs, and enabling the applications in a test or production server environment using WebSphere Process Server.

IBM WebSphere Integration Developer is the development environment for WebSphere Process server. For more information about developing integrated applications in WebSphere Integration Developer, see the IBM WebSphere Business Process Management Version 6.0 information center.

In addition to the WebSphere Integration Developer development environment, Service Component Architecture APIs are published for developers. For more information about Service Component Architecture APIs, see the WebSphere Process Server *Developing and Deploying Modules* PDF file.

Modules, also called Service Component Architecture (SCA) modules when deployed to WebSphere Process Server, determine what artifacts are packaged in enterprise archive (EAR) files that are deployed to the runtime environment.

Within WebSphere Integration Developer, you can use an assembly editor to group services into modules and specify which services are exposed by the module to outside consumers. The modules are then connected to form complete integration solutions.

Service Component Architecture enables you to encapsulate integration logic within modules so that a change to services within a module will not affect any of the other modules in the solution as long as the interface of the changed module stays the same.

Deploying is the act of enabling your applications in either a test or a production environment. While deploying is the same for both environments, there are a few differences between the deployment task in each environment. Because it is best to test any changes to your applications on a test server before committing them to the production environment, use WebSphere Integration Developer to deploy the applications into a test environment before using WebSphere Process Server to deploy the applications into a production environment.

If you need to deploy many application files, you may want to use a batch file. For more information about batch files, see the WebSphere Process Server *Developing and Deploying Modules* PDF file.

Related concepts

"Developing and deploying service applications" on page 32
You can develop service applications that, when deployed onto WebSphere Process Server, enable message-based communication between services and can operate on and manipulate messages in flight between interaction endpoints.

Chapter 9. Administration of applications on WebSphere Process Server

Administering IBM WebSphere Process Server involves preparing, monitoring, and modifying the environment into which applications and resources are deployed, and working with the applications and resources themselves.

For more information about administering applications, see the *Administering* PDF file.

WebSphere Process Server offers several interfaces for administering the runtime environment:

- Administrative console

  The administrative console is a browser-based interface where you can monitor, update, stop, and start a wide variety of applications, services, and resources for the applications running on WebSphere Process Server. The administrative console can also be used to work with relationships and to locate and resolve failed WebSphere Process Server events.

  The administrative console also provide administration capabilities for WebSphere Application Server and other customer-defined products.

  For more information about the administrative console, see the *Administering* PDF file.

- Business Process Choreographer Explorer

  Business Process Choreographer Explorer is a stand-alone Web application that provides a basic set of administration functions for managing business process and human tasks. You can view information about process templates, process instances, task instances, and their associated objects. You can also act on these objects; for example, you can start new process instances, repair and restart failed activities, manage work items, and delete completed process instances and task instances.

- Business Process Choreographer Observer

  Business Process Choreographer Observer is a Web application that creates reports about events that occur during the execution of business processes and human tasks. You can then use these reports to evaluate the effectiveness and reliability of your processes and activities.

- Scripting program

  The WebSphere administrative (wsadmin) scripting program is a non-graphical command interpreter environment that enables you to run administrative options in a scripting language and to submit scripting language programs for execution. It supports the same tasks as the administrative console. The wsadmin tool is intended for production environments and unattended operations.

- Command-line tools

  Command-line tools are simple programs that you run from an operating system command-line prompt to perform specific tasks. Using these tools, you can start and stop application servers, check server status, add or remove nodes, and other tasks. The WebSphere Process Server command-line tools include the

serviceDeploy command, which processes .jar, .ear, .war and .rar files exported from a WebSphere Integration Developer environment and prepares them for installation to the production server.

- Administrative programs

  A set of Java classes and methods under the Java Management Extensions (JMX) specification provide support for administering Service Component Architecture (SCA) and business objects. Each programming interface includes a description of its purpose, an example that demonstrates how to use the interface or class, and references to the individual method descriptions.

Chapter 10. Security on WebSphere Process Server

IBM WebSphere Process Server provides a security infrastructure and mechanisms based on IBM WebSphere Application Server security.

For more information about security, see the WebSphere Process Server *Securing Applications and their Environment* PDF file.

Chapter 11. Monitoring on WebSphere Process Server

You monitor events in WebSphere Process Server to assess problem determination, to tune performance, and to measure the effectiveness of your business processes.

IBM WebSphere Process Server event monitoring capabilities include performance monitoring and service component monitoring.

Monitoring performance: Performance measurements are available for service component event points, and are processed through the Performance Monitoring Infrastructure (PMI) and the Tivoli® Performance Viewer.

You can monitor specific performance measurements for a given event, such as the number of times the event is invoked or the length of time it takes for that event to complete from start to finish. You can also monitor events, and later view their contents, either by viewing the events in a log file or by querying the events stored on the event database. In both cases, you can temporarily specify an event point or points to monitor in order to spot problems with the application logic or with system performance.

Monitoring service component events: WebSphere Process Server monitoring can capture the data in a service component at a certain event point. These events are formatted in a standard called the Common Base Event. You can have the process server publish these events to the logging facilities, or you can use the more versatile monitoring capabilities of a Common Event Infrastructure server database to store and analyze these events.

Some applications that run on the process server include event points that are monitored continually after the application is deployed. You can do this if you are a business analyst and want to observe the effectiveness of the business processes you have modeled and implemented in the applications you deployed on the process server. This allows you to use products, such as IBM WebSphere Business Monitor, to create customized panels -- or ″dashboards″ -- to view key business process metrics.

For more information, see the WebSphere Process Server *Monitoring* PDF file.

Chapter 12. Samples and tutorials

To help you learn how to accomplish your goals with IBM WebSphere Process Server, educational materials that include tutorials and samples are available.

Tutorials and samples are available from the IBM Education Assistant and the Samples Gallery, and tutorials for administrative tasks are available in the IBM WebSphere Process Server information center.

Business Process Choreographer Samples are available directly at http://publib.boulder.ibm.com/bpcsamp/index.html.

Tutorials

Tutorials for common tasks are available in the IBM Education Assistant and in the WebSphere Process Server documentation.

IBM Education Assistant tutorials

The IBM Education Assistant site provides tutorials that you can use at your convenience. To view this educational content, see the IBM Education Assistant: WebSphere Business Process Management Web page.

WebSphere Process Server tutorials

IBM WebSphere Process Server documentation contains tutorial topics to assist you with some administrative, security, and monitoring tasks.

For tutorials on WebSphere Process Server administrative, security, and monitoring tasks, refer to the *Administering WebSphere Process Server* PDF file, the *Securing Applications and their Environment* PDF file, and the *Monitoring* PDF file.

Accessing the Samples (Samples Gallery)

Samples of integration application artifacts are available in the Samples Gallery.

The Samples Gallery contains samples of simple artifacts such as those generated by IBM WebSphere Integration Developer and deployed on IBM WebSphere Process Server.

To install and view the WebSphere Process Server Samples Gallery, perform the following steps.

1. Unload the contents of the WebSphere Process Server for z/OS install media. The samples package is loaded onto the system when you unload the installation media. Run the install script to create the symlinks to the files in the samples directory

   The samples are installed in the *install_root*/samples directory.

   Applications that are run on WebSphere Process Server have XML artifacts, such as business objects, relationship definitions, and business rules, which must be deployed before installing the application. WebSphere Process Server provides a utility named serviceDeploy to build and deploy these artifacts. The enterprise archive (EAR) file in *install_root*/samples/lib for each sample

application contains these artifacts. The sampleDeploy utility invokes serviceDeploy with specific parameters required for the samples. Running sampleDeploy creates a second EAR file named *sample_ name*Deployed.ear in the same directory as the original EAR file. This new EAR file contains the Web archive (WAR) files that were in the original EAR file plus the additional Java archive (JAR) and WAR files that contain the deployed artifacts. The deployed EAR file can be installed as an enterprise application in WebSphere Process Server.

For more information, see Creating an installable EAR file using serviceDeploy in the WebSphere Process Server for z/OS *Developing and Deploying Modules* PDF file.

2. Start the server.
3. The Samples Gallery can be accessed directly from a web browser with the link http://xxxxxxxx:9080/WSsamples/en/index.html, where xxxxxxxx represents a host name or IP address of the target z/OS system. The actions of deploying and installing each sample into the server is part of the samples themselves. WebSphere Process Server samples are initially listed as installable samples in the Samples Gallery.
4. Click each sample that you want to install and follow the instructions that appear in the browser window for installation of each sample.
5. Click **Refresh** in the Samples Gallery and the samples are listed as installed samples. For each installed sample, you can click the sample name to open a browser window with additional information and an option to run the sample.
6. If you are running the samples in a distributed WebSphere Process Server environment without clustering, complete the following steps.
   a. On the machine with the deployment manager node, run the command *install_root*/samples/bin/installwbi -node *node_name* -server *server_name* -samples SamplesGallery WBISamplesGallery.
   b. In the administrative console, expand **Applications**, click **Enterprise Applications**, and start the SamplesGallery and WBISamplesGallery.
   c. Open a browser to access the Samples Gallery at http://*host_name*:*host_port* /WSsamples/index.jsp.
   d. Follow the instructions in the Samples Gallery to install and run each sample, making sure to use the **-node** *node_name* **-server** *server_name* parameters with the installwbi command.
7. If you are running the samples in a distributed WebSphere Application Server environment with clustering, complete the following steps.
   a. In the administrative console, expand **Applications** and click **Install New Application**.
   b. Click the browse button and locate the SamplesGallery.ear file in *install_root*/samples/lib/SamplesGallery directory.
   c. Install the EAR file, accepting all defaults, except for the target mapping panel, where you can designate a server or cluster on which to install the Samples Gallery.
   d. Repeat the previous steps for the WBISamplesGallery.ear file in the *install_root*/samples/lib/SamplesGallery.
   e. Start the applications that you just installed.
   f. Open a browser to access the Samples Gallery at http://*host_name*:*host_port* /WSsamples/index.jsp.
   g. Follow the instructions in the Samples Gallery to install and run each sample, but use **Install New Application** on the administrative console

instead of the installwbi command, which does not support clusters. Locate the deployed EAR files in the *install_root*/samples/lib/*sample_name* directory for each sample.

Chapter 13. Standards compliance

This product is compliant with several government and industry standards, including accessibility standards, information processing standards, software download security standards, and internet protocol standards.

Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

This product uses standard Windows navigation keys.

Accessibility features for WebSphere Process Server

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in WebSphere Process Server. The accessibility features include the following functions:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.

Operating system features that support accessibility are available when you are using WebSphere Process Server

Tip: The WebSphere Process Server Information Center is accessibility-enabled for screen reader software, including IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard navigation

This product uses standard Web browser navigation keys and standard Installshield Multiplatform navigation keys.

(For information about supported Web browsers, see the WebSphere Process Server System Requirements at http://www.ibm.com/software/integration/wps/sysreqs/.)

Interface information

- Installation

  You install WebSphere Process Server for z/OS silently from a command line. This silent installation meets accessibility needs.

  For instructions, see "Running the installation script" in the *Installing and Configuring WebSphere Process Server* PDF file.

  Note: The WebSphere Process Server installer program does not support the Installshield Multiplatform console mode.
- Administration

The administrative console is the primary interface for interacting with the product. This console is displayed within a standard Web browser. By using an accessible Web browser, such as Microsoft® Internet Explorer, administrators are able to:

- Use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Use voice recognition software, such as IBM ViaVoice®, to enter data and to navigate the user interface
- Operate features by using the keyboard instead of the mouse

You can configure and administer product features by using standard text editors and scripted or command-line interfaces instead of the graphical interfaces that are provided.

When appropriate, the documentation for specific product features contains additional information about the accessibility of the features.

Vendor software

WebSphere Process Server includes certain vendor software that is not covered under the IBM license agreement. IBM makes no representation about the accessibility features of these products. Contact the vendor for the accessibility information about its products.

Related accessibility information

See the *IBM Accessibility Center* for more information about the commitment that IBM has to accessibility.

Federal Information Processing Standard

Federal Information Processing Standards (FIPS) are standards and guidelines issued by the National Institute of Standards and Technology (NIST) for federal government computer systems.

WebSphere Process Server relies on IBM WebSphere Application Server for all cryptographic functions, which are compliant with Federal Information Processing Standards.

FIPS are developed when there are compelling federal government requirements for standards, such as for security and interoperability, but acceptable industry standards or solutions do not exist. Government agencies and financial institutions use these standards to ensure that the products conform to specified security requirements. For more information on these standards, see the National Institute of Standards and Technology.

WebSphere Application Server integrates cryptographic modules including Java Secure Socket Extension (JSSE) and Java Cryptography Extension (JCE), which have undergone FIPS 140-2 certification. In the WebSphere Application Server documentation, the IBM JSSE and JCE modules that have undergone FIPS certification are referred to as IBMJSSEFIPS and IBMJCEFIPS.

To enable FIPS, see the **Configuring Federal Information Processing Standard Java Secure Socket Extension files** topic in the WebSphere Application Server for z/OS documentation. When you enable FIPS, several components of the server are

affected including the cipher suites, the cryptographic providers, the load balancer, the caching proxy, the high availability manager, and the data replication service.

Common Criteria

The National Institute of Standards and Technology (NIST) has developed Common Criteria to ensure that you have a safe option for downloading software to use on your systems.

WebSphere Process Server is compliant with the Common Criteria standards developed by the National Institute of Standards and Technology.

Information held by information technology (IT) products or systems is a critical resource that enables organizations to succeed in their mission. Additionally, individuals have a reasonable expectation that their personal information contained in IT products or systems remain private, be available to them as needed, and not be subject to unauthorized modification. IT products or systems should perform their functions while exercising proper control of the information to ensure it is protected against hazards such as unwanted or unwarranted dissemination, alteration, or loss. The term IT security is used to cover prevention and mitigation of these and similar hazards.

Many consumers of IT lack the knowledge, expertise or resources necessary to judge whether their confidence in the security of their IT products or systems is appropriate, and they may not want to rely solely on the assertions of the developers. Consumers may therefore choose to increase their confidence in the security measures of an IT product or system by ordering an analysis of its security (in other words, a security evaluation).

For more information about Common Criteria, see the IBM WebSphere Software Support site, including Recommended Fixes for WebSphere Application Server and WebSphere Process Server.

Internet Protocol Version 6.0

IBM WebSphere Process Server relies on WebSphere Application Server for all Internet Protocol Version 6.0 compatibility.

IBM WebSphere Application Server Version 6.0.x and its JavaMail component support Internet Protocol Version 6.0 (IPv6).

For more information about this compatibility in WebSphere Application Server, see IPv6 support in the WebSphere Application Server for z/OS documentation.

For more information about IPv6, see www.ipv6.org.

Chapter 14. Globalization

Globalized products can be used without language or culture barriers and can be enabled for a specific locale.

WebSphere Process Server provides basic enablement support for all locales. Translations are provided for the following national languages:

- French
- Italian
- German
- Spanish
- Brazilian Portuguese
- Japanese
- Korean
- Simplified Chinese (GB18030 compliant)
- Traditional Chinese

In addition, WebSphere Process Server provides partial translations for the following national languages:

- Arabic
- Hebrew
- Czech
- Hungarian
- Polish
- Russian

For information about the globalizing applications and the internationalization service available through WebSphere Application Server, see WebSphere extensions in the WebSphere Application Server for z/OS documentation.

Bidirectional language support

The data that you process in WebSphere Process Server needs to be in the bidirectional language format of ILYNN (implicit, left-to-right, on, off, nominal), which is also the Windows® bidirectional language format. All other bidirectional language formats for applications you are running on WebSphere Process Server must be converted before being introduced to WebSphere Process Server.

Bidirectional attributes

There are five attributes that must be set for the correct bidirectional language format. The attributes and settings are listed in the following table.

Letter Position	Purpose	Values	Description	Default Setting
1	Order Schema	I or V	Implicit (Logical) or Visual	I
2	Orientation	L or R	Left-to-Right or Right-to-Left	L

Letter Position	Purpose	Values	Description	Default Setting
3	Symmetric Swapping	Y or N	Symmetric Swapping is on or off	Y
4	Shaping	Y or N	Text is shaped or not shaped	N
5	Numeric Shaping	H, C, N	Hindi, Contextual, Nominal	N

It is the responsibility of any client applications, external components (such as Web services, stateless session beans, and custom code), or anyone building solutions to run on WebSphere Process Server, to transform the data into the supported bidirectional language format.

For an example of bidirectional language transformation of a string, refer to Example: Using bidirectional transformation on string-type data.

For an example of bidirectional language transformation of a Service Data Object, refer to Example: Using the bidirectional transformation on DataObject-type data.

Note: The locale setting of the user interface (browser) defines the bidirectional language display and edit format.

For more information about bidirectional language, see the technical articles on IBM developerWorks, available at www.ibm.com/developerworks/websphere/library/techarticles/bidi/bidigen.html.

Example: Using the bidirectional transformation on DataObject-type data

When using information that is in a bidirectional language script, it might be necessary to transform the format of the data. This is a step-by-step example of the coding that performs a transformation on DataObject-type data.

The module requires that DataObject-type information is transformed from one bidirectional format to another.

Note: If you are unfamiliar with the formats, see "Values for the bidirectional format string" on page 55.

1. Include all bidirectional classes that contain the bidirectional engine implementation.

   ```
   import com.ibm.bidiTools.bdlayout.*;
   ```

2. Include all the classes you need to manipulate the DataObject-type object.

   ```
   import commonj.sdo.DataObject;
   import commonj.sdo.Type;
   import commonj.sdo.Property;
   ```

3. Define string variables to contain the different types of strings that a DataObject-type object contains. This step facilitates filtering the attributes of `type String` while transversing recursively the DataObject.

   ```
   String STRING_STR_TYPE = "String";
   String NORM_STRING_STR_TYPE = "normalizedString";
   String TOKEN_STR_TYPE = "token";
   ```

```
String LANG_STR_TYPE = "language";
String NAME_STR_TYPE = "Name";
String NMTOKEN_STR_TYPE = "NMTOKEN";
String NCNANE_STR_TYPE = "NCName";
String ID_STR_TYPE = "ID";
String IDREF_STR_TYPE = "IDREF";
String IDREFS_STR_TYPE = "IDREFS";
String ENTITY_STR_TYPE = "ENTITY";
String ENTITIES_STR_TYPE = "ENTITIES";
```

4. Define the function that verifies if the type of a property is String.

```
private static boolean isStringFamilyType (Property property) {
    boolean rc = false;
    if ((property.getType().getName().equalsIgnoreCase(STRING_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(NORM_STRING_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(TOKEN_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(LANG_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(NAME_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(NMTOKEN_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(NCNANE_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(ID_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(IDREF_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(IDREFS_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(ENTITY_STR_TYPE)) ||
    (property.getType().getName().equalsIgnoreCase(ENTITIES_STR_TYPE)))
       rc = true;
     return rc;
}
```

5. Define the recursive function that applies the bidirectional transformation on the entire DataObject.

   Note: The following explain basic assumptions that code logic follows.

   - bidirectional transformation is applied on properties of string type only
   - the properties of type string in the DataObject are stored in one bidirectional format

```
DataObject BiDiDataObjTransformationBO(DataObject boIn, String formatIn, String formatOut){

   Type type;
   Property property;

   if (boIn == null) return null;

   type = boIn.getType();
   List propertyList = type.getProperties();
   for (int propertyNumber = 0; propertyNumber < propertyList.size(); propertyNumber++){
        property = (Property) propertyList.get(propertyNumber);
        String propertyName = property.getName();
```

   a. Skip all non-string properties.

```
          if (!isStringFamilyType(property))
                continue;

          if (property.isContainment()) {
               if (property.isMany()) {
                     List childsList = boIn.getList(property);
```

   b. Recursively call the transformation to handle children objects.

```
                  for (int childNumber = 0; childNumber < childsList.size();
     childNumber++){                  BiDiDataObjTransformationBO(connectionContext,
   ((DataObject)childsList.get(childNumber)),formatIn, formatOut);
              }
           } else {
```

   c. Recursively call the transformation to handle children objects of any contained business objects.

```
                BiDiDataObjTransformationBO(connectionContext,
          ((DataObject)boIn.get(property)),formatIn, formatOut);
             }
          } else {
```

d. Transform the simple string attributes.

```
             String str = BiDiStringTransformation(
                 (boIn.getString(propertyName),formatIn, formatOut);
             boIn.setString(propertyName, str);
          }
        }
        return boIn;
    }
```

Example: Using bidirectional transformation on string-type data

When using information that is in a bidirectional language script, it might be necessary to transform the format of the data. This is a step-by-step example of the coding that transforms string-type data.

The module requires that string information is transformed from one bidirectional format to another.

Note: If you are unfamiliar with the formats, see "Values for the bidirectional format string" on page 55.

1. Include all bidirectional classes that contain the bidirectional engine implementation.

   ```
   import com.ibm.bidiTools.bdlayout.*;
   ```

2. Define the strings to contain the data object to transform, and the input and output format values.

   The input format is the bidirectional format in which the string object is currently stored. The output format is the bidirectional format in which you want to store the string object.

   ```
   String strIn = new String("Hello world");
   String formatIn = "ILYNN";
   String formatOut = "VLYNN";
   ```

3. Call the BidiStringTransformation function.

   ```
   String strOut = BiDiStringTransformation(strIn, formatIn, formatOut);
   String BiDiStringTransformation(String strIn, String formatIn, String formatOut) {
   ```

   a. Test if input string is null.

      ```
      if (strIn == null) return null;
      ```

   b. Perform transformation

      ```
      BidiFlagSet flagsIn;
      BidiFlagSet flagsOut;
      formatIn = formatIn.toUpperCase();
      formatOut = formatOut.toUpperCase();

      if (formatIn != null)
          flagsIn = new BidiFlagSet(formatIn.toCharArray());
      else
          flagsIn = new BidiFlagSet();

      if (formatOut != null)
          flagsOut = new BidiFlagSet(formatOut.toCharArray());
      else
          flagsOut = new BidiFlagSet();
      ```

```
  if (flagsIn.equals(flagsOut)) return strIn;
  String strOut = BiDiStringTransformation(strIn, flagsIn, flagsOut);
  return strOut;
}
```

Values for the bidirectional format string

The values for the bidirectional language format string control the transformation of bidirectional scripts from one format to another.

Purpose

Use the table below to determine the correct value for either the input string or output string to use when transforming String-type or DataObject-type data from one format to another.

Table 1. Bidirectional language format string values

Letter position	Purpose	Allowable values	Default value	Meaning
1	Ordering schema	I	I	Implicit
		V		Visual
2	Orientation	L	L	Left to right
		R		Right to left
		C		Contextual left to right
		D		Contextual right to left
3	Symmetric swapping	Y	Y	Symmetrical swapping is on
		N		Symmetrical swapping is off
4	Shaping	S	N	Text is shaped
		N		Text is not shaped
		I		Initial shaping
		M		Middle shaping
		F		Final shaping
		B		Isolated shaping
5	Numeric	H	N	Hindi (National)
		C		Contextual
		N		Nominal

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both: IBM, IBM (logo), AIX, CICS, Cloudscape, DB2, DB2 Connect, DB2 Universal Database, developerWorks, Domino, IMS, Informix, iSeries, Lotus, MQSeries, MVS, OS/390, Passport Advantage, pSeries, Rational, Redbooks, Tivoli, WebSphere, z/OS, zSeries

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

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Process Server for z/OS version 6.0.2

Printed in USA